Web Hazard Identification and Detection Using Deep Learning - A Comparative Study

S. Sivanantham[1*], V. Krishnamoorthy[2], D. Karthikeyan[3], M. Sakthivel[4], V. Mohanraj[5], V. Akshaya[6]

[1*]AP/CSSE, Sree Vidyanikethan Engineering College, Tirupati, AP, India. E-mail: sivanantham.s@vidyanikethan.edu

[2]AP/CSE, Bannari Amman Institute of Technology, Sathyamangalam, TN, India. E-mail: krishnamoorthy.v@bitsathy.com

[3]AP (Sr. Grade 1), School of Information Technology, VIT University, Vellore, TN, India. E-mail: karthikeyan.duraisamy@vit.ac.in

[4]Prof/CSE, Sree Vidyanikethan Engineering College, Tirupati, AP, India. E-mail: sakthivel.m@vidyanikethan.edu

[5]Prof/IT, Sona College of Technology, Salem, TN, India. E-mail: vmohanraj06@gmail.com

[6]AP/CSE, Sree Vidyanikethan Engineering College, Tirupati, AP, India. E-mail: akshaya.v@vidyanikethan.edu

Abstract

Surfing the internet has become an integral part of our day-to-day life. This has become the potential source of intruder attacks. Hazard is cybercriminal posed threat, the simple example for the same is creating malicious URL to pose phishing attack and to gain access to user's personal information. The consequences include identity theft, other types of frauds like malware injection onto the computing devices. Malicious URL is a link that redirects the user to a fraudulent web page. Recognition of such malicious URLs is a prolonging problem since machine learning (ML) has evolved. There have been ML classifier like random forest (RF) and deep learning (DL) classifiers such as Convolutional Neural Network (CNN), Back Propagation Neural Networks (BPNN) and Long Short-Term Memory (LSTM) which may address this classification problem of segregating URLs into malicious and normal. Still these techniques are not sufficient to protect the internet users and requires a robust model that will distinguish between the normal and malicious web pages. This paper introduces a comparative study about ML and DL techniques in classification of URLs as malicious and normal in the given dataset. Among the implemented techniques BPNN gave an optimal accuracy of 96.86%

Keywords: Network Attack, Malicious URL, Back Propagation Neural Network, Convolutional Neural Network, Random Forest, Principal Component Analysis, Long Short Term Memory.

DOI: 10.47750/pnr.2022.13.04.145

INTRODUCTION

As the programs are thought of distinctive progressed highlights and functionalities which prompts hazard by losing their own and delicate data. With the quick advancement of the web, more administrations like e-learning and so forth are accessible to clients and they are surfing the internet by means of web application. As clients are not mindful of vindictive sites, it created an opportunity for intruders to infuse the payloads to get distant admittance to casualty's web page. Consequently, the exact identificant particle of web pages in an always developing web climate is very significant. Boycotting administrations were implanted in the programs to confront the difficulties, yet it has a few inconveniences like inaccurate listing.

Phishing is a attack that is posed when a malicious source pretends like the real one and it leads to an extensive fraud. The purpose of this type of attack is to gain access to sensitive data like passwords, personal data, credit card numbers and so on.

There exist a fact that there are a few contrary programming techniques to phishing and they differ in the potential phishing endeavours and finding phishy substance on sites. The new age phishers think of novel and half breed strategies to pool around the non-accessible programs and systems.

Phishing is a tricky method that uses a combination of social design and innovation to collect sensitive individual data, by leaving an impression of a reliable individual or concern in an electronic correspondence. Phishing techniques utilizes the spoof messages that looks valid and getting originated from some reliable sources like funding agencies, ecommerce websites and so forth. They draw users to visit fraudulent sites through the URLs given in the phishing e-mail. These misleading websites emulate the genuine website.

An example is engaging a phishing trap to activate the suspended client's accounts or asking to finish their account upgrade process. The customer may give some touchy information for these reasons or may land in satirize web page.

Supervised learning (particularly classification technique) provides an optimal precision when compared to unsupervised learning yet it provides a swift and reliable way to infer vital information from a dataset. We have used supervised learning techniques in this proposed research work.

With the quick advancement of the web, more administrations like e-learning and so forth are accessible to

clients and they are surfing the internet by means of web application. The reasons and significance of hazard identification can be summed up as

- As clients are not mindful of vindictive sites.
- Significance of exact identificant particle of web pages in an always developing web climate.
- Boycotting administrations were in the programs to confront the difficulties.
- Using AI classifiers group the site into two classes generous and malignant pages.

RELATED WORK

The related literature work to the proposed research shall be discussed in this section,

Han. W et al. introduced an Automated Individual White-List preparation system (AIWLS) recorded the details of familiar benign sites visited by users. The records of AIWLS is compared with user entered URLs and thus prevents the unnecessary disclosure of sensitive information to malicious sites by users. Their technique provided an efficient defence against dynamic phishing attacks. The method reserves a dependency on pre-processing of the AIWLS records i.e. dependent on user input URLs feedbacks [1].

Blum. A et al. proposed Phishing URL Detection based on Lexical Feature. Bigrams are used as indicators for characterizing URLs based on the results obtained from the author's batch learning algorithms. Batch learning algorithms helps in extracting some relevant features, while others are discarded which do not contribute value for further processing. The major drawback of feature spaces in bigrams represented by n-gram models is extreme sparcity. Time and resources spent in retraining the model is considered as a disadvantage of batch learning [2].

Ma, J et al. has presented a work named "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs". This current work is based on their existing Ma et al.'s online learning algorithm, hybrid approach that involves clustering cum classification and URL ranking mechanism. Their work uses a similar set of lexical features. The difference here is the Blum et al.'s [2] technique ignores using host based features. Use of online learning algorithm based classifier helps us achieve an accuracy of 97% if quality training data is provided. Missing values handling is tough and nonlinear parameters (being not suitable for both categorical and continuous data) affect the performance [3].

Sanjukta. Mohanty et. al. worked out the scheme named Hazard Identification and Detection using Machine Learning Approach. A novel lightweight self-learning technique is introduced to classify the malicious webpage based on the features. A MAL URL framework was introduced where the random forest (RF) machine learning algorithm is a part and is used to train classifiers and thus making the system capable of malicious web URLs detection [4]. Compared to the above discussed works related to hazard detection, this current work reduces over fitting, handles non-linear parameters, less influenced by noise.

Limitations of this model are,

1. **Complexity**: Random Forest (RF) is a combination multiple trees and their outputs are consolidated. By default, the number of trees is 100 when implemented using Python sklearn library. The RF algorithm requires huge computational resources. If simplicity is a concern, decision tree is the best choice since it does not require much computational resources.
2. **Longer Training Period:** RF technique demand much time to train when compared to decision tree method as it generates a lot of trees and decision is taken based on the majority voting.

BACKGROUND

This section deals with the state of art techniques used in classification task.

A. Random Forest

Both classification and regression problems can be solved by Random forest algorithm. Random Forest belongs to supervised learning category in machine learning. RF algorithm is basically an ensemble learning technique where multiple classifier's results are combined to find solution for a complex problem and thus the performance of the model improved [5].

As the name says, random forest algorithm works with multiple decision trees it operates through bagging or bootstrap aggregating. To say about Bagging, it is an ensemble meta-algorithm that provides accurate classification results among the machine learning algorithms available.

The working principle of Random Forest is that it creates a number of decision trees on various subsets of the provided dataset and considers the average to improvise the prediction accuracy in the dataset.

The prediction accuracy seems to be fair since the technique is not going to rely on outcome of one decision tree, it takes the prediction from every decision tree and conducts a majority voting of the predictions made, then it decides the final outcome [6].

The below Fig.3.1 depicts the functionality of the Random Forest algorithm.

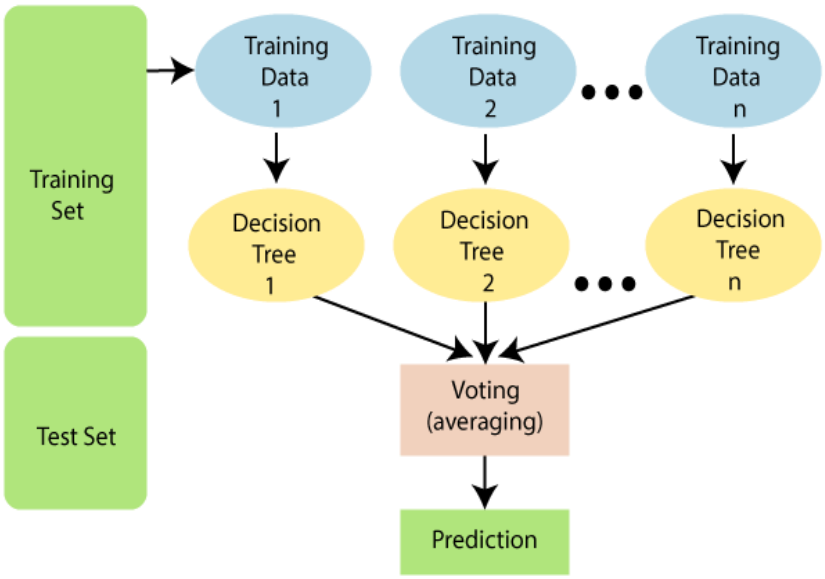

Figure 3.1: Principle of Random Forest algorithm

B. Convolutional Neural Network (CNN)

Convolutional Neural Network is the popular deep learning technique. The main application of CNN is image classification and recognition. Some other fields where CNN is applied are scene labelling, objects detections, and face recognition, etc., [7].

In image recognition, CNN considers an image as input, which is segregated and process them as certain category. The computer deals an image as an array of pixels and number of pixel depends on the resolution of the image. The image will be seen as h X w X d, where h is the height, w is the width and d is the dimension of the image. For example, An RGB image will be of 8 X 6 X 4 sized multi-dimensional array of the matrix, and the grayscale image is 5 X 4 X 1 sized multi-dimensional array matrix.

CNN is a layered structure in which each input image will pass through various things like sequence of convolution layers, pools, fully connected layers and filters. The Soft-max function is used to classify an object with probabilistic values 0 & 1 as shown in the Fig 4.1.

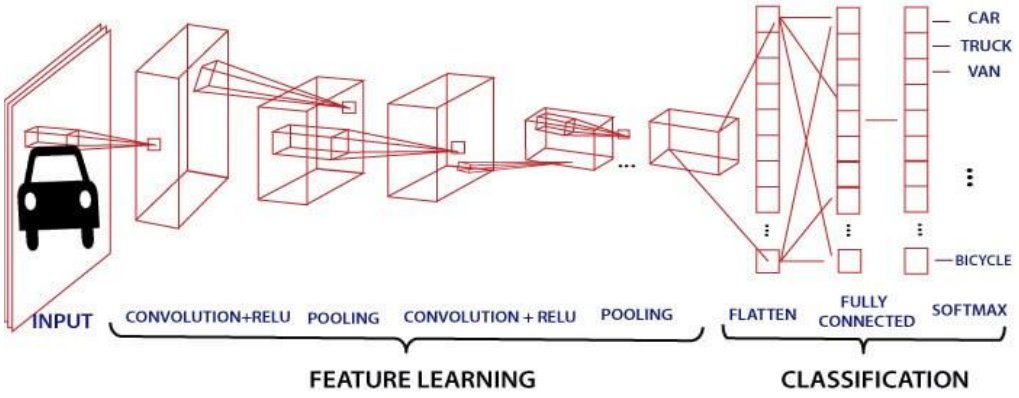

Figure 4.1. CNN Process

C. Convolution Layer

As shown in Fig. 4.2, the convolution layer is the very first layer to extract features from an input image. The convolutional layer maintains the link between pixels by learning visual properties using a tiny square of input data. Using an image matrix and a kernel or filter as two inputs, it performs a mathematical action.

The dimension of the input image is H X W X D.

The dimension of the filter used is FH X FW X D.

The dimension of the output is given as (H-FH+1) X (W-FW+1) X 1.

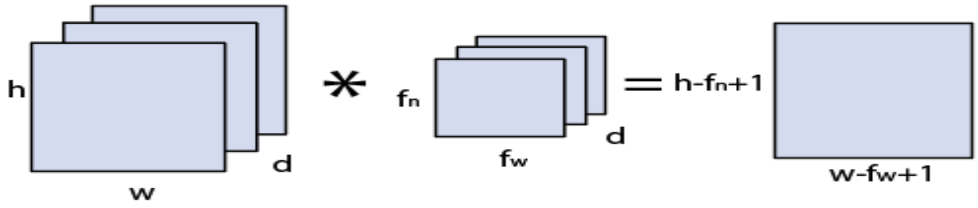

Figure 4.2. Image matrix multiplies kernel or filter matrix.

THE PLANS PROPOSED

The proposed research work includes the implementation of random forest and convolutional neural network and the classification results are compared in terms of accuracy.

The generic steps in the implementation of any classification technique like importing of data, pre-processing and feature selection are taken up. The proposed work is Jupyter Notebook using python.

A. Importing of Data

The data import is done by importing pandas, here we are using different packages to load and read the datasets. By using pandas, we can read the.csv file and then we can display the shape of the dataset with that we can also display the dataset in the correct form. We will be training and testing the data, when supervised learning is used it means data is labelled. By getting the testing, training data and their labels we can evaluate different machine learning algorithms but before performing the predictions and accuracies, the data is need to be pre-processing i.e. the null values which are not readable are required to be removed from the data set and the data is required to be converted into vectors by normalizing and tokening the data so that it could be understood by the machine. Next step is by using this data, getting the visual reports, which we will get by using the Mat Plot Library of Python and Scikit Learn. This library helps us in getting the results in the form of histograms, pie charts or bar charts.

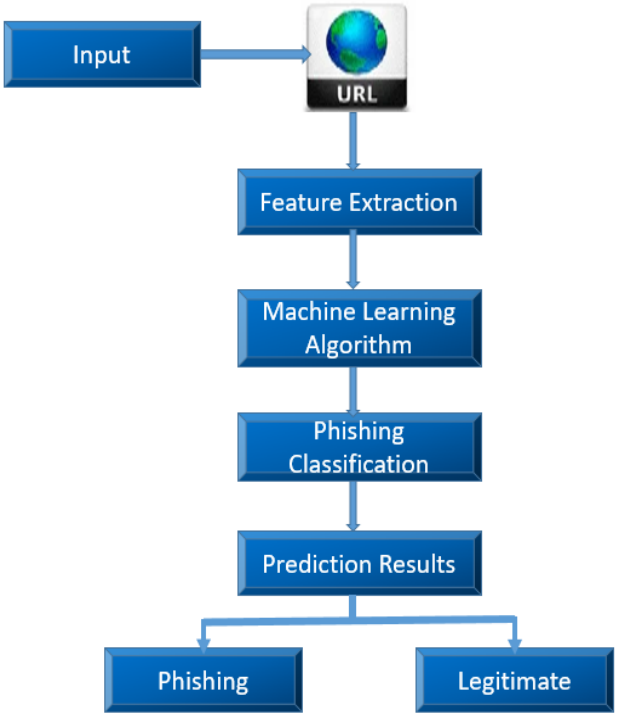

Figure 4.3. The Proposed Model

The overall architecture of the proposed system is depicted in Fig 4.3.

B. Preprocessing

The data set used here contains 420464 instances with 18% (75643) bad and 82% (344821) good labeled urls. Dataset is divided into a training set and a testing set containing in Dataset 336371 training data and 84093 testing i.e is 8:2 ratio respectively. Cleaning the data is always the first step. In this, those nulls are removed from the dataset. That helps in mining the useful information. Whenever we collect data online, it sometimes contains the undesirable characters like not null, noisy data which creates hindrance while phishing urls[6] detection. Preprocessing helps in removing the labels which are independent entities and integrate the logic which can improve the accuracy of the identification of the item of interest [8]. Multivariate feature Imputation technique is used for dealing the null and missing values in the dataset [9].

C. Feature Extraction

The process of choosing a subset of pertinent features to be used in model creation is known as feature extraction. Methods for feature extraction aid in the development of precise predictive models. They assist in deciding which attributes will provide more accuracy. The input data will be turned into a reduced and illustrated set of features known as feature vectors, when the input data to an algorithm is too vast to accommodate and is intended to be redundant. Employing the smaller representation rather than the full-size input and changing the input data to carry out the intended task. Prior to using any technique, feature extraction is carried out on the converted data in feature space on the raw data. Principle Component Analysis (PCA) is used for feature selection in the proposed model [10], [11].

D. Training the Classifier

Scikit-Learn Machine leanring library is used for implementing the architecture. Scikit Learn is an open source python Machine Learning library which comes bundled in 3rd distribution anaconda. This just needs importing the packages and one can compile the command as soon as you write it. Three different machine learning and deep learning algorithms i.e. RF, CNN, Long Short Term Memory (LSTM) and Back propagation neural networks are used as models in training. We can predict the label of each url with the trained models. Once the classifiers are trained, we checked the performance of the models on test-set and the performance is compared.

RESULTS

The implementation results for this paper is as follows: URLs are classified in to three categories namely domain, subdomain and domain suffix as shown in Fig 5.1.

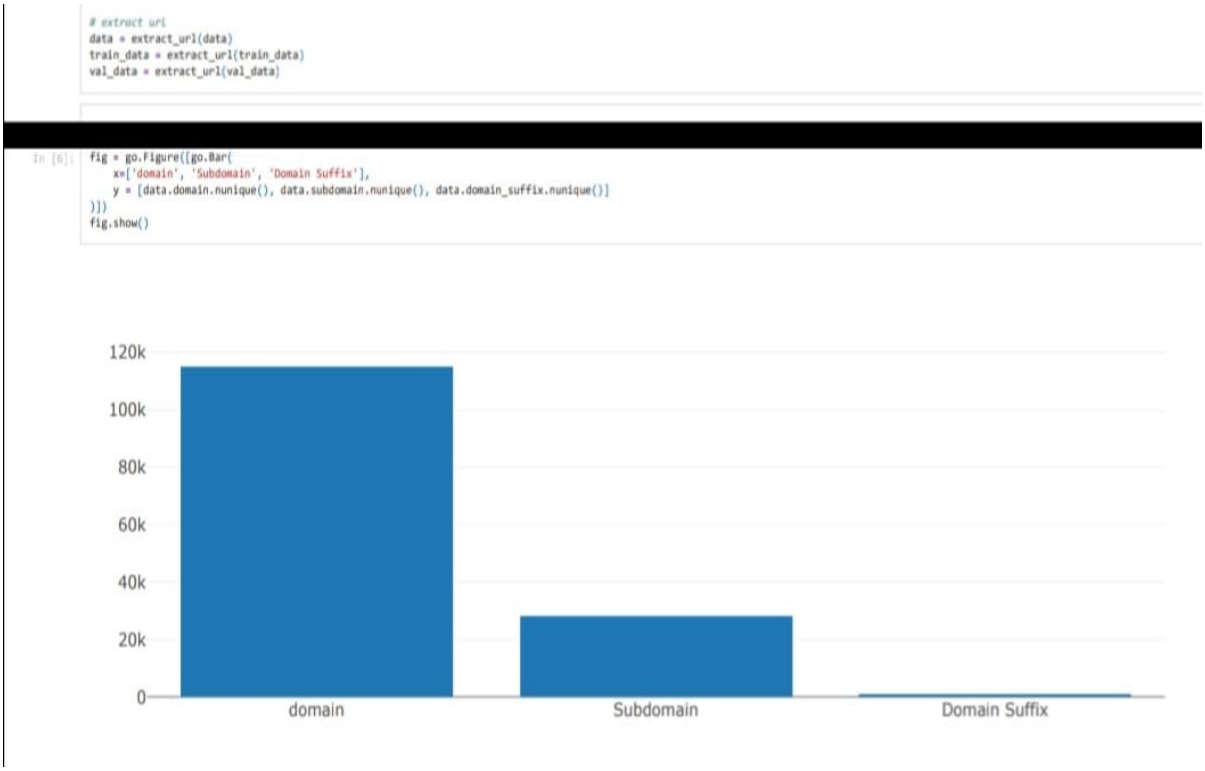

Figure 5.1: Domain classification

Text is divided into a set of meaningful parts through the process of tokenization. These pieces are called tokens. This is the initial step to be done before the classification can actually be done. For instance, we could break up a passage of text into words or phrases. We can create our own conditions to split the input text into relevant tokens based on the task at hand. Tokenization can be done by three techniques [12]

Padding: “pre” or “post” padding (by default pre). Pre and post allow us to pad (add 0) before and after each sequence, respectively.

Maxlen: defines the maximum length of all sequences. If not provided, by default it will use the maximum length of the longest sentence [13].

Truncating: ‘pre or post (pre is the default). Sequence lengths that are longer than the specified maxlen value will be shortened to that value. The "pre" option will truncate the sequences at the start, whereas the "post" option will truncate them at the end [14] [15].

```
[11] tokenizer = Tokenizer(filters='', char_level=True, lower=False, oov_token=1)

     # fit only on training data
     tokenizer.fit_on_texts(train_data['url'])
     n_char = len(tokenizer.word_index.keys())

     train_seq = tokenizer.texts_to_sequences(train_data['url'])
     val_seq = tokenizer.texts_to_sequences(val_data['url'])

     print('Before tokenization: ')
     print(train_data.iloc[0]['url'])
     print('\nAfter tokenization: ')
     print(train_seq[0])

     Before tokenization:
     mister-ed.com/welcome/file/update/rbc/login.php

     After tokenization:
     [12, 5, 9, 7, 2, 10, 15, 2, 16, 13, 8, 3, 12, 6, 26, 2, 14, 8, 3, 12, 2, 6, 25, 5, 14, 2, 6, 19, 17, 16, 4, 7, 2, 6, 10, 21, 8, 6, 14, 3, 20, 5, 11, 13, 17, 18, 17]

[12] sequence_length = np.array([len(i) for i in train_seq])
     sequence_length = np.percentile(sequence_length, 99).astype(int)
     print(f'Before padding: \n {train_seq[0]}')
     train_seq = pad_sequences(train_seq, padding='post', maxlen=sequence_length)
     val_seq = pad_sequences(val_seq, padding='post', maxlen=sequence_length)
     print(f'After padding: \n {train_seq[0]}')

     Before padding:
      [12, 5, 9, 7, 2, 10, 15, 2, 16, 13, 8, 3, 12, 6, 26, 2, 14, 8, 3, 12, 2, 6, 25, 5, 14, 2, 6, 19, 17, 16, 4, 7, 2, 6, 10, 21, 8, 6, 14, 3, 20, 5, 11, 13, 17, 18, 17]
     After padding:
      [12  5  9  7  2 10 15  2 16 13  8  3 12  6 26  2 14  8  3 12  2  6 25  5
      14  2  6 19 17 16  4  7  2  6 10 21  8  6 14  3 20  5 11 13 17 18 17  0
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
       0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0]
```

Figure 5.2 Tokenization and padding

The dataset can be visualized in Fig 5.3 and here it is clear that the top 20 domains are grouped by labels[16].

```
In [7]: plot(
            data=data.groupby('label')['domain'],
            n_group=20,
            title='Top 20 Domains Grouped By Labels (Logarithmic Scale)'
        )
```

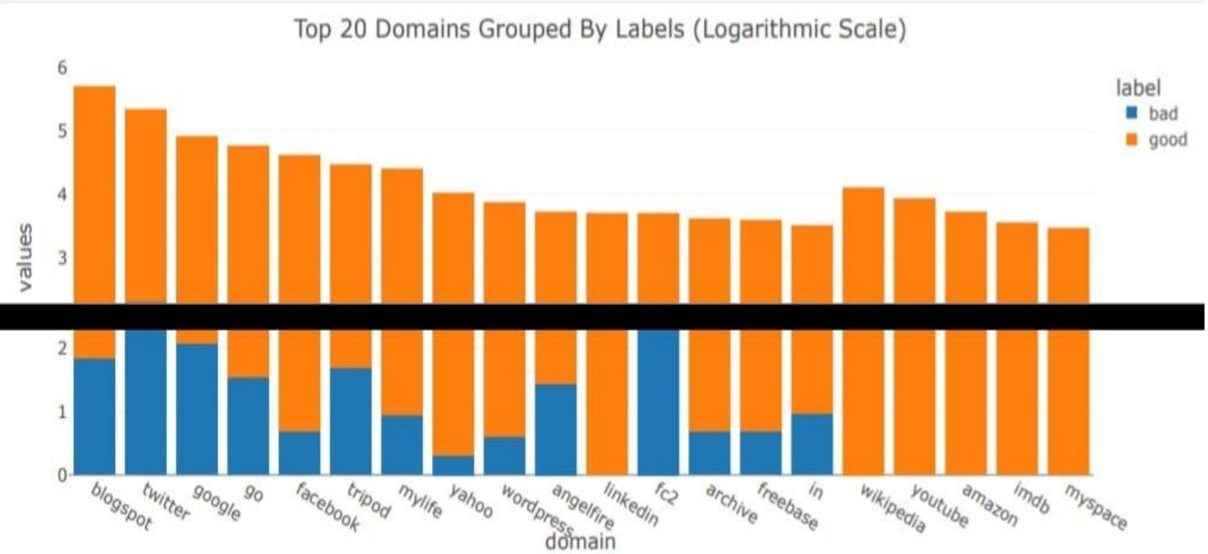
Top 20 Domains Grouped By Labels (Logarithmic Scale)
values
6
5
4
3
2
1
0
domain
blogspot
twitter
google
go
facebook
tripod
mylife
yahoo
wordpress
angelfire
linkedin
fc2
archive
freebase
in
wikipedia
youtube
amazon
imdb
myspace
label
bad
good

Figure 5.3 Top 20 domains grouped by labels

The Fig 5.4 shows the implementation of Random forest classifier. The average accuracy during training and testing is 80.62%

```
val_x = [val_seq, val_data['subdomain'], val_data['domain'], val_data['domain_suffix']]
val_y = val_data['label'].values

val_pred = model.predict(val_x)
val_pred = np.where(val_pred[:, 0] >= 0.5, 1, 0)
print(f'Validation Data:\n{val_data.label.value_counts()}')
print(f'\n\nConfusion Matrix:\n{confusion_matrix(val_y, val_pred)}')
print(f'\n\nClassification Report:\n{classification_report(val_y, val_pred)}')
```

```
Validation Data:
0    68964
1    15129
Name: label, dtype: int64


Confusion Matrix:
[[68408   556]
 [ 1569 13560]]


Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98     68964
           1       0.96      0.90      0.93     15129

    accuracy                           0.97     84093
   macro avg       0.97      0.94      0.96     84093
weighted avg       0.97      0.97      0.97     84093
```

Figure 5.4: Performance of Random Forest classifier

The Fig 5.5 shows the implementation of Convolutional Neural network and it classifies the URL's 84.09% accurately.

```
[18] # Import Random Forest classifier
     from sklearn.ensemble import RandomForestClassifier


     # instantiate the classifier

     rfc = RandomForestClassifier(random_state=0)


     # fit the model

     rfc.fit(X_train, y_train)


     # Predict the Test set results

     y_pred = rfc.predict(X_test)


     # Check accuracy score

     from sklearn.metrics import accuracy_score

     print('Model accuracy score with 10 decision-trees : {0:0.4f}'. format(accuracy_score(y_test, y_pred)))

     Model accuracy score with 10 decision-trees : 0.9416

[19] from sklearn.metrics import classification_report
     print(classification_report(y_test,y_pred))

                  precision    recall  f1-score   support

               0       0.97      0.96      0.96     68964
               1       0.83      0.85      0.84     15129

        accuracy                           0.94     84093
       macro avg       0.90      0.91      0.90     84093
    weighted avg       0.94      0.94      0.94     84093
```

Figure 5.5: Performance of Convolutional Neural Network

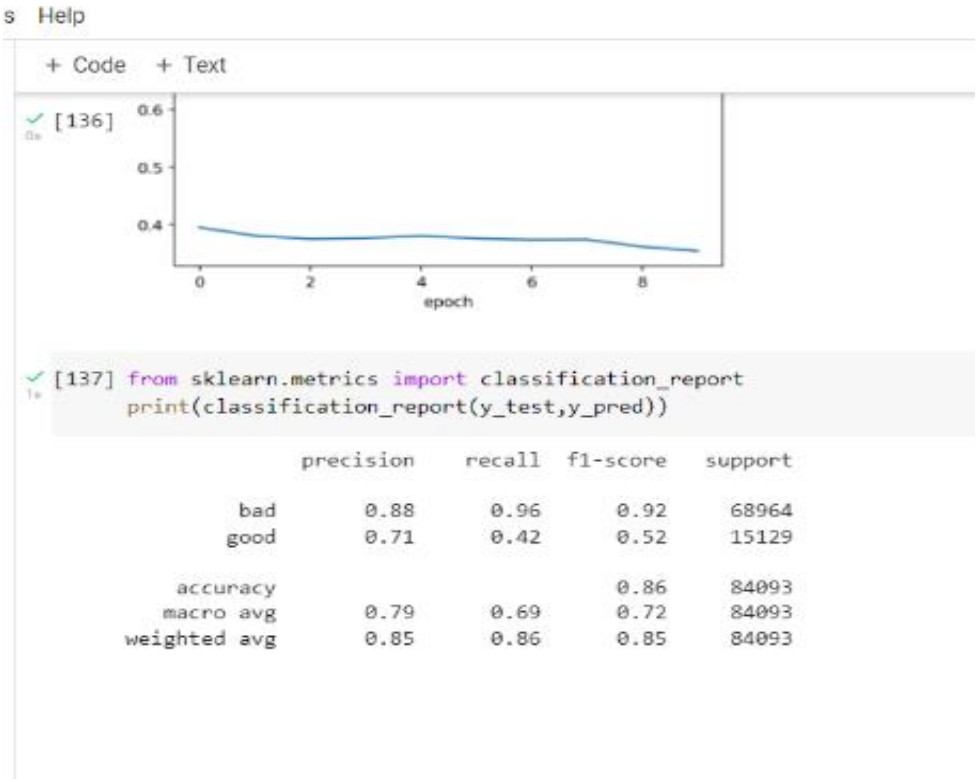

Figure 5.6: Performance of LSTM

The LSTM model is built on the dataset and the classification accuracy of the LSTM technique is measured as 84.1% as shown in Fig 5.6.

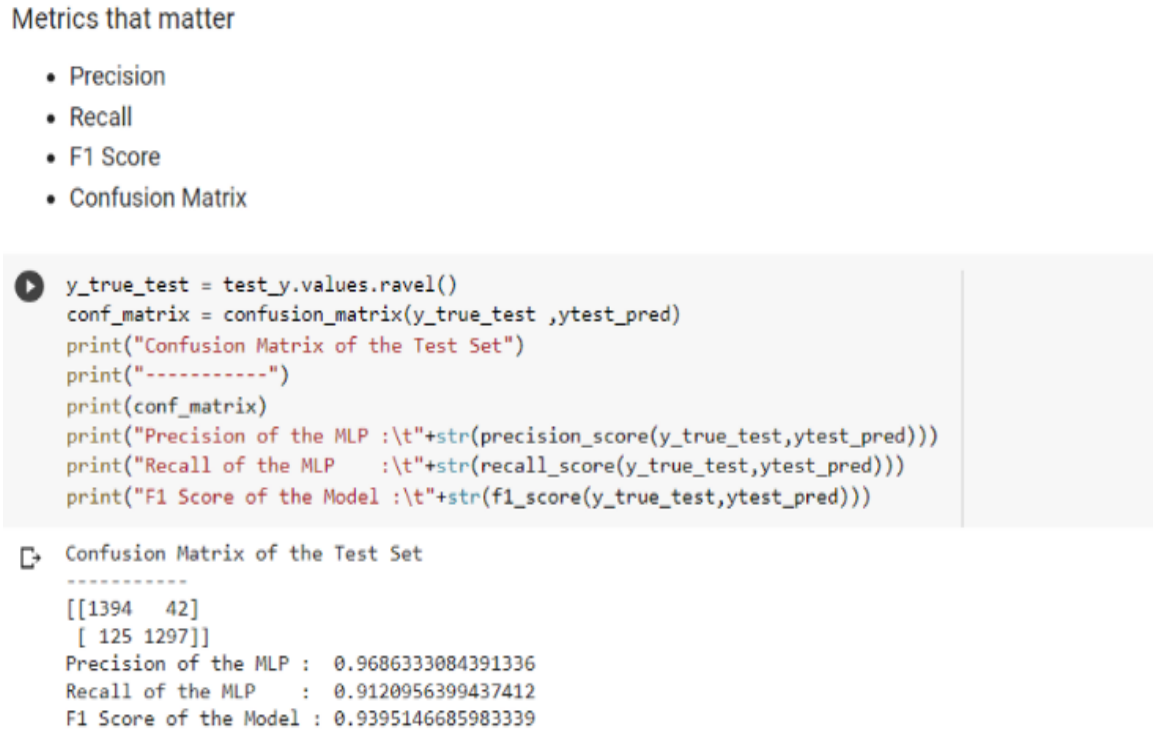

Figure 5.7: Back propagation NN

The back propagation NN [17] is also implemented and the accuracy achieved is 96.86% as shown in Fig 5.7. It is evident that the Back propagation neural network achieves the maximum accuracy compared to the other classifiers.

CONCLUSION

Phishing is a form of cybercrime that involves obtaining personal information about someone through social engineering and specialised deceit. Phishing is also regarded as a significant form of fraud. Recent reliable phishing data sets have been used in experiments using several classification algorithms that were trained using various learning techniques. Accuracy measurement serves as the experiments' foundation. This study aims to determine whether a given URL is a phishing website or not. Back propagation Neural Network-based classifiers find out to be the top classifier in the experiment, with a classification accuracy of 96.86% for the dataset of phishing sites. We may use this model in the future to analyse larger-scale phishing datasets and examine the efficacy of various classification algorithms in terms of classification accuracy. This proposed system will be helpful to specific applications in the future, such as anti-virus, firewall, data protection and security Software.

REFERENCES

Han W, Cao Y, Bertino E and Yong. Using of automated individual white-list to protect web digital identities. Expert Systems with Applications. ACM.org. 2012; 88: doi.10.1016, feb. 2012.

Blum A, Wardman B, Solorio T, and Warner G. Lexical Feature Based Phishing URL Detection Using Online Learning. AI Sec: 10 Proceedings of the 3rd ACM workshop on Artificial Intelligence and Security, Illinois. ACM New York, NY, USA. 2010; 54-60.

Ma J, Saul L, Savage S, and Voelker G. Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs. KDD'09 Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, ACM New York, NY, USA. 2009; 1245-1254. DOI=http://dl.acm.org/citation.cfm?id=1557019.1557153&coll=DL,feb.2 2009.

S. Mohanty, A. A. Acharya, L. Sahu and S. K. Mohapatra. Hazard Identification and Detection using Machine Learning Approach. 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS). 2020; 1239-1244. doi: 10.1109/ICICCS48265.2020.9121048..

Scikit-learn, Machine Learning in Python. [Online]. Available: https://scikit-learn.org/stable/ [Accessed: 10- November- 2019].

Mohammed Nazim Feroz, Susan Mengel. Phishing URL Detection Using URL Ranking. IEEE International Congress on Big Data. 2015.

Mahdieh Zabihimayvan and Derek Doran. Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection. International Conference on Fuzzy Systems (FUZZ-IEEE), New Orleans, LA, USA, June 2019.

Sakthivel M, Sivanantham S, Kamalraj R & Krishnamoorthy V. An Analysis of Machine Learning Depend on Q-MIND for Defencing the Distributed Denial of Service Attack on Software Defined Network. International Journal of Early Childhood Special Education. 2022; 14(05): 3769 – 3776.

Sivanantham S, Dhinagar S.R, Kawin P, Amarnath J. Hybrid Approach Using Machine Learning Techniques in Credit Card Fraud Detection. In: Suresh, P., Saravanakumar, U., Hussein Al Salameh, M. (eds) Advances in Smart System Technologies, Advances in Intelligent Systems and Computing. 2021; 1163. Springer, Singapore.

Akshaya V, Sathyapriya M, Ranjini Devi R, Sivanantham, S. Detecting Credit Card Fraud Using Majority Voting-Based Machine Learning Approach, In: Reddy, V.S., Prasad, V.K., Mallikarjuna Rao, D.N., Satapathy, S.C. (eds), Intelligent Systems and Sustainable Computing. Smart Innovation, Systems and Technologies, 2022 vol 289. Springer, Singapore.

Sivanantham S, Mohanraj V, Suresh, Y, & Senthilkumar J. Association Rule Mining Frequent-Pattern-Based Intrusion Detection in Network. Computer Systems Science and Engineering. 2023; 44(2):1617-1631.

Siva Kumar Depuru & Dr. K. Madhavi. Autoencoder Integrated Deep Neural Network for effective analysis of malware in distributed Internet of Things (IoT) Devices. International journal of analytical and experimental modal analysis. 2019; 11(12): 26-232.

D. Karthikeyan, V. Mohan Raj, J. Senthilkumar and Y. Suresh. Intrusion detection using ensemble wrapper filter based feature selection with stacking model. Intelligent Automation & Soft Computing. 2023; 35(1): 645–659.

Lakshmi Haritha. M & K. Ramani. Impact of Deep Learning on Localizing and Recognizing Handwritten Text in Lecture Videos. International journal of Advanced Computer Science and Applications; 12(4), 2021.

CH Prathima, R. Anusuya, M. Ram Kumar Prabhu (2022). Comprehensive Design Analysis of Digital Marketing in Agriculture Sector. International Journal of Early Childhood Special Education; 14(5):2022.

P. Yogendra Prasad, Dumpa Prasad, D Naga Malleswari (2022).Implementation of Machine Learning Based Google Teachable Machine in Early Childhood Education. International Journal of Early Childhood Special Education (INT-JECSE); 14(3):1308-5581.

P. Dhanalakshmi et al. (2022) Application of Machine Learning in Multi-Directional Model to Follow Solar Energy Using Photo Sensor Matrix. International journal of Photo energy; 9:1-9.