I.J. Wireless and Microwave Technologies, **2024, 5, 31-43**
Published Online on October 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijwmt.2024.05.03

Development of a Phishing Website Detection Model Using Classification Algorithm

Olugbenga A. Madamidola*
Federal University of Technology, Akure Ondo State, Nigeria
Email: oamadamidola@futa.edu.ng
ORCID iD: https://orcid.org/0000-0001-5991-4067
*Corresponding Author

Ilobekemen P. Oladoja
Federal University of Technology, Akure Ondo State, Nigeria
Email: ipoladoja@futa.edu.ng
ORCID iD: https://orcid.org/0000-0002-1717-4985

Peace B. Falola
University of Ibadan, Nigeria
Email: peacefalola@gmail.com
ORCID iD: https://orcid.org/0000-0001-8581-4123

Matthew W. Omojola
Precious Cornerstone University, Ibadan, Nigeria
Email: williamsomojola83@gmail.com
ORCID iD: https://orcid.org/0009-0003-6157-8463

Received: 10 April, 2024; Revised: 24 June, 2024; Accepted: 15 July, 2024; Published: 08 October, 2024

Abstract: In the contemporary digital landscape, the proliferation of malware presents a significant threat to the security and integrity of computer systems and networks. Traditional signature-based detection methods are increasingly ineffective against the evolving landscape of sophisticated malware variants. Consequently, there is a pressing need for innovative approaches to malware detection that can adapt to emerging threats in real-time. This research aims to develop a malware detection system using machine learning algorithms. Random Forest classifier and Logistic regression were deployed for the classification of malware based on the features extracted from the CIC-MalMem-2022 dataset. The Malware detection system model was implemented using the Python programming language and evaluated using major performance metrics like F1-score, precision, recall, and accuracy to assess the model's performance. A comparison between the logistic regression model and the random forest model showed that the Random Forest model approach performed better than the logistic model in detecting malware, achieving accuracies of 98% and 94% respectively. In summary, the report concludes that the developed Malware Detection System using Machine Learning, specifically the Random Forest and Logistic regression models, shows promise in effectively detecting malware and highlights the importance of leveraging Artificial Intelligence for combating malware threats in the computing community.

Index Terms: Phishing, Classification Algorithm, Website Detection, Machine Learning, Regression Analysis, Malware

1. Introduction

The internet is an invaluable resource for information, knowledge, and learning, as well as a platform for conducting legitimate and convenient transactions. However, it also harbors cybercriminals who engage in malicious activities such as phishing, which has caused significant harm to private businesses and large industries. To protect users from these threats, developers and white-hat hackers continuously seek effective solutions. As cybersecurity attacks increase in both scope and sophistication, social engineering remains a primary method for accessing confidential information. [1] Phishing, as defined by the United States Computer Emergency Preparedness Team

(US-CERT), is a type of social engineering attack where personal information is solicited through malicious emails or websites that impersonate trusted organizations. It is one of the most prevalent and dangerous illegal activities conducted online.

The growth of internet usage for services provided by governments and financial organizations has led to a significant rise in phishing attempts in recent years. Phishers, motivated by financial gain, employ various techniques such as voice over IP (VoIP), spoofed links, chat, and fraudulent websites to deceive unsuspecting users. [2] Creating fake websites is relatively simple; they mimic legitimate sites in layout and information but aim to harvest personal data, including account numbers, debit and credit card passwords, and login credentials. One objective of phishing is to obtain sensitive information for identity theft or financial fraud. Attackers may also pose security questions to gain high-level security answers from users, thereby tricking them into divulging their details. [3]

Additionally, hackers install malicious software on computers to steal login credentials, exploiting these systems to capture users' online account information. Phishers use multiple methods, including text messages, phone calls, forum posts, emails, Uniform Resource Locators (URLs), and instant messages, to steal user information. Phishing content often closely resembles legitimate content, misleading users into accessing these malicious sites and unknowingly providing their private information. For an average internet user, identifying phishing sites can be challenging unless they know the specific URL, which can be time-consuming. To address this issue, this work aims to develop a machine-learning algorithm to help users and business organizations identify malicious URLs effectively.

2. Literature Review

In order to examine how well machine learning can detect phishing attempts. websites, [4] developed and examined four models. Additionally, it compared the four models' most accurate models with solutions that had already been published. These models were developed using random forest (RF), decision trees (DTs), support vector machines (SVMs), and artificial neural networks (ANNs). Furthermore, the models were evaluated against the UCI Phishing Domains dataset obtained from the Universal Resource Locator (URL) as a benchmark. Their findings showed that, among the four approaches, the random forest technique-based model is the most accurate and outperforms other solutions found in the literature.

Authors in [5] introduced the Phish-Sight framework, a machine-learning-based system that uses a visual inspection method to detect phishing websites. Phish-Sight use machine learning techniques well-known brand names injected into URLs' web pages to identify fraudulent websites. The most common color attributes as well as well-known trademarks from websites were predicted using five machine learning techniques. With a 99.13% accuracy rate and 98.43% of genuine positives, the Random Forest algorithm surpassed the competition in phishing scam detection. With an estimated execution time of 7.6 seconds per web page, Phish-Sight could have real-time applications.

In [6] authors compared and examined phishing websites with authentic ones using information gathered through an assessment of open-source platforms. The authors also suggested employing Decision Tree and Random Forest techniques to identify bogus websites. The poll, which had 30 participants, was conducted using Microsoft Form. The majority of participants lack basic knowledge, are vulnerable to phishing attacks, and do not review the interface's features before using the search browser. As part of the data collection for the survey, Decision Tree and Random Forest were trained and assessed to determine the best phishing website detection. The outcomes demonstrated that Random Forest performs the most favorable in regards of obtaining high rates of accuracy and feature importance identification.

In [7], authors incorporated hybrid URL and hyperlink-based elements into a machine learning-based approach for real-time phishing website detection, achieving high accuracy without requiring any external systems. Heuristic, visual similarity, blacklist, and whitelist-based anti-phishing solutions cannot detect newly launched websites or zero-hour phishing attempts. Furthermore, because they rely on external sources like search engines, older systems are complicated and inappropriate for real-time environments. Therefore, a significant difficulty in the field of cybersecurity is identifying recently created phishing websites in a real-time setting. They also generate a new dataset for testing using popular machine-learning categorization methods. Their test findings showed that the suggested phishing detection approach performed better than more established techniques, with an XG Boost technique-based detection accuracy of 99.17%.

Authors in [8] examined the use of feature selection algorithms for the fast and precise identification of phishing websites. A comparison of machine learning algorithms was done based on how effective they were both with and without feature selection. A phishing dataset of 30 attributes, 4898 phishing pages, and 6157 benign sites was used in the trials. The best results were achieved by combining several machine learning techniques. After that, a feature selection technique was used to increase the models' effectiveness. With a large reduction in model construction time, random forest was able to achieve the best accuracy both before and after feature selection.The result showed that using feature selection techniques in addition to Machine learning methods have the potential to accelerate the creation of phishing detection classification models while maintaining their accuracy.

In order to identify phishing assaults, authors in [9] used a number of machine learning algorithms. It is suggested to use two priority-based algorithms. The ultimate fusion classifier was selected in accordance with the output of these algorithms. They employed a novel fusion classifier on a information from the University of California, Irvine, and got a 97% accuracy rate. For the implementation of our experiments, they employed Python**.**

In 2020, authors in [10] suggested a method to use a stacking model to identify phishing websites. Feature selection algorithms such as information gain, gain ratio, Relief-F, and recursive feature elimination (RFE) are employed to analyze the attributes of phishing datasets. Two features were created by combining the best and worst attributes. Principal component analysis was performed on the recommended and residual features using a range of machine learning techniques, such as bagging, support vector machines, random forests, neural networks, k-nearest neighbor, and support vector machines. In order to improve classification accuracy, the top-scoring classifiers were then combined to apply two stacking models: Stacking1 (RF + NN + Bagging) and Stacking2 (kNN + RF + Bagging).The accuracy of each classifier was much improved by the suggested features. The results showed how important RFE is for eliminating the least important feature from the set of data. In terms of classification accuracy, Stacking1 (RF + NN + Bagging) performed better than any other classifier, identifying phishing websites with a 97.4% accuracy rate.

Authors in [11] evaluated the ability of six machine learning methods to recognize between phishing and trustworthy websites: Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), Logistic Regression (LR), and Naive Bayes (NB). There were two distinct training groups used to train these algorithms in the WEKA platform. Three different datasets with different percentages of phishing and non-phishing cases were used to assess the algorithms' accuracy, precision, TP rate, and FP rate. Results showed that Multi-Layer Perceptron had the lowest results in terms of detection accuracy, whereas Nave Bayes classifier had the best detection accuracy among other classifiers for detecting phishing websites. Additionally, the outcome demonstrated that Support Vector Machines have a higher FP rate than other classifiers. Additionally, all phishing websites may be accurately classified as phishing using Random Forest, Decision Tree, and Naive Bayes. This indicates that the TP rate for these classifiers is 100%. This study recommended utilizing NB as the top classifier for distinguishing between legitimate and fraudulent websites.

3. Methodology

3.1 Language

Python was used as the primary language in the development of this model. Python is a programming language that emphasizes machine learning extensively. It offers many libraries for machine learning that can be used directly by importing them. Due to its vast collection of machine learning libraries, Python is utilized all over the world to handle with machine learning.

3.2 Dataset

The dataset was downloaded from kaggle ,and this dataset was used to train the machine learning models with 549,346 good and malicious URL. The dataset is split into training and testing sets to evaluate the model's performance. A split ratio is 70% for training and 30% for testing. This ensures that the model is trained on a subset of the data and tested on unseen data to assess its generalization ability.

3.3 Data Preprocessing

Data pre-processing cleans up raw, unstructured data to create well-structured data. and tidy datasets. Preprocessing will be carried out on the dataset to ensure a good training accuracy.

In the context of malware detection, the following steps were likely involved:

a. Data Cleaning: This step involves handling missing values, outliers, and inconsistencies in the dataset to ensure data quality.
b. Data Transformation: Converting non-numeric data into a numeric format suitable for machine learning algorithms.
c. Normalization: Scaling the data to a standard range to prevent features with larger scales from dominating the model training process.
d. Feature Encoding: Converting categorical variables into numerical representations for model compatibility.
e. Feature Selection: Identifying and selecting relevant features that have a significant impact on the target variable to improve model performance.

3.4 Feature Extraction

Feature extraction involves selecting and transforming raw data into a format that is suitable for model training. In the context of malware detection, features related to file behavior, system calls, API calls, and other characteristics of malware samples may have been extracted from the dataset to train the machine learning models.. e.g To reduce the length of malicious URL ,attackers always use URL shortening service. In the case of Random Forest and Logistic Regression models, hyperparameters such as the number of trees in the Random Forest, maximum depth of trees, regularization parameter in Logistic Regression, etc., is tuned using techniques like grid search or random search to find the best combination for optimal model performance. Table 1 below shows features that will be used for this study. These features are important for detecting phishing domains.

Table 1. Features for detecting phishing domains

Address based Features	Domain of the URL URL IP Address Redirection of the URL("//') Http/Https in the Domain name Use of URL shortening Service Prefix or Suffix in the domain "-" Length of the URL Depth of the URL "@" Symbol in the URL
Domain based features	• Domain name system record • Web traffic • Age of the Domain • Period at which the domain will expire
HTML &Java script based Features	• I frame Redirection • Website Forwarding • Disabling right click

3.5 Machine Learning Models

Two supervised machine learning algorithms were used in this study, which are as follows:
1. Logistic Regression (LR)
2.Multinomial Naives bayes

3.5.1 Logistic Regression

Within the framework of Supervised Learning, among the most widely used Machine Learning algorithms is logistic regression. It is employed to predict, from a set of independent variables, the categorical dependent variable utilizing logistic regression to predict results of a dependent variable that is categorical. Consequently, the output needs to be discrete or categorical. It can be anything from True to False to Yes or No, and so on, but rather than providing exact values between 0 and 1, The probabilistic values it offers fall between 0 and 1. Observations can be categorized using a variety of data types, and logistic regression can be used to quickly identify the variables that work best for classification.

The well-known logistic regression machine learning technique is a component of the supervised learning methodology. This approach uses a series of independent variables to forecast the category dependent variable. The result of a categorical dependent variable is predicted using logistic regression. For this reason, the outcome needs to be discrete or categorical. Instead than giving the precise values of 0 and 1, it gives the probability values that fall between 0 and 1. It can be True or False, Yes or No, 0 or 1, and so on. Logistic regression is a useful tool for classifying observations based on different kinds of data. This approach finds the variables that are most useful for categorization quickly.

3.5.2 Multinomial Naives Bayes

The foundation of the Nave Bayes method, which is applied to several classification applications, is the Bayes Theorem. A straightforward mathematical formula for estimating conditional probabilities is the Bayes theorem. Conditional probability is the likelihood that an event will occur in light of the likelihood that another event has already occurred (based on an assumption, inference, claim, or piece of evidence).

3.6 Architecture of The System

Figure 1 shows the arcitecture of the system.

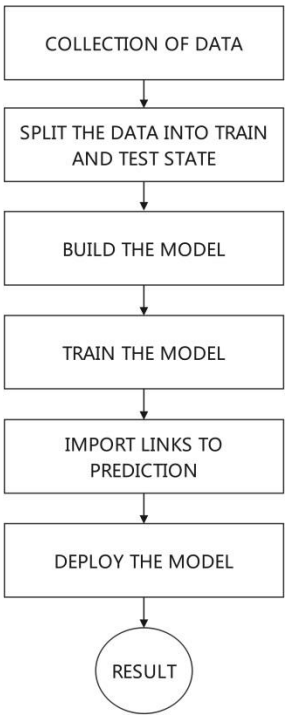


Fig. 1. Architecture of the system

4. Results and Discussion

The dataset was downloaded from Kaggle. It was in a csv (Comma-separated Values) format .There were 549,346 legitimate and malicious URL in the datasets. Then the dataset was preprocessed. Figure 4 shows a part of the processed data. Dataset was tokenized by dividing the long links or text into smaller pieces and also got the text stemmed by mapping a group of word to the same stem. Figure 2 shows a part of the dataset after it was tokenized. Figure 3 shows a part of the word stem. A graph was then drawn to differentiate the amount of good and bad URLs in the datasets. In the feature extraction file, the recovered features from the genuine and phishing URL datasets were simply concatenated, with no shuffling. To balance the distribution, the data was shuffled while dividing it into training and testing sets. This reduces the potential of the model over-fitting during training. Visualization was done using word cloud

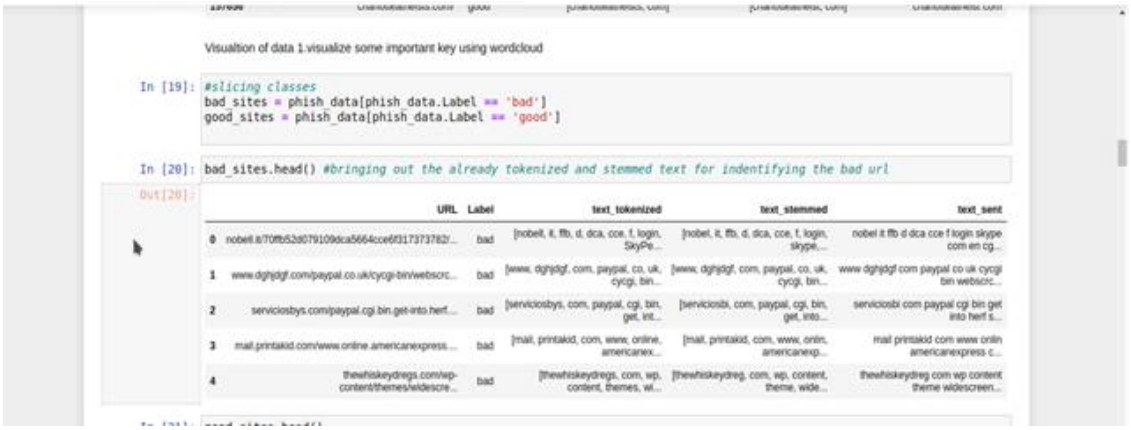
```
In [12]: print('Getting words tokenized ...')
         t0= time.perf_counter()
         phish_data['text_tokenized'] = phish_data.URL.map(lambda t: tokenizer.tokenize(t)) # doing with all rows
         t1 = time.perf_counter() - t0
         print('Time taken',t1 ,'sec')

         Getting words tokenized ...
         Time taken 3.598875489000079 sec

In [13]: phish_data.sample(5)
```

Out[13]:

	URL	Label	text_tokenized
42860	pastehtml.com/view/bvu1fs4vc.html	bad	[pastehtml, com, view, bvu, fs, vc, html]
14460	paypula.altervista.org/login.doR/index.html	bad	[paypula, altervista, org, login, doR, index, ...
170105	en.wikipedia.org/wiki/1974_Montreal_Expos_season	good	[en, wikipedia, org, wiki, Montreal, Expos, se...
274997	amazon.com/Eye-Every-Storm-Neurosis/dp/B00029LNUA	good	[amazon, com, Eye, Every, Storm, Neurosis, dp,...
179271	en.wikipedia.org/wiki/Pedal_to_the_Metal_(Bles...	good	[en, wikipedia, org, wiki, Pedal, to, the, Met...

Fig. 2. Tokenized word

The process of breaking down text data into individual tokens or words for analysis. The input to the tokenization process is raw text data, which is in the form of sentences, paragraphs, or documents.



Visualtion of data 1.visualize some important key using wordcloud

```
In [19]: #slicing classes
bad_sites = phish_data[phish_data.Label == 'bad']
good_sites = phish_data[phish_data.Label == 'good']

In [20]: bad_sites.head() #bringing out the already tokenized and stemmed text for indentifying the bad url
```

Out[20]:

	URL	Label	text_tokenized	text_stemmed	text_sent
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad	[nobell, it, ffb, d, dca, cce, f, login, SkyPe...	[nobel, it, ffb, d, dca, cce, f, login, skype,...	nobel it ffb d dca cce f login skype com en cg...
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	bad	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...	www dghjdgf com paypal co uk cycgi bin webscrc...
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad	[serviciosbys, com, paypal, cgi, bin, get, int...	[serviciosbi, com, paypal, cgi, bin, get, into...	serviciosbi com paypal cgi bin get into herf s...
3	mail.printakid.com/www.online.americanexpress....	bad	[mail, printakid, com, www, online, americanex...	[mail, printakid, com, www, onlin, americanexp...	mail printakid com www onlin americanexpress c...
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad	[thewhiskeydregs, com, wp, content, themes, wi...	[thewhiskeydreg, com, wp, content, theme, wide...	thewhiskeydreg com wp content theme widescreen...

Fig. 3. Word Stemmed

The process of reducing words to their root or base form to normalize the text data, aiming to reduce inflected or derived words to their root form to capture the core meaning of words.

```
In [14]: stemmer = SnowballStemmer("english") # choose a language ,i used english

In [15]: print('Getting words stemmed ...')
t0= time.perf_counter()
phish_data['text_stemmed'] = phish_data['text_tokenized'].map(lambda l: [stemmer.stem(word) for word in l])
t1= time.perf_counter() - t0
print('Time taken',t1 ,'sec')

Getting words stemmed ...
Time taken 50.94475107100004 sec

In [16]: phish_data.sample(5)
```

Out[16]:

	URL	Label	text_tokenized	text_stemmed
277450	amazon.com/The-Devils-Brigade/dp/B001EYK71S	good	[amazon, com, The, Devils, Brigade, dp, B, EYK...	[amazon, com, the, devil, brigad, dp, b, eyk, s]
29499	paypal-services.de/km/paypal/verifikation/8330...	bad	[paypal, services, de, km, paypal, verifikatio...	[paypal, servic, de, km, paypal, verifik, inde...
147609	bankingcanada.net/desjardins+group/	good	[bankingcanada, net, desjardins, group]	[bankingcanada, net, desjardin, group]
225217	paraprint.com/	good	[paraprint, com]	[paraprint, com]
422357	reunion.com/jeffreyalper/	good	[reunion, com, jeffreyalper]	[reunion, com, jeffreyalp]

Fig. 4. Data Preprocessing

The data is further fed into the logistic regression and naive bayes models. Figures 5 and 6 show the training of the logistic regression and the result respectively. The training and testing accuracies for the logistic regression model were 96% and 97% respectively

```
In [37]: trainX, testX, trainY, testY = train_test_split(feature, phish_data.Label)
```

LOGISTIC REGRESSION

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on.

```
In [38]: # create lr object
lr = LogisticRegression()

In [39]: lr.fit(trainX,trainY)
Out[39]: LogisticRegression()

In [40]: lr.score(testX,testY)
Out[40]: 0.9629961335983748
```

LOGISTIC REGRESSION IS GIVING 96% accuracy, Now i will store scores in dict to see which model perform best

Fig. 5. Training of Logistic Regression model

The training of a Logistic Regression model involves preprocessing the data, normalizing the features, fitting the model to the training data, optimizing the cost function through gradient descent, and evaluating the model's performance on unseen data. This process helps in building a predictive model for classification tasks in the malware detection,

```
In [41]: Scores_ml = {}
         Scores_ml['Logistic Regression'] = np.round(lr.score(testX,testY),2)

In [42]: print('Training Accuracy :',lr.score(trainX,trainY))
         print('Testing Accuracy :',lr.score(testX,testY))
         con_mat = pd.DataFrame(confusion_matrix(lr.predict(testX), testY),
                     columns = ['Predicted:Bad', 'Predicted:Good'],
                     index = ['Actual:Bad', 'Actual:Good'])


         print('\nCLASSIFICATION REPORT\n')
         print(classification_report(lr.predict(testX), testY,
                                     target_names =['Bad','Good']))

         print('\nCONFUSION MATRIX')
         plt.figure(figsize= (6,4))
         sns.heatmap(con_mat, annot = True,fmt='d',cmap="YlGnBu")

         Training Accuracy : 0.9773063209784252
         Testing Accuracy : 0.9629961335983748
```

Fig. 6. Testing and training accuracies for logistic regression

For Naive Bayes Classifier, the training and testing accuracies were 95% and 97%. Figures 7 and 8 show the training of the model and the result respectively.

```
In [43]: # create mnb object
         mnb = MultinomialNB()

In [44]: mnb.fit(trainX,trainY)
Out[44]: MultinomialNB()

In [45]: mnb.score(testX,testY)
Out[45]: 0.9573312363019434

         multinomial naives bayes gave me 95% accuracy
```

Fig. 7. Training of Na ïve Bayes model

The training of a Naïve Bayes model involves preprocessing the data, feature engineering, estimating class and feature probabilities, building a predictive model based on these probabilities, and evaluating the model's performance on unseen data. This process helps in building a probabilistic classifier for classification tasks,

```
In [46]: Scores_ml['MultinomialNB'] = np.round(mnb.score(testX,testY),2)

In [47]: print('Training Accuracy :',mnb.score(trainX,trainY))
         print('Testing Accuracy :',mnb.score(testX,testY))
         con_mat = pd.DataFrame(confusion_matrix(mnb.predict(testX), testY),
                     columns = ['Predicted:Bad', 'Predicted:Good'],
                     index = ['Actual:Bad', 'Actual:Good'])


         print('\nCLASSIFICATION REPORT\n')
         print(classification_report(mnb.predict(testX), testY,
                                     target_names =['Bad','Good']))

         print('\nCONFUSION MATRIX')
         plt.figure(figsize= (6,4))
         sns.heatmap(con_mat, annot = True,fmt='d',cmap="YlGnBu")

         Training Accuracy : 0.97405639197202
         Testing Accuracy : 0.9573312363019434
```

Fig. 8. Testing and training accuracies for Na ïve Bayes

The accuracy of the individual models on the training and test datasets is as shown in figure 9. The F1-score, Precision and Recall of Logistic regression is as shown in figure 10.

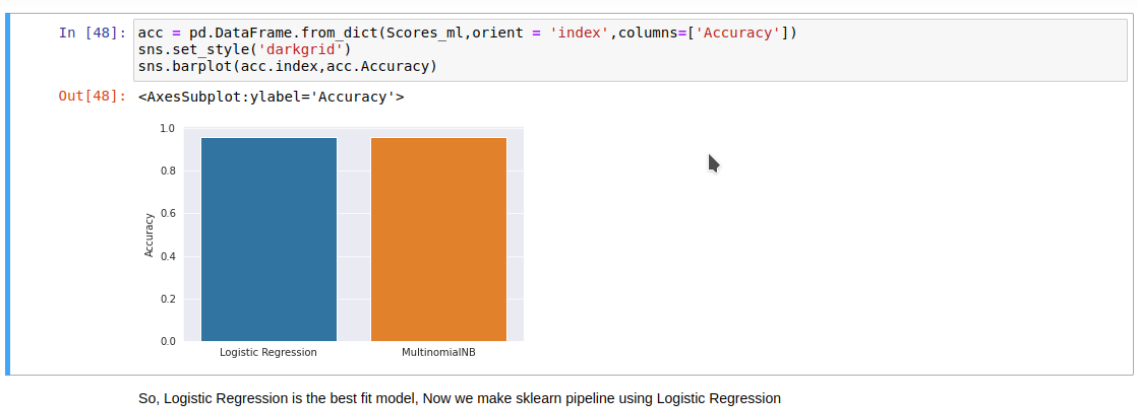


Fig. 9. Accuracies of the models

The Random Forest model outperformed the Logistic Regression model in terms of accuracy, with a training accuracy of 99.99% and a validation accuracy of 98%. On the other hand, the Logistic Regression model achieved a validation accuracy of 94.04%. These accuracies indicate the performance of the models in detecting malware based on the features extracted from the CIC-MalMem-2022 dataset. The higher accuracy of the Random Forest model suggests its effectiveness in classifying malware instances compared to the Logistic Regression model.

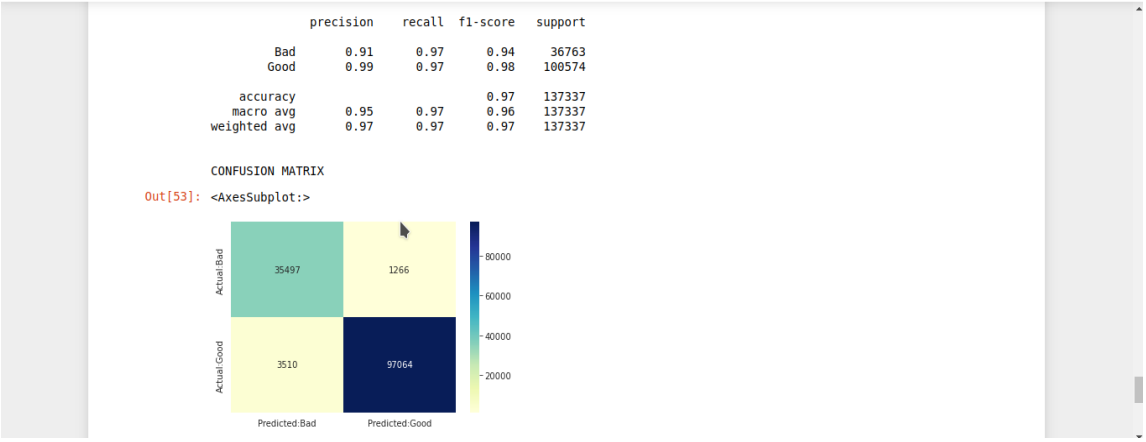


Fig. 10. F1-score, Precision and Recall of logistic regression

Logistic Regression has a higher accuracy as compared to Naïve Bayes. A pipeline was created for logistic regression model as shown in figure 11, it significantly improved the readability and comprehension of the process. It ensured the implementation and sequencing of the different phases in this study.

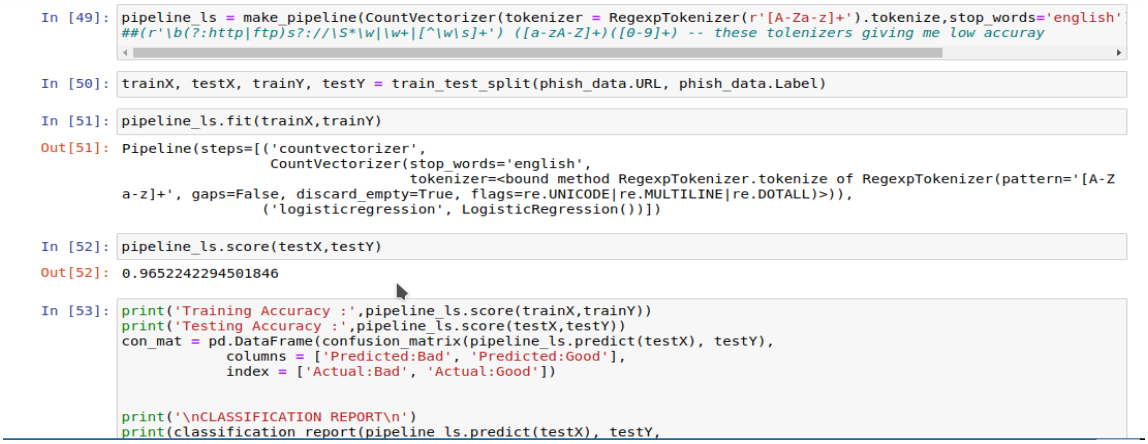


Fig. 11. Scikit-Learn Pipelines

After building the model, it was dumped using (pickle). Dumping is the process of converting a Python object into a byte stream so that it can be saved to a file or database, kept in memory between sessions, or sent over a network. Then the model was loaded again to check the accuracy which was 96.5% as seen in figure 12. The dumped model in the operating system used is as shown in figure 13

```
In [54]: pickle.dump(pipeline_ls,open('phishing.pkl','wb'))

In [55]: loaded_model = pickle.load(open('phishing.pkl', 'rb'))
         result = loaded_model.score(testX,testY)
         print(result)

         0.9652242294501846
```

Fig. 12. Dumping model using Pickle

Dumping a model using Pickle in Python is the process of serializing (converting) a machine learning model object into a byte stream that can be saved to a file. This serialized object can then be stored persistently on disk for later use, sharing with others, or deployment in production systems. Pickle is a built-in Python module that provides a convenient way to serialize and deserialize Python objects.

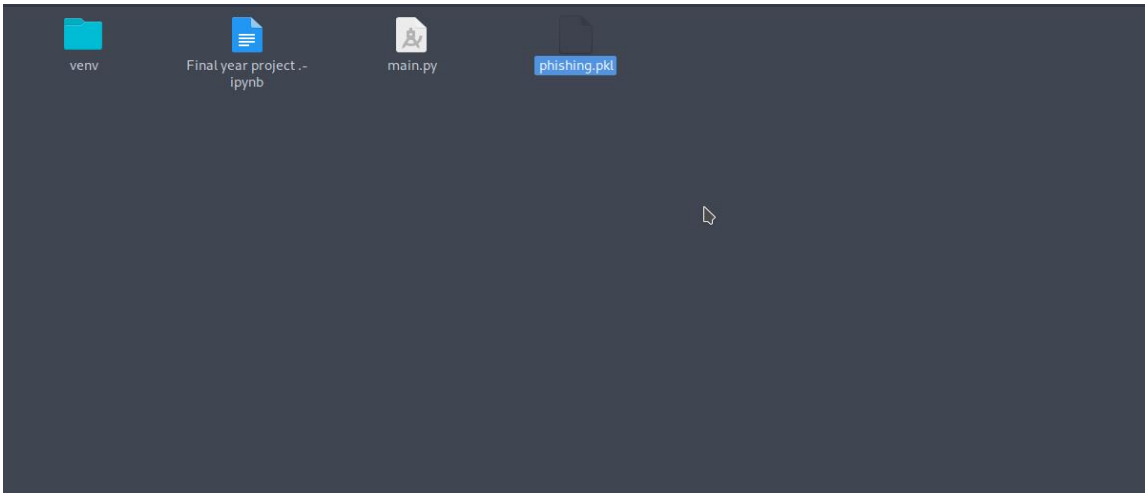


Fig. 13. The dumped model

To aid a good interaction between users and the system, a graphical interface was created. The Graphical interface was created using uvicorn and the pkl file was imported as shown in figure 14.

```
main.py
from fastapi import FastAPI
import joblib,os
import socket

app = FastAPI()

#pkl
phish_model = open('phishing.pkl','rb')
phish_model_ls = joblib.load(phish_model)


# ML Aspect
@app.get('/predict/{feature}')
async def predict(features):
    X_predict = []
    X_predict.append(str(features))
    y_Predict = phish_model_ls.predict(X_predict)
    if y_Predict == 'bad':
        result = "This is a Phishing Site"
    else:
        result = "This is not a Phishing Site"


    return (features, result)
if __name__ == '__main__':
    uvicorn.run(app,host="127.0.0.1",port=8000)
```

Fig. 14. Code of the graphical user interface

The model was tested with a non-phishing website. The model predicted well that “www.google.com” is not a phishing website as shown in figure 15. A phishing website as shown in figure 16 was also used to test our model. The result showed that the website is a phishing site as seen in figure 17.

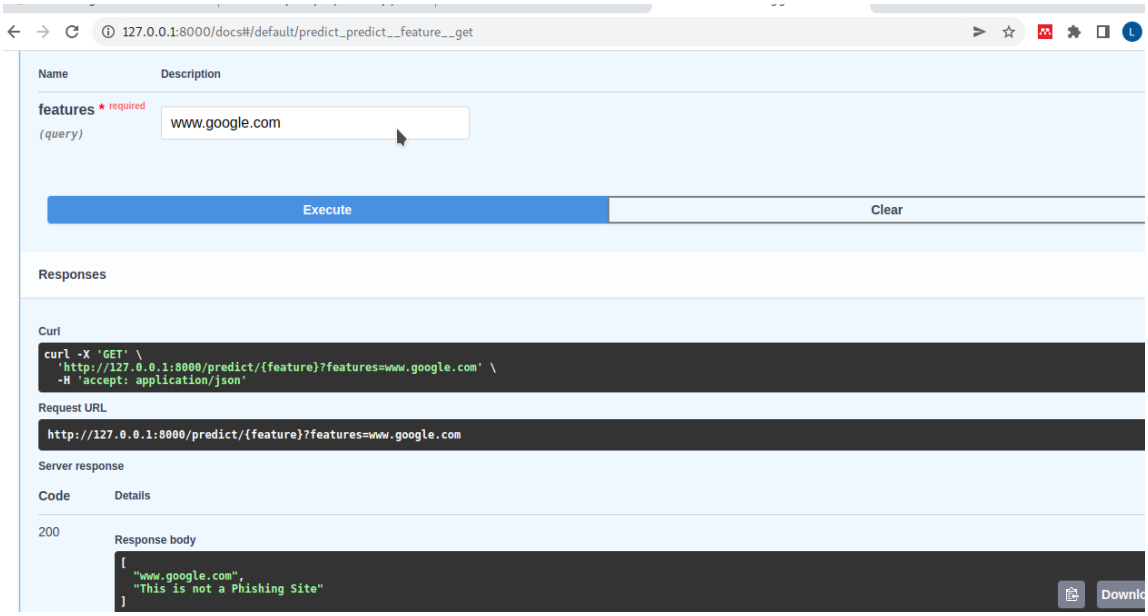


Fig. 15. Non-phishing web prediction

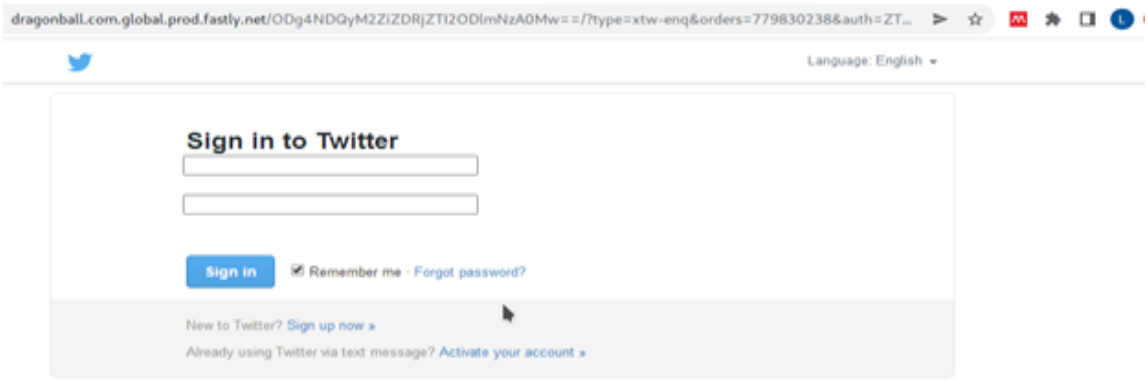


Fig. 16. Twitter phishing site

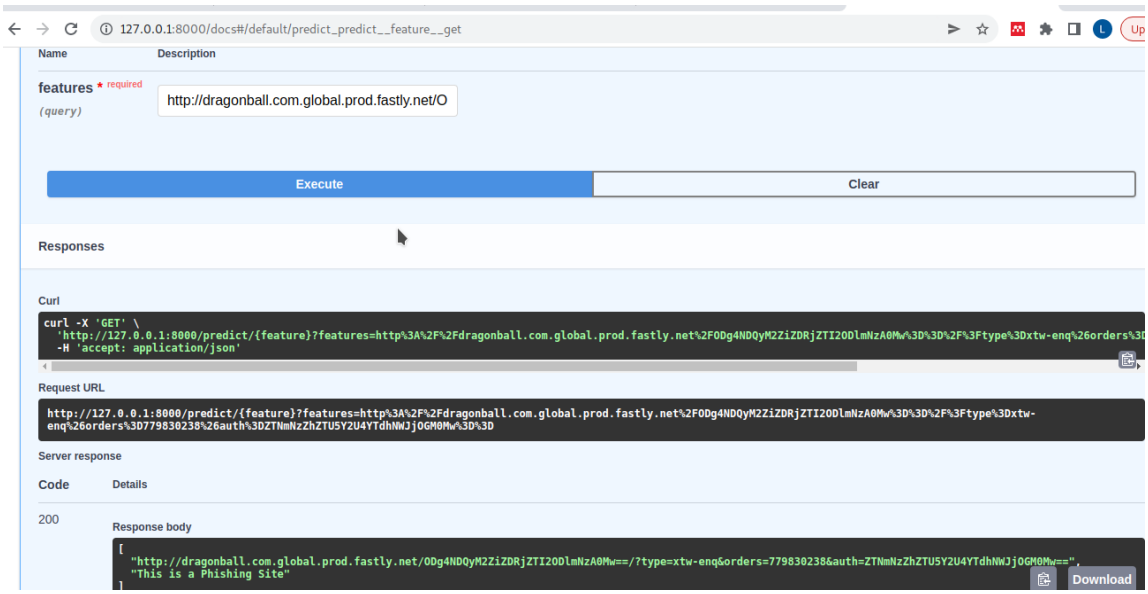


Fig. 17. Phishing site prediction

5. Conclusion

The machine-learning algorithms utilized in this study can be used to determine the legitimacy of a website or not. It can be concluded that logistic regression did better with high accuracy by including 16 features. The training accuracy was 97% and the testing accuracy was 96%. This study will help prevent the user from accessing deceptive sites. We recommend that more work be done to block the user from accessing the phishing site by auto-detecting it instead of the URL being input by the user to the model interface before it can be verified.

The work adopted Pearson correlation coefficient for feature selection. However, this method may overlook non-linear relationships between features. Utilizing more advanced feature selection techniques like recursive feature elimination or Lasso regularization could improve model performance. Also the paper does not explicitly address the issue of imbalanced data, which can lead to biased models. Implementing techniques such as oversampling, under sampling, or using algorithms like SMOTE to handle imbalanced classes could enhance model robustness.

While the paper highlighted evaluating the models using accuracy, incorporating additional metrics like precision, recall, F1-score, and ROC-AUC can provide a more comprehensive assessment of model performance, especially in the context of malware detection. However, these limitations can be improving by combining multiple models such as Random Forest, Gradient Boosting, or Neural Networks through ensemble techniques like stacking or blending could potentially enhance the overall predictive power of the system. Exploring advanced feature engineering techniques like polynomial features, interaction terms, or domain-specific feature creation could help extract more relevant information from the dataset and improve model performance.

Also, incorporating techniques for model explain ability, such as SHAP values or LIME, could provide insights into how the models make predictions, especially in critical applications like malware detection where interpretability is essential, Finally, implementing robust cross-validation strategies like k-fold cross-validation or stratified cross-validation can provide a more reliable estimate of model performance and generalization to unseen data.

Suggested ways for further research could be extended by xploring of different algorithms by:

a. investigating the use of deep learning architectures such as Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) for malware detection. These models can capture complex patterns in malware behavior and improve detection accuracy.
b. Explore ensemble methods like AdaBoost, XGBoost, or stacking multiple models to leverage the strengths of different algorithms and enhance overall performance.
c. Consider incorporating anomaly detection algorithms like Isolation Forest or One-Class SVM to identify novel and previously unseen malware patterns.

Also Utilization of Larger Datasets by

a. Acquire and utilize larger and more diverse malware datasets from real-world sources to improve the model's ability to generalize to different types of malware.
b. Explore techniques for handling streaming data in real-time to adapt the model to evolving malware threats and ensure continuous monitoring and detection.

Addressing Real-Time Implementation Challenges by:

a. Address scalability challenges to deploy the model in real-time systems, ensuring efficient processing of large volumes of data without compromising performance.
b. Optimize the model for low latency to enable real-time detection of malware threats, especially in time-sensitive environments like network security.
c. Develop lightweight models that can operate efficiently on resource-constrained devices or systems, such as IoT devices or edge computing platforms.
d. Incorporate techniques for explaining model decisions, such as SHAP values, LIME, or model-specific interpretability methods, to enhance trust and understanding of the detection system.
e. Investigate methods to enhance the model's robustness against adversarial attacks specifically designed to evade malware detection systems.
f. Implement techniques for incremental learning to adapt the model to new malware variants and continuously improve detection accuracy over time.

References

[1] Gururaj Harinahalli Lokesh & Goutham BoreGowda (2021). Phishing website detection based on effective machine learning approach. *Journal of Cyber Security Technology.* Vol 5, issue 1, pp 1-14, DOI: 10.1080/23742917.2020.1813396

[2] Kiruthiga, R., & Akila, D. (2019). Phishing websites detection using machine learning. *International Journal of Recent Technology and Engineering*, *8*(2 Special Issue 11), 111–114. https://doi.org/10.35940/ijrte.B1018.0982S1119

[3] Dutta, A. K. (2021). Detecting phishing websites using machine learning technique. *PLoS ONE*, *16*(10 October), 1–17. https://doi.org/10.1371/journal.pone.0258361

[4] Alnemari Shouq, and Majid Alshammari (2023). Detecting Phishing Domains Using Machine Learning. *Applied Sciences*. Vol 13, No 8. https://doi.org/10.3390/app13084649

[5] Pankaj Pandey & Nishchol Mishra (2023). Phish-Sight: a new approach for phishing detection using dominant colors on web pages and machine learning. *International Journal of Information Security.* https://doi.org/10.1007/s10207-023-00672-4

[6] Mohammed Hazim Alkawaz, Stephanie Joanne Steven, Omar Farook Mohammad, Md Gapar Md Johar (2022). Identification and Analysis of Phishing Website based on Machine Learning Methods. *2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. 10.1109/ISCAIE54458.2022.9794467

[7] Sumitra Das Guptta, Khandaker Tayef Shahriar, Hamed Alqahtani, Dheyaaldin Alsalman & Iqbal H. Sarker (2022). Modeling Hybrid Feature-Based Phishing Websites Detection Using Machine Learning Techniques. Annals of Data Science. https://doi.org/10.1007/s40745-022-00379-8

[8] Gandotra, E., Gupta, D. (2021). An Efficient Approach for Phishing Detection using Machine Learning. In: Giri, K.J., Parah, S.A., Bashir, R., Muhammad, K. (eds) Multimedia Security. Algorithms for Intelligent Systems. Springer, Singapore. https://doi.org/10.1007/978-981-15-8711-5_12

[9] A. Lakshmanarao, P.Surya Prabhakara Rao & M M Bala Krishna (2021). 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS). 10.1109/ICAIS50930.2021.9395810

[10] Zamir, A., Khan, H.U., Iqbal, T., Yousaf, N., Aslam, F., Anjum, A. and Hamdani, M. (2020). Phishing web site detection using diverse machine learning algorithms. *The Electronic Library,* vol. 38, No. 1, pp. 65-80. https://doi.org/10.1108/EL-05-2019-0118

[11] Zamani, H., & Mohammed Amin, M. K. (2016). Classification of phishing websites using machine learning techniques. *Journal of Advanced Research in Applied Sciences and Engineering Technology Journal Homepage*, *5*(2), 12–19. www.akademiabaru.com/araset.html

Authors' Profiles

Olugbenga A. Madamidola received the B.Sc. from Bowen University, Iwo, Osun state Nigeria, 2012, M.Tech, from the Federal University of Technology Akure, Nigeria, 2017 and currently a Doctoral Candidate at The Federal University of Technology Akure, Nigeria,. He currently holds a teaching position at the Information Technology Department, School of Computing, Federal University of Technology Akure. His research interests include, Computer Communication, Information Security, Intelligent Sytems and Computer Vision. He is a member of Internet Society, ACM, Nigeria Computer Society and IAENG.

Ilobekemen P. Oladoja received the B.Sc. from Achievers University Owo, M.Tech. and PhD degrees from the Federal University of Technology Akure, Nigeria, 2014, 2019 and 2020 respectively. She is currently lecturing at the Computer Science Department, School of Computing, Federal University of Technology Akure. Her research interests include, Cloud Computing, Soft and High-performance Computing, and Machine Learning.

Peace B. Falola holds a teaching position at the Department of Computer Science, University of Ibadan, Ibadan, Nigeria. She is also a PhD student at the University of Ibadan, Ibadan, Nigeria. Her research areas span Artificial Intelligence, Named Entity Recognition, Information Security, Machine Learning, Deep Learning, and Human-Computer Interaction. She has quite several publications in reputable journals. Her current research focus is developing Named Entity Recognition Models for African Languages. She is a member of The Organization for Women in Science for the Developing World (OWSD).

Matthew W. Omojola received the B.Sc. from Precious Cornerstone University, Ibadan, Oyo State Nigeria. He is a seasoned cybersecurity expert with five years of experience in protecting data and computer systems. Specializing in penetration testing, cloud security, and DevSecOps, he provides comprehensive security solutions through meticulous analysis, detailed planning, and thorough preparation.

How to cite this paper: Olugbenga A. Madamidola, Ilobekemen. P. Oladoja, Peace B. Falola, Matthew W. Omojola, "Development of a Phishing Website Detection Model Using Classification Algorithm", International Journal of Wireless and Microwave Technologies(IJWMT), Vol.14, No.5, pp. 31-43, 2024. DOI:10.5815/ijwmt.2024.05.03