ISTANBUL KÜLTÜR UNIVERSITY

Graduation Project
CLASSIFICATION OF PHISHING WEBSITES BASED ON MACHINE LEARNING TECHNIQUES

Submitted By

Safa Orhan 1600003764
Eyüp Usta 1600003762

Project Advisor
Assis. Prof. Dr. Öznur Şengel

Department of Computer Engineering
İstanbul Kültür University

2021 Spring

May
2021

ISTANBUL KÜLTÜR UNIVERSITY

Graduation Project
CLASSIFICATION OF PHISHING WEBSITES BASED ON MACHINE LEARNING TECHNIQUES

Submitted By

Safa Orhan 1600003764
Eyüp Usta 1600003762

Advisor
Assis. Prof. Dr. Öznur Şengel

THE EXAMINATION COMMITTEE

Jury Member		Signature
1.	Assis. Prof. Dr. Öznur Şengel	…….
2.	Assis. Prof. Dr. Fatma Patlar Akbulut	…….
3.	Instructor Ezgi Demircan Türeyen	…….

ABSTRACT

Phishing is an attack aimed at stealing people's information. Phishing attacks send a legitimate-looking email to e-mails, and when users click the URL in that mail, their information is stolen. Many models have been developed against phishing attacks. However, in phishing attacks, people are the weak link, attackers exploit vulnerabilities, and attackers keep their attacks up-to-date. These and other problems require these models to be constantly updated and developed. In this study, we proposed one of the models that emerged using DL (DNN, RNN, and CNN) and ML (SVM, Decision Tree, KNN, Gaussian Naive Bayes, Random Forest, Bernoulli Naive Bayes) techniques against phishing attacks. Our aim in this project was to find the model that gives the closest result to 100% in comparison methods. Also, in this study, we preferred our dataset obtained from the UCI Machine Learning Repository site. Our data set consists of 11055 different samples and 30 features. For this purpose, we first trained our data set with ML algorithms and tested them using comparison methods. Then, we trained our data set with DL algorithms and tested it with comparison methods. As a result, we have proposed our model, which we think will give the best results against phishing attacks in line with the results we have achieved. In other words, the results show that the model we propose will be successful against phishing attacks.

ÖZET

Kimlik avı, insanların bilgilerini çalmayı amaçlayan bir saldırıdır. Kimlik avı saldırganları, e-postalara yasal görünümlü bir posta gönderir ve kullanıcılar bu postadaki URL'yi tıkladığında bilgileri çalınır. Kimlik avı saldırılarına karşı birçok model geliştirilmiştir. Ancak, kimlik avı saldırılarında insanlar zayıf halkadır, saldırganlar güvenlik açıklarından yararlanır ve saldırganlar saldırılarını güncel tutar. Bu ve benzeri sorunlar, bu modellerin sürekli güncellenmesini ve geliştirilmesini gerektirir. Bu çalışmada, kimlik avı saldırılarına karşı DL (DNN, RNN, and CNN) and ML (SVM, Decision Tree, KNN, Gaussian Naive Bayes, Random Forest, Bernoulli Naive Bayes) teknikleri kullanılarak ortaya çıkan modellerden birini önerdik. Bu projede amacımız karşılaştırma yöntemlerinde %100'e en yakın sonucu veren modeli bulmaktı. Ayrıca bu çalışmada, UCI Machine Learning Repository sitesinden elde ettiğimiz veri setimizi kullandık. Veri setimiz 30 özellik ve 11055 farklı örnekten oluşmaktadır. Bu amaçla önce veri setimizi ML algoritmaları ile eğitip, karşılaştırma yöntemleriyle test ettik. Ardından veri setimizi DL algoritmaları ile eğitip, karşılaştırma yöntemleri ile test ettik. Sonuç olarak, elde ettiğimiz sonuçlar doğrultusunda kimlik avı saldırılarına karşı en iyi sonuçları vereceğini düşündüğümüz modelimizi önerdik. Yani sonuçlar, önerdiğimiz modelin kimlik avı saldırılarına karşı başarılı olacağını gösteriyor.

ACKNOWLEDGEMENTS

We would like to acknowledge the effort of Assis. Prof. Dr. Öznur Şengel for her guidance. Express appreciation to all of those authors whose references we used in this research work. Also, we would like to thank Prof. Dr. Özgür Koray Şahingöz and Assis. Prof. Dr. Bahar İlgen for their contributions throughout our university education life. Finally, we would also like to thank IKU Computer Engineering students Batuhan Üçsu and Taha Anıl Pulat, who have been with us for four years and have always been with us.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS & ABBREVIATIONS

ACM: Association for Computing Machinery

AI: Artificial Intelligence

ANN: Artificial Neural Network

APWG: Anti-Phishing Working Group

ASM: Attribute Selection Measure

AUC: Area Under the Curve

AZ: Arizona

BNB: Bernoulli Naïve Bayes

BPTT: Backpropagation Through Time

CA: California

ccTLD: Country Code Top Level

CDS: Computing and Data Science

CNN: Convolutional Neural Network

DCNN: Deep Convolution Neural Network

DFS: Decision Function Shape

DL: Deep Learning

DNN: Deep Neural Network

DNS: Domain Name System

DT: Decision Tree

FN: False Negative

FP: False Positive

FTRL: Follow the Regularized Leader

GB: Gigabyte

GNB: Gaussian Naïve Bayes

GNN: Graph Neural Networks

HP: Hewlett-Packard

HTML: Hypertext Markup Language

HTTPS: Hypertext Transfer Protocol Secure

IC4ME2: International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering

ICCCI: International Conference on Computer Communication and Informatics

ICCCNT: International Conference on Computing, Communication and Networking Technologies

ICECTA: International Conference on Electrical and Computing Technologies and Applications

ICIET: International Conference on Innovation in Engineering and Technology

ICTCS: International Conference on New Trends in Computing Sciences

IEEE: Institute of Electrical and Electronics Engineers

IG: Information Gain

IJCNN: International Joint Conference on Neural Networks

IKU: Istanbul Kültür University

IP: Internet Protocol

IPDS: Intelligent Phishing Detection System

ISI: Intelligence and Security Informatics

ITNEC: Information Technology, Networking, Electronic and Automation Control Conference

KNN: K-Nearest Neighbors

LSTM: Long Short-Term Memory

MDPI: Multidisciplinary Digital Publishing Institute

ML: Machine Learning

MSE: Mean Squared Error

NB: Naïve Bayes

NLP: Natural Language Processing

PA: Pennsylvania

RAM: Random Access Memory

RBF: Radial Basis Function

RF: Random Forest

RNN: Recurrent Neural Network

SFH: Server Form Handler

SGD: Stochastic Gradient Descent

SKIMA: Software, Knowledge, Information Management and Applications

TN: True Negative

TP: True Positive

UCI: University of California Irvine Campus

UK: United Kingdom

URL: Uniform Resource Locator

USA: United States of America

1. INTRODUCTION

1.1. Problem Statement

Phishing aims to obtain users' credentials by cheating them via presenting fake web pages which appearing legitimate sites. For example, you receive mail from a bank or a company to your e-mail address. You trust it to be a well-known organization and click the URL in the incoming mail and your information is stolen. Phishing detection mechanisms are grouped into heuristics based, list based, machine learning, and visual similarity based based technical categories. In this project, a dataset will be preferred to compare DL and ML algorithms while detecting phishing websites. ML algorithms will be applied to determine anomalies such as DT, SVM, NB, KNN. Later, DL algorithms such as CNN, DNN and RNN will be applied. In addition, models from various articles will be applied. The most tested algorithm for assessment the performance of models with prearranged target data in machine learning is the confusion matrix to be created for each technique. Deep learning is a field that includes one or more layers and covers machine learning techniques. Also, precision, accuracy, F1-score, and recall will be evaluated to check deep learning and machine learning algorithms.

1.2. Project Purpose

To categorize phishing sites using ML and DL techniques, to facilitate the detection of such sites. To develop various models in the classification process. To reach the best result by testing these developed models on the data set. For this purpose, we will follow various stages in our project. First, we will examine and understand our data set. We will test ML techniques from his friend on our dataset and get the models with the best results. Later, we will test the models from various articles on our data set and expand our model scope. Finally, we will test DL techniques on our dataset and get the models that give the best results. As a result, we will recommend the best model or models after various evaluations. We will facilitate and speed up the detection of phishing sites through the model or models that give the best results.

1.3. Project Scope

There are many problems in phishing such as people, security vulnerabilities, and attackers constantly updating themselves. As a result of our work, we will propose a model or models against attackers trying to exploit people's weaknesses using seemingly safe URLs. Thus, it will be more effective against fake e-banking, e-commerce style URLs coming via e-mail.

1.4. Objectives and Success Criteria of the Project

To find and develop the model that gives the best result, that is, the closest to 100% value. It is our success criterion that the model we will obtain as a result of the studies has the closest value to 100% in comparison methods. It is impossible to obtain 100% value in such a study.

1.5. Report Outline

In the related work part, we will first look at the methods of current work on phishing. Then we will look at the general problems of current studies. Finally, we will compare the methods in existing studies with our own method. The methodology part will follow. In this part, we will first talk about the studies we tested ML and DL techniques on our data set. Later, we will talk about model studies that we took from the papers and tested them. We will also talk about testing models on different data sets. Later, we will talk about ML and DL techniques we use in this study. We will also explain our dataset. Finally, we will talk about the model or models we proposed. The experimental result will follow this part. In this part, we will talk about the results of the model or models we have proposed. After that, the discussion part will follow, and we will talk about the inferences we have obtained from the results we have obtained in that part. Finally, the conclusion part will come. In the conclusion part, we will talk about the results we obtained in our study in general.

2. RELATED WORK

In this part, we will examine existing systems and their overall problems and compare them with our proposed method.

2.1. Existing Systems

P. Yang and his friends [12] used a network of convolutional and recurrent layers. They also used character level embeddings. They used a dataset of about one million samples and achieved an accuracy of 98.61%.

Wang and his friends [28] used a network structure consisting of convolutional and recurrent layers. They also used character level embeddings. They used a dataset of about five million samples and achieved an F1 score of 95.52%.

Firstly, Adebowale and his friends [4] applied the feature extraction and machine learning steps to their websites. Then, IPDS fed the DL system with the features it received. Then they applied the trained LSTM-CNN network to detect which websites were what. The sites were classified according to the differences between the extracted features and the IPDS model. They obtained an accuracy of 93.28% in IPDS (CNN + LSTM). As the data set, they preferred a dataset containing 1000000 URL samples and a dataset of 10000 images that they collected themselves from legitimate and phishing sites.

S. Yang [16] used a model that contains 5 LSTM layers with 128 nodes each and a regular CNN model. In the LSTM model the author achieved 99.14% accuracy and in the CNN model the author achieved 97.42% accuracy.

Opara and her friends [7] used Embedding and Convolutional layers in their model. They achieved 93% accuracy on the test set.

Kumar and his friends [10] used a data set that contains 100000 URLs for training their models. The authors used three different models which are Logistic Regression, GNB, and RF. In Logistic Regression model 97.7% accuracy is achieved, in the Random Forest model 98.03% accuracy is achieved and lastly using Gaussian model 97.18% accuracy achieved.

2.2. Overall Problems of Existing Systems

Phishing is a method used to obtain information by directing people to the site with legitimate looking URLs sent to e-mails. Looking at the overall problems of existing systems on phishing:

- For phishing sites, the weakest link in the structure is people. That is why phishing sites target people. For example, you think that the URL in a mail sent to your email is legitimate and you click it, and you are redirected to a site and thus your information is captured [6].
- People trying to steal people's information using phishing are constantly updating their methods. Because they are aware that the method they use will be resolved after a while [25].
- Phishing attacks try to exploit technical vulnerabilities. So, they try to make the URL look legit. In addition, software phishing detection techniques make the performance of phishing attacks unknown. This situation causes security vulnerabilities in many organizations [6].
- Researchers is hold the positive rate low in List-based methods, but lists are not comprehensive and cannot detect an attack on day zero [7].
- No matter how comprehensive the feature sets are, if they are not updated regularly, they will fail to detect new phishing attacks [7].

There are various problems such as these problems and systems/models are tried to be developed by calculating these problems.

2.3. Comparison Between Existing and Proposed Method

Table 2.1 shows the accuracy of our models and other proposed approaches based on the precision, accuracy, f-measure, and recall in the papers. In Order to make a comparison, we calculated these four metrics based on our experimental results. Our experimental values showed that the best model to use in machine learning is Random Forest. Reference [10] has obtained the highest precision over our Random Forest model but our Random Forest model has achieved the highest accuracy, f-measure and recall over Reference [10]. In deep learning, RNN Model 2 has achieved the best scores of the metrics over our other deep learning models but not over other proposed approaches.

Reference [16] has achieved the best accuracy and recall. Reference [12] has achieved the best precision and f1-score. Reference [4] and Combined Model 3 both are using LSTM and CNN layers. Reference [4] has achieved 93.28% accuracy, Combined Model 3 has achieved 96.26% accuracy. Combined Model 3 has achieved the 0.96 of precision, recall and f1-score whereas Reference [4] has achieved 0.93 on these metrics.

Table 2.1: Comparison of methods

Model	Accuracy	Precision	Recall	F1-Score
Our RF Model	99.09%	0.9895	0.9940	0.9917
RNN Model 2	97.27%	0.9769	0.9694	0.9732
Combined Model 3	96.26%	0.9635	0.9678	0.9656
[12]	98.61%	0.9941	0.9857	0.99
[28]	95.60%	0.9733	0.9378	0.9552
[4]	93.28%	0.9327	0.9330	0.9329
Random Forest [10]	98.03%	1.0	0.96	0.98
Logistic Regression [10]	97.7%	1.0	0.96	0.98
Gaussian Naïve Bayes [10]	97.18%	1.0	0.95	0.97
[7]	93%	0.92		0.91
CNN [16]	97.42%	0.9648	0.9723	
LSTM [16]	99.14%	0.9874	0.9891	

3. METHODOLOGY

In this section, we will talk about ML and DL techniques we use to obtain our model in our project, other models we have tried, and the data sets we use.

3.1. Overview of the Dataset/Model

3.1.1. ML Techniques

A. Support Vector Machine (SVM)

We tested to get the best accuracy value by combining methods such as kernel, C, degree, gamma, and decision function shape. We tested the combinations we made on our dataset [32] we got from the UCI machine learning repository site. We tried values between 15% and 35% as test size in increments of five.

- Kernels: linear, sigmoid, polynomial, RBF
- C: 1, 2, 3, 4, 5
- Degree: 1, 2, 3, 4, 5, 6
- Gamma: scale, auto
- Decision function shape (DFS): ovo, ovr

The test size for which we got the best accuracy value was 25%. When we examine Figure 3.1, the best average result is the kernel RBF, and the worst average result is the kernel sigmoid. When we examine Figure 3.2, we see that as the C value approaches 3, the accuracy value increases. After the C value exceeds 3, the accuracy value starts to decrease. The combination that works the best is "kernel: RBF, C: 1, degree: 2, gamma: scale, DFS: ovo" and this combination has an accuracy of 98.37%. The combination that works worst is "kernel: sigmoid, C: 3, degree: 1, gamma: scale, DFS: ovo" and this combination has an accuracy of 83.78%. We used 2400 different model combinations in SVM and achieved an average accuracy of 92.42%.

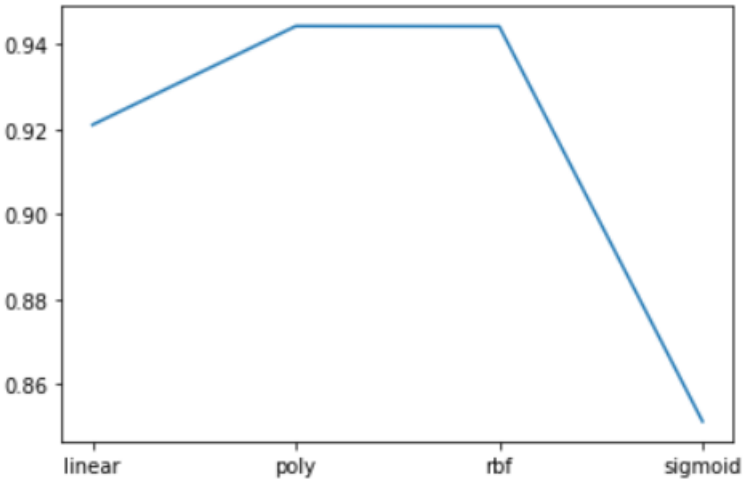

Figure 3.1: Average accuracy values according to kernels

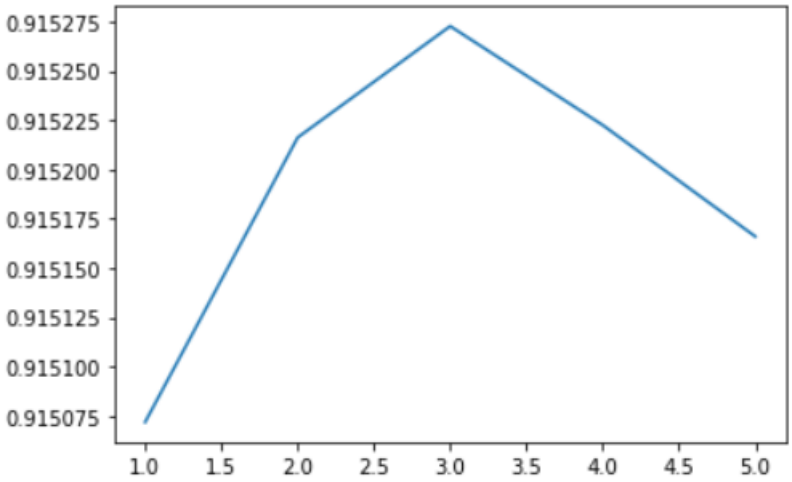

Figure 3.2: Average accuracy values according to C values

B. Gaussian Naïve Bayes

We tested the combinations we made on our dataset [32] we got from the UCI machine learning repository site. We tried values between 15% and 35% as test size in increments of five.

The test size for which we got the best accuracy value was 30%. We got 61.34% as the accuracy value. This result is not a good result, as we have no chance of being successful against phishing attacks with this accuracy rate. Therefore, gaussian naive bayes is not a method that will work for us in this study.

C. Linear SVM

We tried to get the best accuracy value by combining methods such as losses, penalty, C, and multi class. We tested the combinations we made on our dataset [32] we got from the UCI machine learning repository site. We tried values between 15% and 35% as test size in increments of five.

- Losses: hinge, squared hinge
- Penalty: L2
- C: 1, 2, 3, 4, 5
- Multi class: ovr, crammer singer

The test size for which we got the best accuracy value was 15%. The combination that works the best is "losses: squared hinge, penalty: L2, C: 1, multi class: ovr" and this combination has an accuracy of 93.97%. When we examine Figure 3.3, as the value of C changes, a bumpy graphic emerges, but the C that grant the best average result is 4. The combination that works worst is "losses: hinge, penalty: L2, C: 4, multi class: ovr" and this combination has an accuracy of 93.03%. We used 100 different model combinations in linear SVM and achieved an average accuracy of 93.11%.

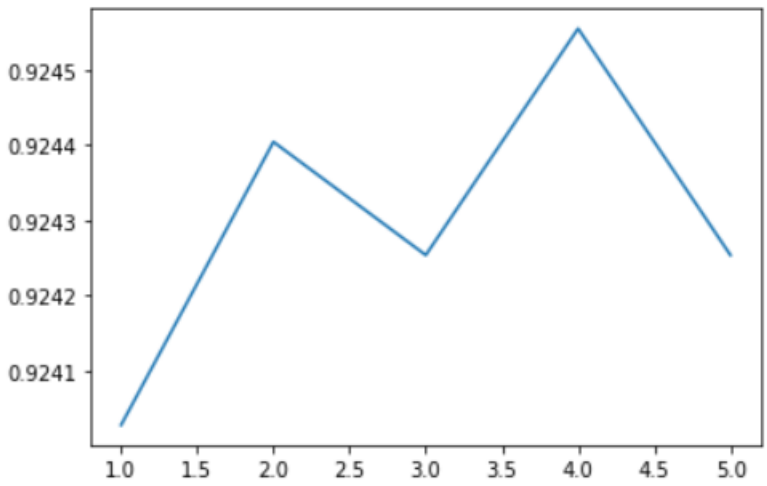

Figure 3.3: Average accuracy values according to C values

D. Bernoulli Naïve Bayes

We tried to get the best accuracy value by combining methods such as alpha, binarize, and fit prior. We tested the combinations we made on our dataset [32] we got from the

UCI machine learning repository site. We tried values between 15% and 35% as test size in increments of five.

- Alpha: 0, 1, 2, 3, 4, 5, 7, 9, 11
- Binarize: 0, 1, 2, 3, 4, 5, 7, 9, 11
- Fit prior: true, false

The test size for which we got the best accuracy value was 15%. The combination that works the best is "alpha: 0, binarize: 0, fit prior: false" and this combination has an accuracy of 91.86%. The combination that works worst is "alpha: 0, binarize: 1, fit prior: true" and this combination has an accuracy of 56.01%. We used 810 different model combinations in bernoulli naive bayes and achieved an average accuracy of 60.36%.

E. K-Nearest Neighbors (KNN)

We tried to get the best accuracy value by combining methods such as K, weights, algorithm, and metric. We tested the combinations we made on our dataset [32] we got from the UCI machine learning repository site. We tried values between 15% and 35% as test size in increments of five.

- K: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- Algorithm: kd tree, ball tree, brute, auto
- Weights: uniform, distance
- Metric: chebyshev, manhattan, minkowski, euclidean

The test size for which we got the best accuracy value was 25%. When we examine Figure 3.4, Manhattan gives the best average accuracy value and Chebyshev gives the worst average accuracy value. When we examine Figure 3.5, we see that as the K increases, the accuracy value decreases. Combination that works the best is "K: 5, weights: distance, algorithm: auto, metric: Manhattan" and this combination has an accuracy of 98.91%. The combination that works worst is "K: 8, weights: uniform, algorithm: auto, metric: Chebyshev" and this combination has an accuracy of 78.17%. We used 1440 different model combinations in KNN and achieved an average accuracy of 95.99%.

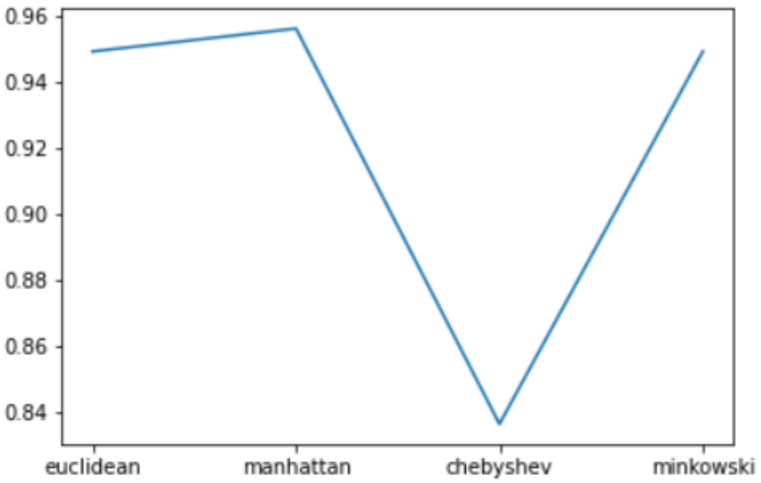

Figure 3.4: Average accuracy values according to metrics

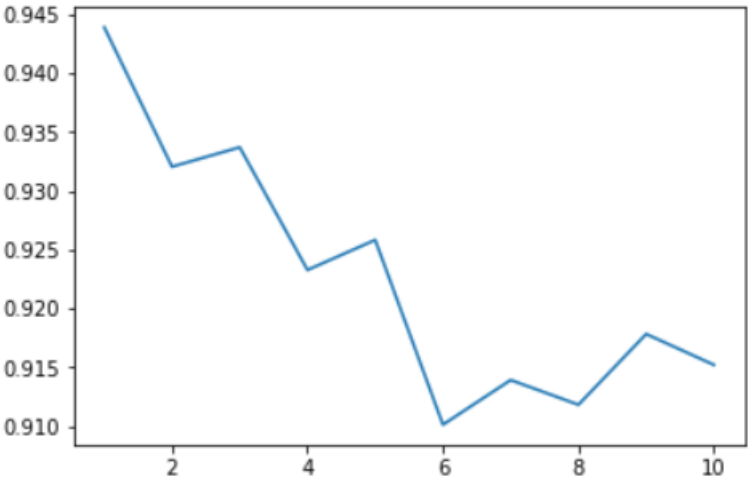

Figure 3.5: Average accuracy values according to K values

F. Decision Tree

We tried to get the best accuracy value by combining methods such as max features, criterion, class weight, and splitter. We tested the combinations we made on our dataset [32] we got from the UCI machine learning repository site. We tried values between 15% and 35% as test size in increments of five.

- Criterion: gini, entropy
- Max features: auto, none, log2, sqrt
- Splitter: random, best
- Class weight: none, balanced subsample, balanced

The test size for which we got the best accuracy value was 25%. The combination that works the best is "max features: none, criterion: entropy, splitter: best, class weight: none" and this combination has an accuracy of 98.76%. The combination that works worst is "max features: log2, criterion: gini, splitter: random, class weight: none" and this combination has an accuracy of 95.64%. When we examine Figure 3.6, we see the average accuracy values based on the combination of max features and class weight. We used 160 different model combinations in the decision tree and achieved an average accuracy of 95.64%.

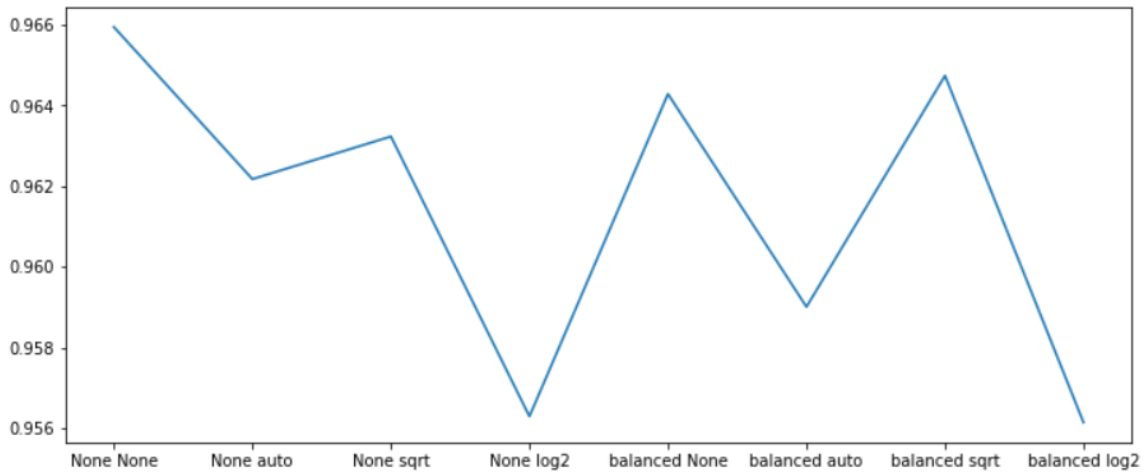

Figure 3.6: Average accuracy values according to max features and class weights

G. Random Forest

We tried to get the best accuracy value by combining methods such as max features, criterion, class weight, and warm start. We tested the combinations we made on our dataset [32] we got from the UCI machine learning repository site. We tried values between 15% and 35% as test size in increments of five.

- Max features: auto, none, log2, sqrt
- Class weight: none, balanced subsample, balanced
- Warm start: true, false
- Criterion: entropy, gini

The test size for which we got the best accuracy value was 25%. When we examine Figure 3.7, the class weight that gives the worst average accuracy value is none. Class

weight, which gives the best average accuracy value, is also a balanced subsample. The combination that works the best is "max features: auto, criterion: gini, class weight: balanced subsample, warm start: true" and this combination has an accuracy of 99.09%. The combination that works worst is "max features: none, criterion: entropy, class weight: balanced, warm start: true" and this combination has an accuracy of 96.68%. We used 240 different model combinations in the random forest and obtained an average accuracy of 97%.

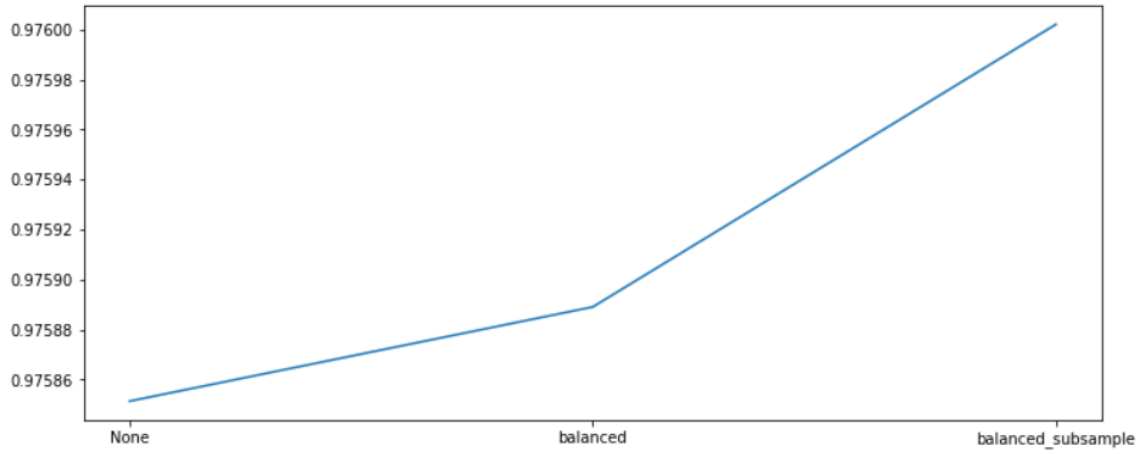

Figure 3.7: Average accuracy values according to class weights

3.1.2. Paper Implementation

We looked at the accuracy values by applying the models in some articles that we used as a reference to make our work better, to our own data set. We tested the models we get on our dataset [32] we got from the UCI machine learning repository site.

First study [19], models were made using KNN, multilayer perceptron, random forest, and SVM. Also, the data set used by the authors is the same as ours. The authors used "K: 5, weights: uniform, algorithm: auto" as the KNN model. The authors achieved an accuracy value of 94.81% in this model, while we obtained an accuracy value of 93.72%. The authors tried 3 different models of the multilayer perceptron. The first model is "hidden layers: 30, max iterations: 3000". The authors obtained an accuracy value of 97.22% in the first model, while we obtained an accuracy value of 95.41%. The second model is "hidden layers: 150, max iterations: 1000". The authors achieved an

accuracy of 90.28% in the second model, while we obtained an accuracy value of 96.98%. Third model is "hidden layers: 100, max iterations: 1000". The authors achieved an accuracy value of 96.71% in the third model, while we obtained an accuracy value of 97.04%. The authors used 2 models in the random forest. First model "n estimators: 7, max depth: 11, criteria: entropy". The authors obtained an accuracy value of 95.25% in the first model, while we obtained an accuracy value of 95.17%. Second and final model "n estimators: 7, max depth: 8, criteria: entropy". The authors obtained an accuracy value of 89.16% in the second model, while we obtained an accuracy value of 95.38%. The authors tried 3 different models of SVM. The first model is "C: 1, kernel: linear". While the authors obtained an accuracy value of 92.71% in the first model, we obtained an accuracy value of 92.67%. The second model is "C: 1, kernel: polynomial, degree: 1". While the authors achieved 92.57% accuracy in the second model, we obtained 92.58% accuracy. The third and final model is "C: 1, kernel: polynomial, degree: 2". While the authors achieved an accuracy value of 93.88% in the third model, we obtained an accuracy value of 94.21%. In addition, in this third model, we observed what result we would get as we increased the degree. When we examine Table 3.1, we see that the accuracy value increases as the degree increases. As of degree 10, the accuracy value starts to decrease.

Table 3.1: The accuracy value according to the degree of the "C: 1, kernel: polynomial" model

Degree	Accuracy
1	92.58%
2	94.21%
3	95.17%
4	95.62%
5	95.99%
6	96.20%
7	96.38%
8	96.56%
9	96.71%
10	96.59%

In the second study [21] we looked at the authors applied the model shown in Figure 3.8 [21]. The model first enters 17 features. These 17 features are given in Table 3.2. Then the properties enter the hidden layer with 3 stages. In this layer, the net input is calculated by multiplying the inputs by their weights. The properties then move to the log-sigmoid layer. Here, first the activation function operates, and the results are multiplied by the input weights to find a net input. Finally, it goes to the output layer properties. The results are compared with the predetermined threshold value. If the obtained value is greater than the threshold, it is a legitimate site, but if the obtained value is less than the threshold, it is a phishing site.

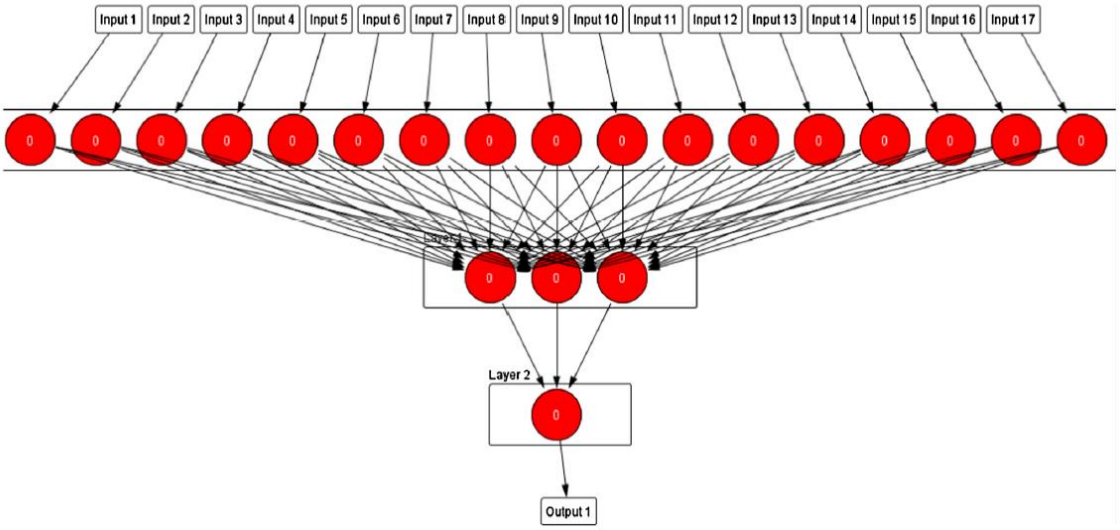

Figure 3.8: Structure produced when the Epoch value is 500.

Table 3.2: 17 features used in the second study.

Features
Abnormal URL
SFH
HTTPS Token
Iframe
Right Click
Pop up Window
Redirect
Having IP Address
Having Sub Domain
URL Length
Request URL
Age of Domain
Having @ Symbol
URL of Anchor
DNS Record
Prefix Suffix
Web Traffic

The authors used a different data set than ours. The data set used by the authors is from starting point directory, Millersmiles, yahoo directory, and Phishtank [21]. We tested this model by increasing the epoch value and as the epoch value, we tried the values between 50 and 1000 in increments of fifty. We got the best result with 1000 epochs and the accuracy was 92.72%. We got the worst result with 50 epochs and the accuracy was 56.09%. In addition, in Table 3.3, we compared the accuracy values obtained by the

authors in epoch values with the accuracy values we obtained. As a result, we observed that as the epoch value increased, the accuracy value increased and we showed this in Figure 3.9.

Table 3.3: The accuracy values we and the authors obtained according to the epoch.

Epoch	Our Accuracy	Authors' Accuracy
50	56.09%	91.32%
100	87.52%	92.33%
200	89.07%	93.07%
500	90.90%	93.45%
1000	92.72%	94.07%

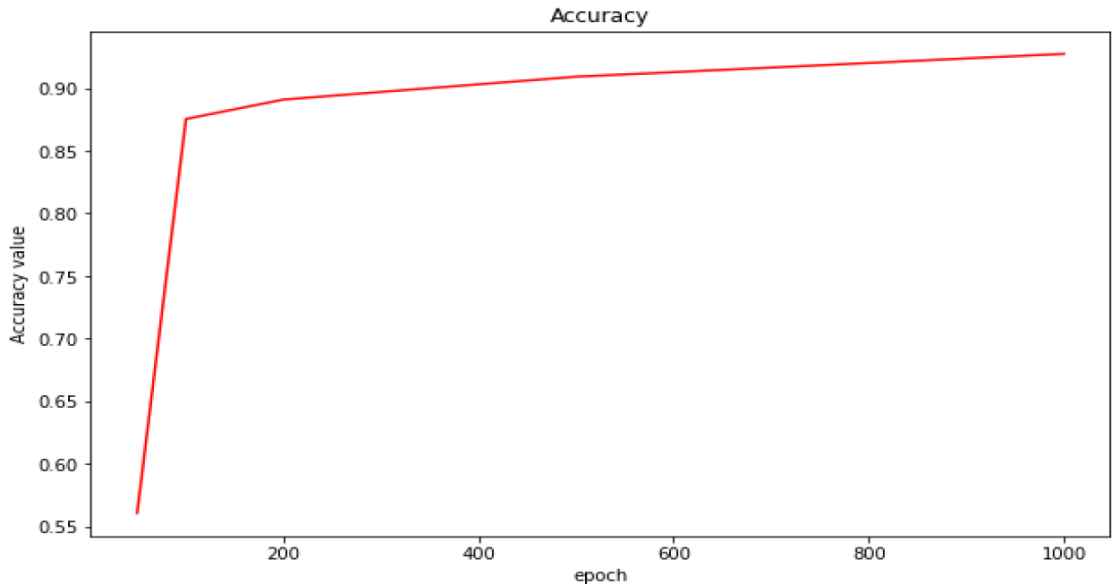

Figure 3.9: Accuracy values according to epoch values

In the third study [18] we looked at the authors used the DCNN model. As shown in Figure 3.10 [18], the DCNN model is used. Also, the data set used by the authors is the same as ours. While the authors achieved an accuracy value of 99.3% in this model, we obtained an accuracy value of 55.43%. As a result, we could not get the efficiency we wanted from this model and we got bad results.

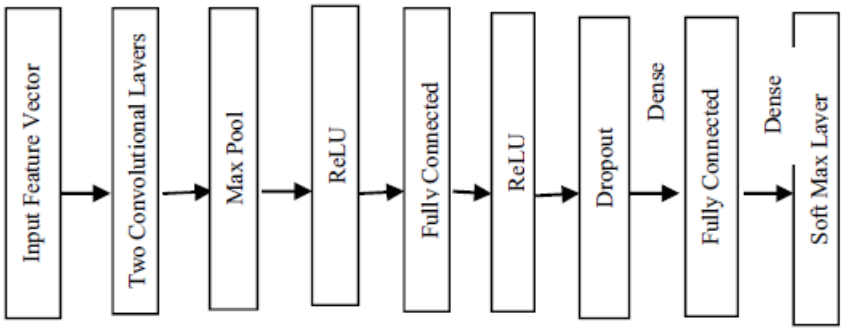

Figure 3.10: DCNN model

In the fourth and last study [20], the authors categorized the features used in phishing detection and applied naive bayes and the decision tree to these categories. You can see these categories in detail in Figure 3.11 [20]. Also, the data set used by the authors is the same as ours. The accuracy values obtained by the authors as a result of using naive bayes and the decision tree and the accuracy values obtained as a result of our use are shown in Table 3.4 and Table 3.5. When we look at the tables, the authors achieved successful results in other categories except the JavaScript and HTML category, while we did not achieve good results in any category except the decision tree experiment for the abnormality category and the address bar category. As a result, we cannot say that we have achieved a good result in this model.

Category	Feature Name	Value
Address bar	Having IP Address	1,0
	Having long url	1,0,-1
	Uses ShortningService	0,1
	Having '@' Symbol	0,1
	Double slash redirecting	0,1
	Having Prefix Suffix	-1,0,1
	Having Sub Domain	-1,0,1
	SSLfinal State	-1,1,0
	Domain registration Length	0,1,-1
	Favicon	0,1
	Is standard Port	0,1
	Uses HTTPS token	0,1
Abnormality	Request URL	1,-1
	Abnormal URL anchor	-1,0,1
	Links in tags	1,-1,0
	SFH	-1,1
	Submitting to email	1,-1
	Abnormal URL	1,0
HTML-JavaScript	Redirect	0,1
	on mouseover	0,1
	RightClick	1,-1
	popUpWindow	-1,1
	Iframe	0,1
Domain	Age of domain	-1,0,1
	on DNS Record	1,0
	Web traffic	-1,0,1
	Page Rank	-1,0,1
	Google Index	0,1
	Links pointing to page	1,0,-1
	Statistical report	1,0

Figure 3.11: Table showing the categorization of features used in phishing detection.

Table 3.4: Accuracy values by categories as the result of the use of naïve bayes

Category	Our Accuracy	Authors' Accuracy
Address Bar	89.59%	89.95%
Abnormality	72.20%	88.45%
HTML - JavaScript	56.01%	54.12%
Domain	69.88%	80.35%

Table 3.5: Accuracy values by categories as the result of the use of decision tree

Category	Our Accuracy	Authors' Accuracy
Address Bar	90.32%	90.19%
Abnormality	87.63%	89.05%
HTML - JavaScript	57.43%	58.02%
Domain	72.74%	81.55%

3.1.3. DL Techniques

One subfield of machine learning is called DL. DL architectures as GNN, RNN, DNN, and CNN can be applied to most of the fields such as NLP, machine translation, computer vision e.g. In our project, we applied three of the network architectures to our dataset which are RNN, DNN, and CNN. Also, we tested the models we get on our dataset [32] we got from the UCI machine learning repository site. Activation functions in Table 3.6 are used for output layer. When compiling models, the loss functions and optimizers on Table 3.6 are used. In addition, the accuracy values of the models we use for DL, used the features, epoch values and test sizes are shown in Table 3.7.

Table 3.6: Activation Functions, Loss Functions, and Optimizers

Activation Functions	Optimizers	Loss Functions
Softplus	SGD	Binary Cross Entropy
Softsign	Rmsprop	Categorical Cross Entropy
Selu	Adam	Hinge
Elu	Adadelta	Squared Hinge
Exponential	Adagrad	Huber
Tanh	Adamax	
Sigmoid	Nadam	
Relu	FTRL	

Table 3.7: Model features, epoch values, test sizes and accuracy values

Models	Features	Accuracy	Test Size	Epoch
DNN Model 1	Categorical	55.84%	30%	150
DNN Model 1	Categorical	75.76%	15%	10
DNN Model 1	Categorical	75.54%	35%	10
DNN Model 2	Categorical	75.79%	15%	10
DNN Model 2	Categorical	75.40%	35%	10
DNN Model 2	Non-categorical	98.84%	30%	150
RNN Model 1	Non-categorical	98.81%	30%	150
RNN Model 2	Non-categorical	98.31%	30%	150
RNN Model 3	Non-categorical	96.85%	30%	150
RNN Model 4	Non-categorical	96.21%	30%	150
CNN Model 1	Non-categorical	93.19%	30%	150
CNN Model 2	Non-categorical	92%	30%	150
Combined Model 1	Non-categorical	90.02%	30%	150
Combined Model 2	Non-categorical	96.29%	30%	150
Combined Model 3	Non-categorical	96.26%	30%	150

A. Convolutional Neural Networks (CNN)

In Figure 3.12, dataset features are not turned into categorical values and 320 different combinations of activation functions, optimizers and loss functions are tested. In the combinations that are using elu and softplus as activation functions in the output layer performed accuracy below 55%. Also, in the combinations that are using the FTRL as an optimizer, accuracy is below 55%. In the other different combination of the CNN Model 1, accuracy between 80% and 94% was achieved. The best performing model combination is “optimizer: adamax, activation function in output layer: tanh, loss function: huber” with 93.19% of accuracy.

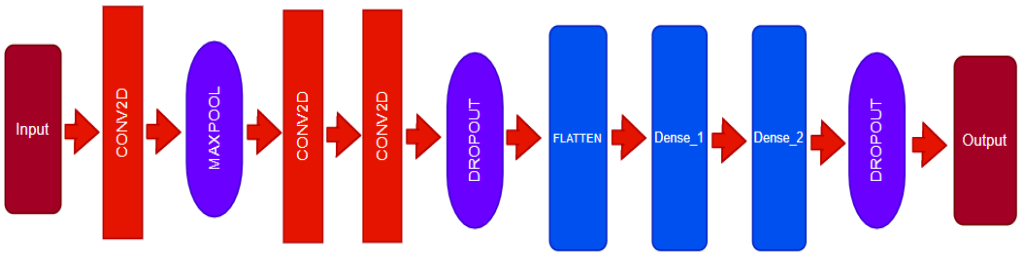

Figure 3.12: CNN Model 1

In Figure 3.13, the model is tested with "loss function: binary cross entropy, optimizer: adam" and epoch number of 150. 92% accuracy is achieved as a train accuracy and 89.79% accuracy is achieved as a test accuracy.

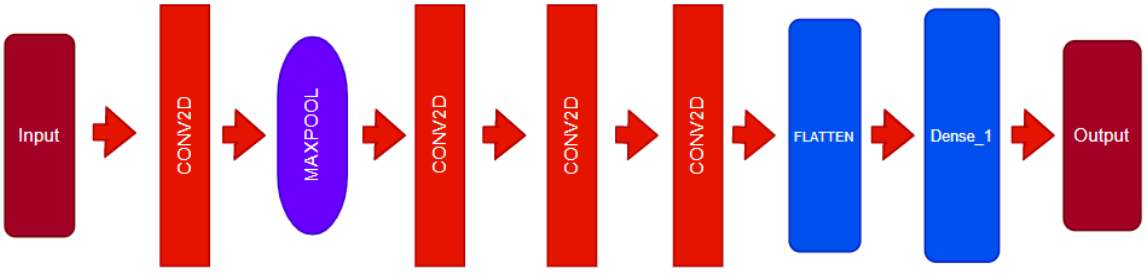

Figure 3.13: CNN Model 2

B. Deep Neural Networks (DNN)

In Figure 3.14, dataset features are turned into categorical features and 320 different combinations of activation functions, optimizers and loss functions are tested along with the two different test sizes which were 15% and 35%. For the test size 15%, we achieved 75.76% accuracy as the best accuracy with an average accuracy of 69.04% over different combinations and for the test size 35%, we achieved 75.54% accuracy as the best accuracy with an average accuracy of 69.19% over different combinations. The model in DNN Model 1 is tested with "activation function of the output layer: sigmoid, loss function: binary cross entropy, optimizer: SGD". The test size was 30% this time and the accuracy was 55.84%.

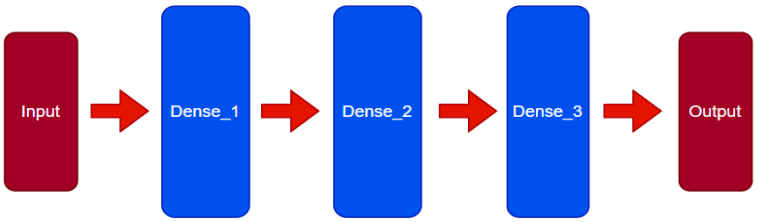

Figure 3.14: DNN Model 1

In Figure 3.15, dataset features are turned into categorical features and 320 different combinations of activation functions, optimizers and loss functions are tested along with the two different test sizes which were 15% and 35%. For the test size 15%, we achieved 75.79% accuracy as the best accuracy with an average accuracy of 69.29% over different combinations and for the test size 35%, we achieved 75.40% accuracy as the best accuracy with an average accuracy of 68.96% over different combinations. With the result of DNN Model 1 and DNN Model 2 we interpreted that the test size does not affect our accuracy much.

DNN Model 2 is used again. Dataset features are not turned into categorical values and 320 different combinations of activation functions, optimizers and loss functions are tested. In deep layers, neuron numbers are incremented and as a test size 30% is chosen. 98.84% accuracy achieved as the best accuracy. The combination of the best performing model is sigmoid as an activation function in the output layer, adamax as an optimizer and huber is used as a loss function. Average accuracy of 81.42% is achieved over 320 different combinations over the model. The model performed accuracy below 55% when using FTRL as an optimizer and when using relu as an activation function in output layer. Result of DNN Model 2 with more neurons in the deep layers and not turning dataset features into categorical showed that adding more neurons and not turning dataset features into categorical does help increase the accuracy by a significant amount.

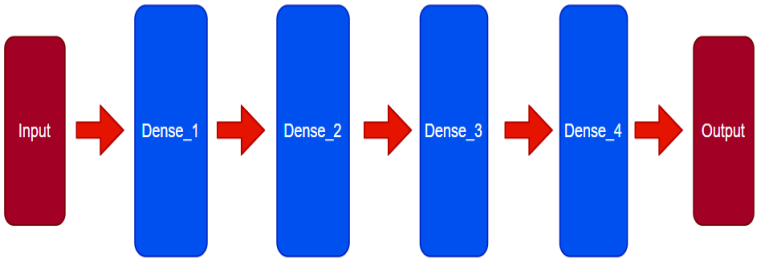

Figure 3.15: DNN Model 2

C. Recurrent Neural Networks (RNN)

In Figure 3.16, the model is tested with the 320 different combinations of activation functions, optimizers, and loss functions. In the combinations that are using the FTRL as an optimizer performed accuracy below 55%. In other combinations model performed accuracy between 90% and 99%. The best performing model combination is "optimizer: adamax, activation function in output layer: tanh, loss function: binary cross entropy" with accuracy of 98.81%.

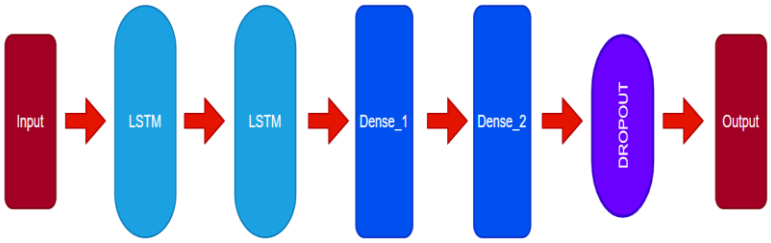

Figure 3.16: RNN Model 1

In Figure 3.17, the model is tested with the 320 different combinations of activation functions, optimizers, and loss functions. In the combinations that are using the FTRL as an optimizer or relu, selu and elu as an activation function in the output layer performed accuracy below 55%. In other combinations model performed accuracy between 90% and 98%. The best performed model combination is "loss function: binary cross entropy, optimizer: adamax" with 98.31% accuracy.

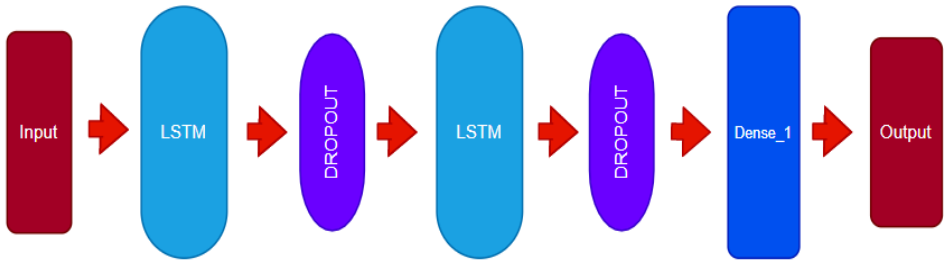

Figure 3.17: RNN Model 2

In Figure 3.18, the model is tested with "loss function: binary cross entropy, optimizer: SGD" and epoch number of 150. 96.85% accuracy is achieved.

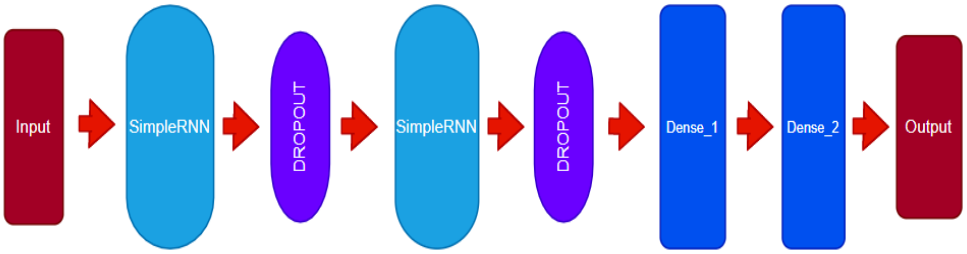

Figure 3.18: RNN Model 3

In Figure 3.19, the model is tested with "loss function: binary cross entropy, optimizer: SGD" and epoch number of 150. 96.21% accuracy is achieved.

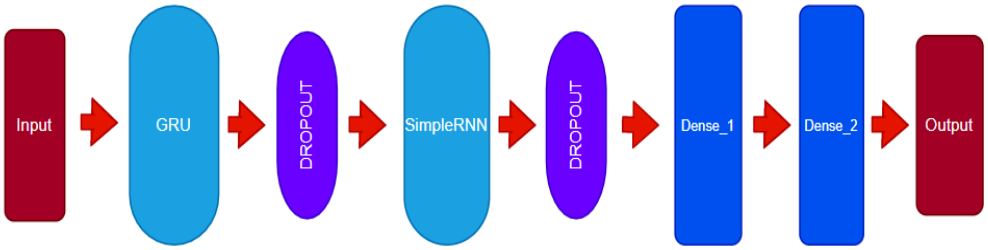

Figure 3.19: RNN Model 4

D. Combined Models

Using the best combinations of CNN Model 1 and RNN Model 1 we created two different models. First by appending the CNN Model 1 to the end of RNN Model 1 and second by appending the RNN Model 1 to the end of CNN Model 1. Model compiled

with “output layer activation function: softplus, optimizer: adamax, loss function: binary cross entropy”. In the Combined Model 2, 96.29% accuracy is achieved and in the Combined Model 1 90.02% accuracy is achieved.

In Figure 3.20, the Combined Model 3 is tested with “output layer activation function: softplus, optimizer: adamax, loss function: binary cross entropy” and epoch number of 150. 96.26% accuracy is achieved.

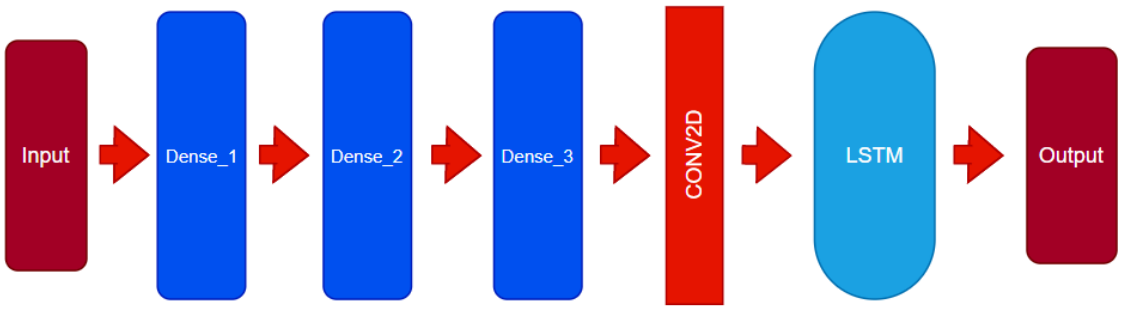

Figure 3.20: Combined Model 3

3.1.4. Model Tests with Different Datasets

First dataset [22] is taken from data mendeley. It has 48 features, and the dataset size is 10000 samples. DNN Model 2, RNN Model 1, and CNN Model 1 are tested with this dataset. DNN Model 2 performed 95.43%, RNN Model 1 performed 93.93%, CNN Model 1 performed 88.87% accuracy. Second dataset [34] is taken from kaggle. It has 14 features, and the dataset size is 11000 samples. DNN Model 2, RNN Model 1, and CNN Model 1 are tested with this dataset. DNN Model 2 performed 89.45%, RNN Model 1 performed 99.12%, CNN Model 1 performed 99.68% accuracy. Third dataset [36] is taken from data mendeley. It has 35 features, and the dataset size is 13071 samples. DNN Model 2, RNN Model 1, and CNN Model 1 are tested with this dataset. DNN Model 2 performed 94.11%, RNN Model 1 performed 93.19%, CNN Model 1 performed 92.76% accuracy. Fourth dataset [35] is taken from data mendeley. It has 87 features, and the dataset size is 11430 samples. DNN Model 2, RNN Model 1, and CNN Model 1 are tested with this dataset. DNN Model 2 performed 75.51%, RNN Model 1 performed 77.34%, CNN Model 1 performed 77.81% accuracy. Fifth and last dataset [37] is taken

from Canadian Institute for Cybersecurity. It has 79 features, and the dataset size is 15367 samples. DNN Model 2, RNN Model 1, and CNN Model 1 are tested with this dataset. DNN Model 2 performed 48.88%, RNN Model 1 performed 48.88%, CNN Model 1 performed 48.88% accuracy. In addition, in Table 3.8, the results of the models tested on different datasets are compared with the results of the testing on our dataset.

Table 3.8: Accuracy values of models in different datasets

Dataset	DNN Model 2	RNN Model 1	CNN Model 1
First Dataset [22]	95.43%	93.93%	88.87%
Second Dataset [34]	89.45%	99.12%	99.68%
Third Dataset [36]	94.11%	93.19%	92.76%
Fourth Dataset [35]	75.51%	77.34%	77.81%
Fifth Dataset [37]	48.88%	48.88%	48.88%
Our Dataset [32]	98.84%	98.81%	93.19%

3.1.5. Our Dataset Description

One of the most faced difficulties by the research world is the rarity of trustable datasets. In our project we have used the Phishing Websites Dataset [32] that is published on the UCI Machine Learning Repository by Rami M. Mohammad and his friends [2].

In Phishing Websites Data Set, there are 11055 different samples of legitimate and phishy websites. 6157 of the websites samples belongs to legitimate class of websites and 4898 of the websites samples belongs to phishy class of websites. Looking at Figure 3.21, we see a graph showing how many of the examples are legitimate and how many are phishing. We have a dataset with 30 features and these features are divided into 4 categories.

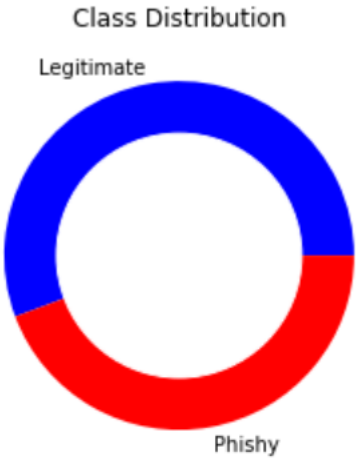

Figure 3.21: Class distribution of our dataset

A. Address Bar Based Features

- Using the IP Address: The URL's domain name is “phishing” if it contains an IP address, and “legitimate” if it does not.

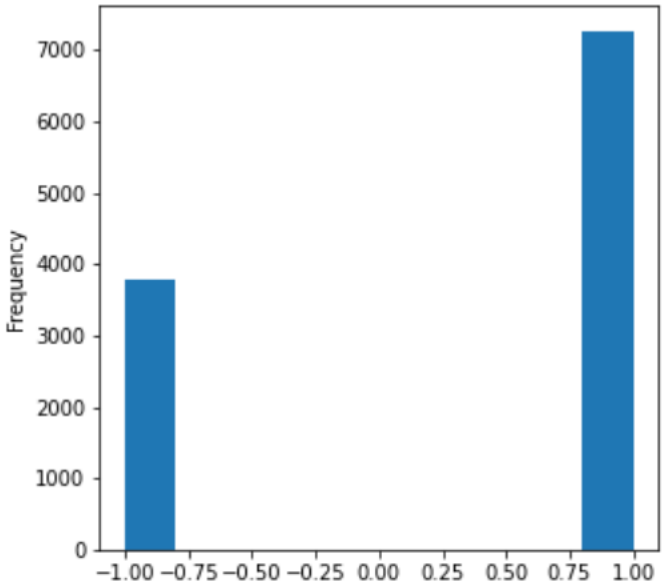

Figure 3.22: Frequency values of Using the IP Address

- Long URL to Hide the Suspicious Part: If the URL length is more than 53 characters, it is “phishing”, otherwise it is “legitimate.”

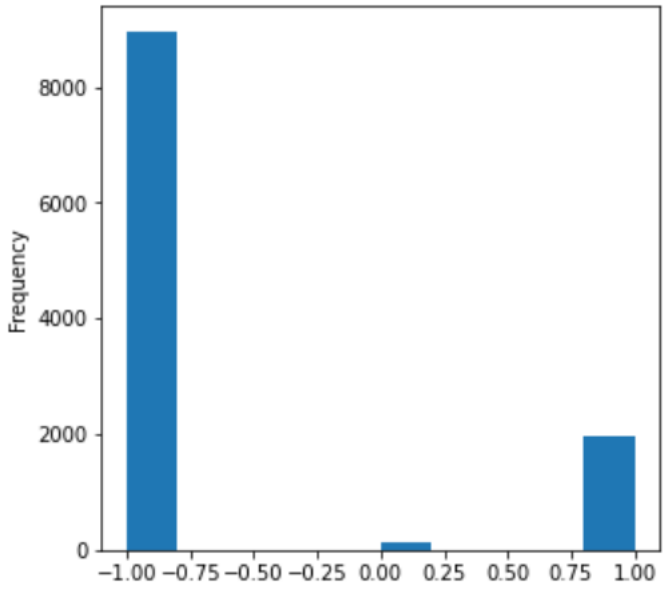

Figure 3.23: Frequency values of Long URL to Hide the Suspicious Part

- Using URL Shortening Services "TinyURL": It is "phishing" if the shortening service is used to hide the URL length, otherwise it is "legitimate."

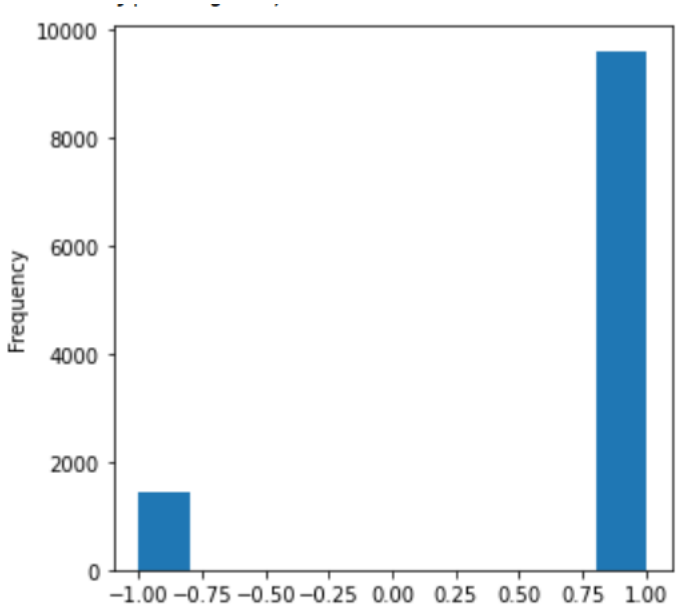

Figure 3.24: Frequency values of Using URL Shortening Services "TinyURL"

- URL's having "@" Symbol: If there is "@" in the URL, it is "phishing", otherwise it is "legitimate."

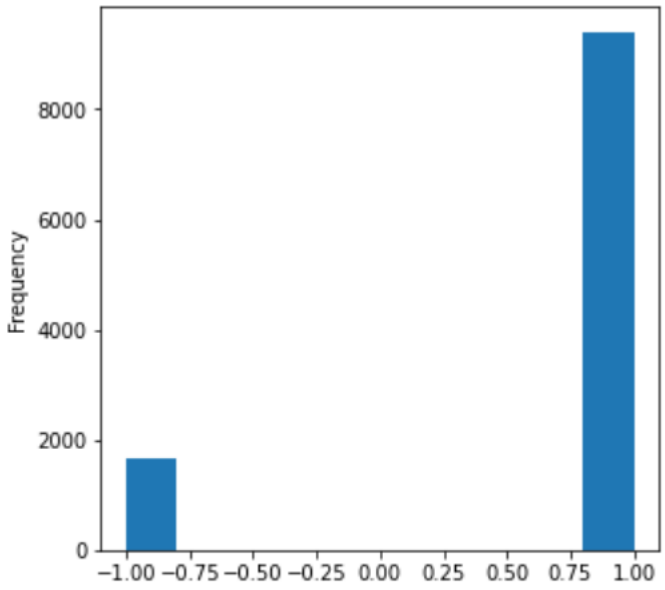

Figure 3.25: Frequency values of URL's having "@" Symbol.

- Redirecting using "//": Passing place of "//" is "phishing" if greater than 7, otherwise "legitimate."

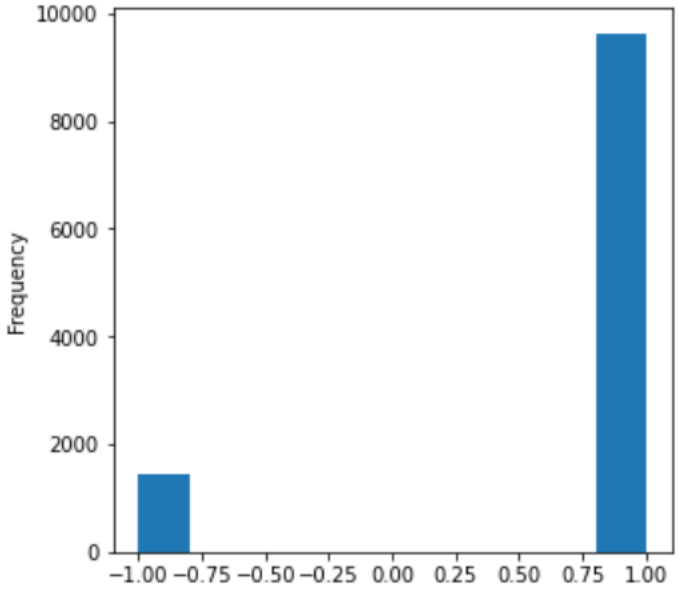

Figure 3.26: Frequency values of Redirecting using "//"

- Adding Prefix or Suffix Separated by (-) to the Domain: It is "phishing" if the domain contains "-" in the URL, and "legitimate" if it does not.

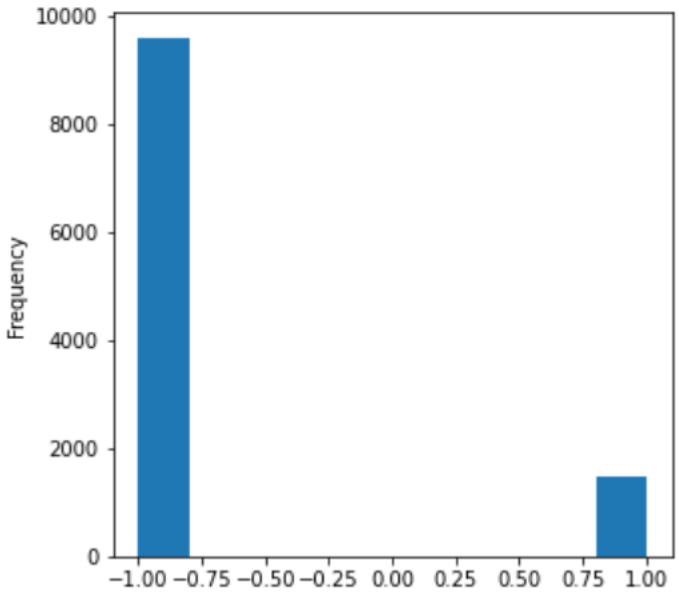

Figure 3.27: Frequency values of Adding Prefix or Suffix Separated by (-) to the Domain.

- Sub Domain and Multi Sub Domains: We omit the (www.) from the URL which is in fact a sub domain on itself and the ccTLD if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as “Suspicious”, if the dots are greater than two, it is classified as “Phishing”, otherwise if the URL has no sub domains, it is classified as “Legitimate”.

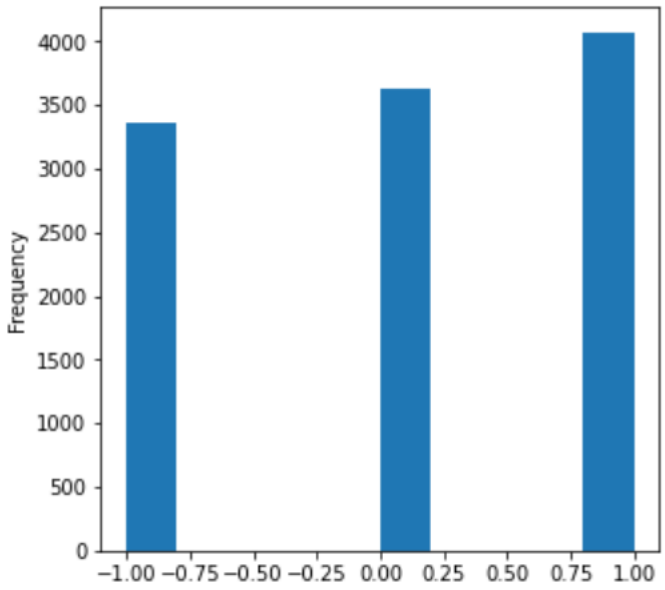

Figure 3.28: Frequency values of Sub Domain and Multi Sub Domains

- HTTPS: If the webpage is using HTTPS and the issuer of the certificate is trusted and the age of the certificate is greater than 1 years then it is classified as “Legitimate”, if the webpage is using HTTPS but the issuer of the certificate is not trusted then it is classified as “Suspicious”, otherwise “Phishing”.

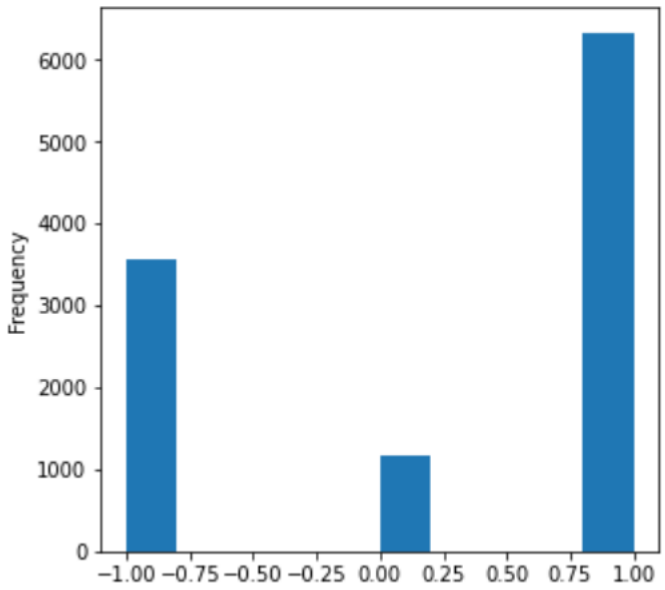

Figure 3.29: Frequency values of HTTPS

- Domain Registration Length: The domain time is “legitimate” if it is more than 1 year, otherwise it is “phishing.”

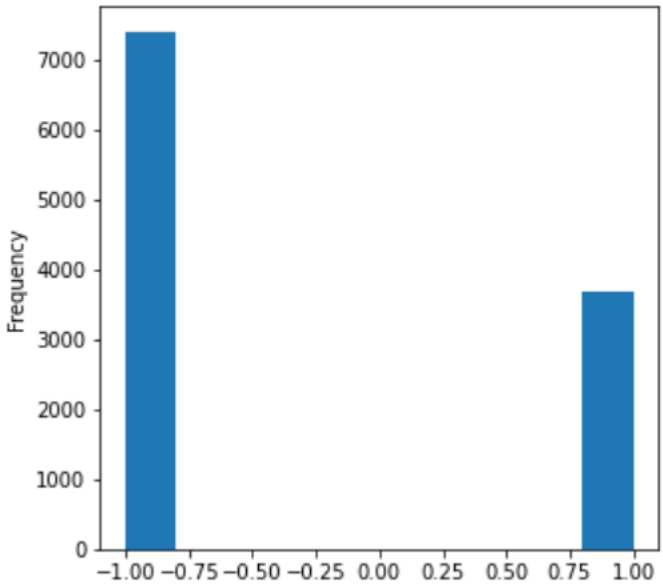

Figure 3.30: Frequency values of Domain Registration Length

- Favicon: If the webpage loads the favicon in the address bar, it is “legitimate”, if not, it is “phishing.”

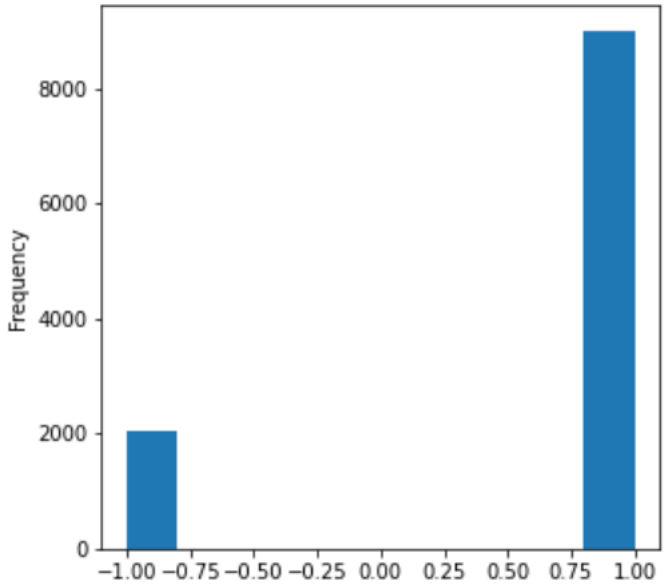

Figure 3.31: Frequency values of Favicon

- Using Non-Standard Port: “legitimate” if the port number is not in the preferred state, if it is in the preferred state, “phishing.”

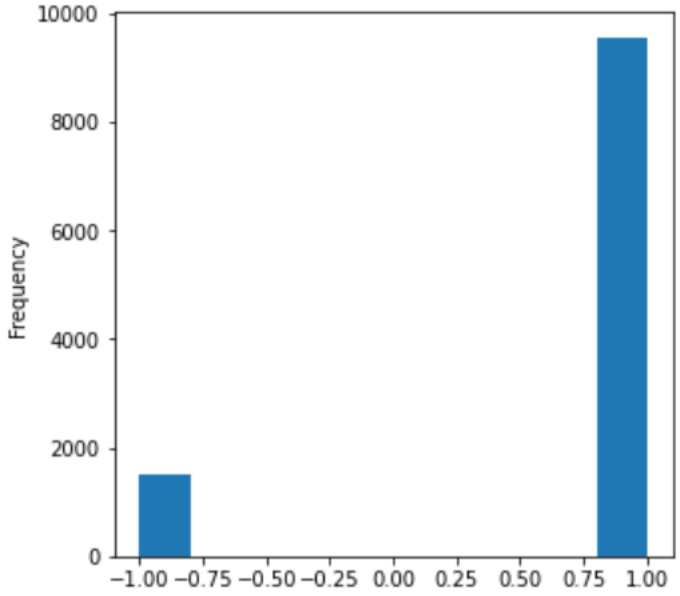

Figure 3.32: Frequency values of Using Non-Standard Port

- The Existence of “HTTPS” Token in the Domain Part of the URL: If the domain part contains “HTTPS” token then it is classified as “Phishing”, otherwise “Legitimate”.

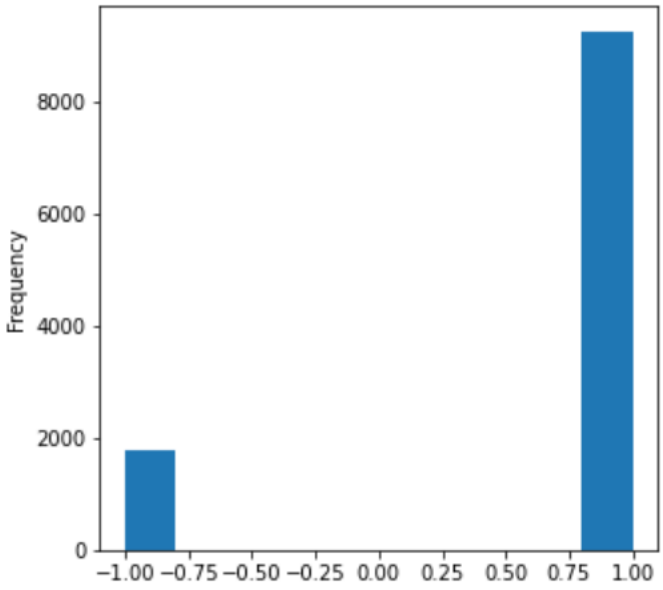

Figure 3.33: Frequency values of The Existence of “HTTPS” Token in the Domain Part of the URL

Table 3.9: What the values of the Address Bar based features mean and under what conditions they are determined.

Feature	Condition	Value	Meaning
Using the IP Address	If the Domain Part has an IP Address	-1	Phishing
	Otherwise	1	Legitimate
Long URL to Hide the Suspicious Part	URL length < 54	1	Legitimate
	54 ≤ URL length ≤ 75	0	Suspicious
	Otherwise	-1	Phishing
Using URL Shortening Services “TinyURL”	TinyURL	-1	Phishing
	Otherwise	1	Legitimate
URL’s having “@” Symbol	Url Having @ Symbol	-1	Phishing
	Otherwise	1	Legitimate
Redirecting using “//”	The Position of the Last Occurrence of “//” in the URL > 7	-1	Phishing
	Otherwise	1	Legitimate
Adding Prefix or Suffix Separated by (-) to the Domain	Domain Name Part Includes (-) Symbol	-1	Phishing
	Otherwise	1	Legitimate
Sub Domain and Multi Sub Domains	Dots in Domain Part=1	1	Legitimate
	Dots in Domain Part=2	0	Suspicious
	Otherwise	-1	Phishing
HTTPS	Use HTTPS and Issuer Is Trusted and Age of Certificate ≥ 1 Years	1	Legitimate
	Using HTTPS and Issuer Is Not Trusted	0	Suspicious
	Otherwise	-1	Phishing

Table 3.9: What the values of the Address Bar based features mean and under what conditions they are determined (continue).

Feature	Condition	Value	Meaning
Domain Registration Length	Domains Expires on ≤ 1 years	-1	Phishing
	Otherwise	1	Legitimate
Favicon	Favicon Loaded from External Domain	-1	Phishing
	Otherwise	1	Legitimate
Using Non-Standard Port	Port # is of the Preferred Status	-1	Phishing
	Otherwise	1	Legitimate
The Existence of "HTTPS" Token in the Domain Part of the URL	Using HTTP Token in Domain Part of the URL	-1	Phishing
	Otherwise	1	Legitimate

B. Abnormal Based Features

- Request URL: If the webpage loads less than 22% of the externel objects from different domain then it is classified as "Legitimate", if it loads more than 22% but less than 61% of the externel objects from different domain then it is classified as "Suspicious", otherwise "Phishing".

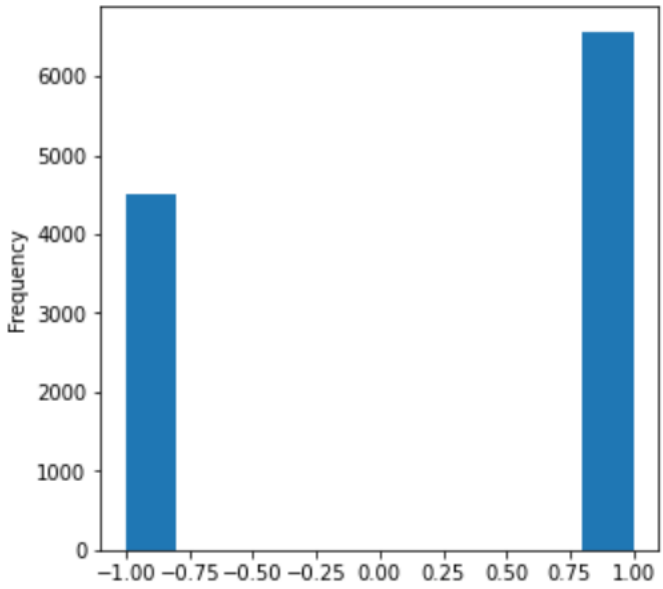

Figure 3.34: Frequency values of Request URL

- URL of Anchor: If the websites and the <a> tags in the HTML file contains different domain names more than 67% then it is classified as “Phishing”, if less than 67% but more than 31% then it is classified as “Suspicious”, otherwise “Legitimate”.

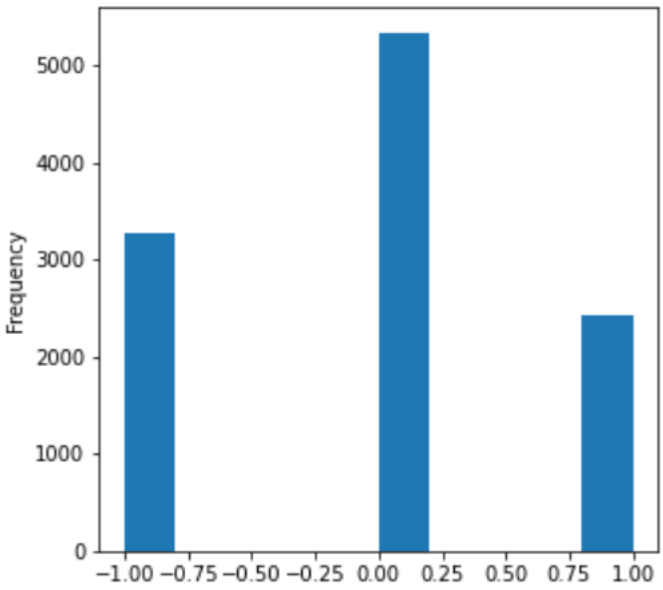

Figure 3.35: Frequency values of URL of Anchor

- Links in <Meta>, <Script> and <Link> tags: If less than 17% of links in <Meta>, <Script> and <Link> tags contain different domain names than it is classified as “Legitimate”, if more than 17% but less than 81% then it is “Suspicious”, otherwise “Phishing”.

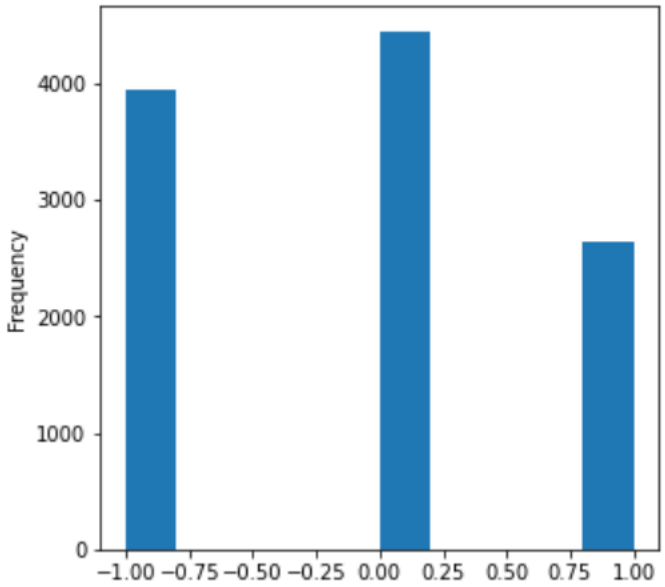

Figure 3.36: Frequency values of Links in <Meta>, <Script> and <Link> tags

- Server Form Handler (SFH): If the SFH of the web page is “about: empty” or empty, it is “phishing”, “suspicious” if attribution to a different domain, and “legitimate” otherwise.

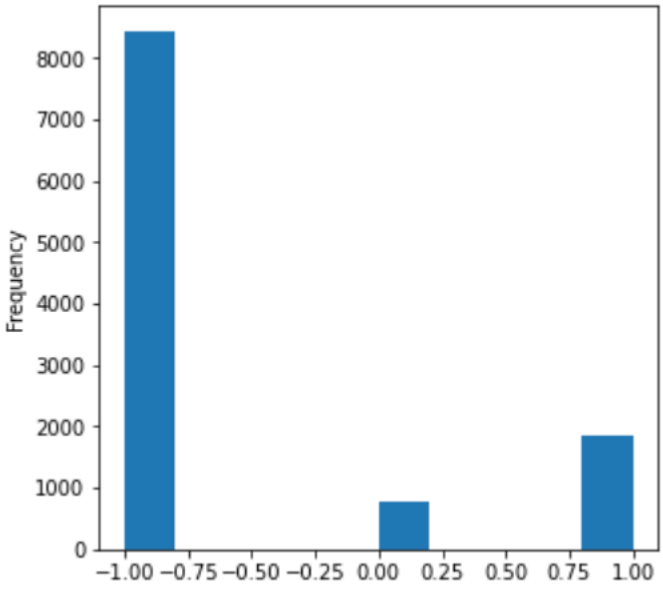

Figure 3.37: Frequency values of SFH

- Submitting Information to Email: The web page using “mail()” or “mailto” is “legitimate” if it does not send user information, and “phishing” if it does.

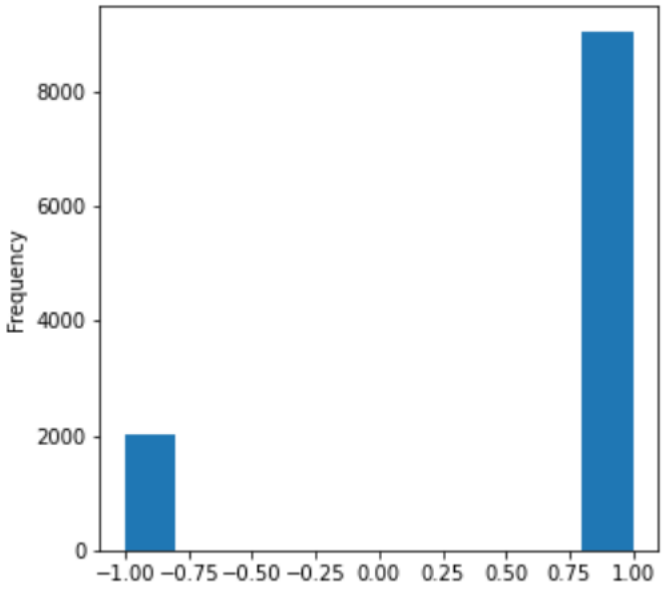

Figure 3.38: Frequency values of Submitting Information to Email

- Abnormal URL: If the URL contains the hostname it is “legitimate”, if URL does not contains it is “phishing.”

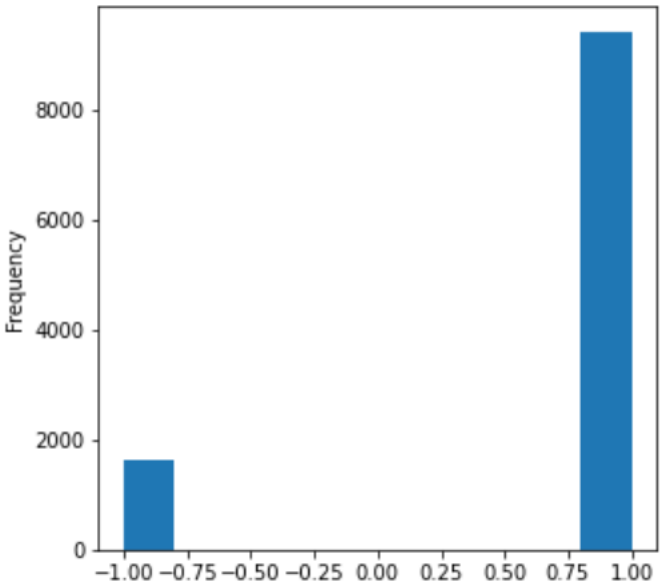

Figure 3.39: Frequency values of Abnormal URL

Table 3.10: What the values of the Abnormal based features mean and under what conditions they are determined.

Feature	Condition	Value	Meaning
Request URL	% of Request URL < 22%	1	Legitimate
	22% ≤ % of Request URL ≤ 61%	0	Suspicious
	Otherwise	-1	Phishing
URL of Anchor	% of URL of Anchor < 31%	1	Legitimate
	31% ≤ % of URL of Anchor ≤ 67%	0	Suspicious
	Otherwise	-1	Phishing
Links in <Meta>, <Script> and <Link> tags	% of Links < 17%	1	Legitimate
	17% ≤ % of Links ≤ 81%	0	Suspicious
	Otherwise	-1	Phishing
SFH	SFH is "about: blank" Or Is Empty	-1	Phishing
	SFH Refers to A Different Domain	0	Suspicious
	Otherwise	1	Legitimate
Submitting Information to Email	Using "mail()" or "mailto:" Function to Submit User Information	-1	Phishing
	Otherwise	1	Legitimate
Abnormal URL	The Host Name Is Not Included In URL	-1	Phishing
	Otherwise	1	Legitimate

C. HTML and JavaScript Based Features

- Website Forwarding: If the number of time that a website has been redirected is greater than 4 then it is classified as "Phishing", if less than 4 but greater than 2 then it is classified as "Suspicious", otherwise "Legitimate".

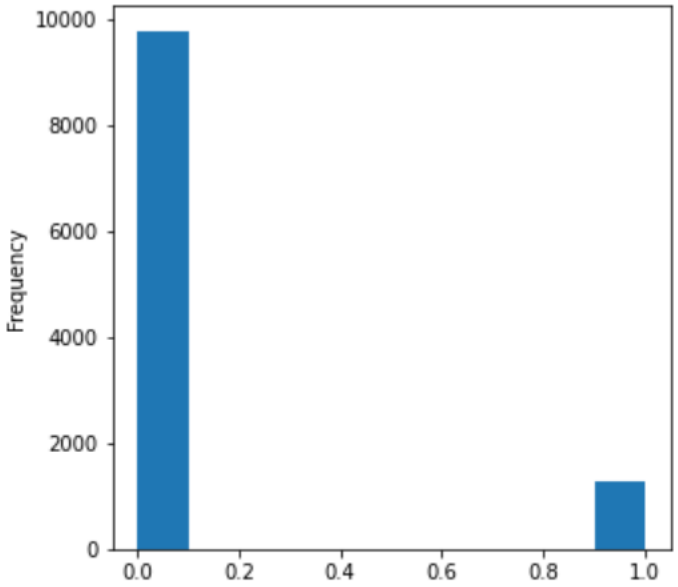

Figure 3.40: Frequency values of Website Forwarding.

- Status Bar Customization: If "onMouseOver" does not change the status bar it is "legitimate", if it does it is "phishing".

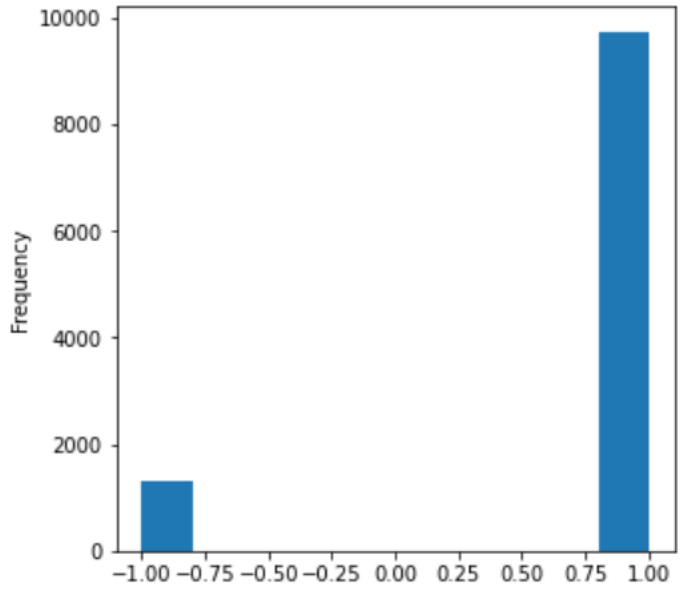

Figure 3.41: Frequency values of Status Bar Customization

- Disabling Right Click: If the right-click is active, it is "legitimate", otherwise, it is "phishing."

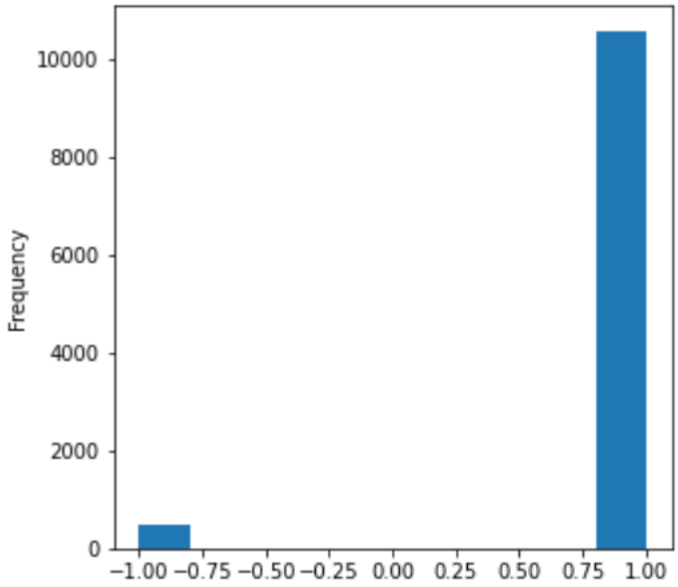

Figure 3.42: Frequency values of Disabling Right Click

- Using Pop-up Window: If the popup does not contain a text field it is “legitimate”, if it contains a text field it is “phishing.”

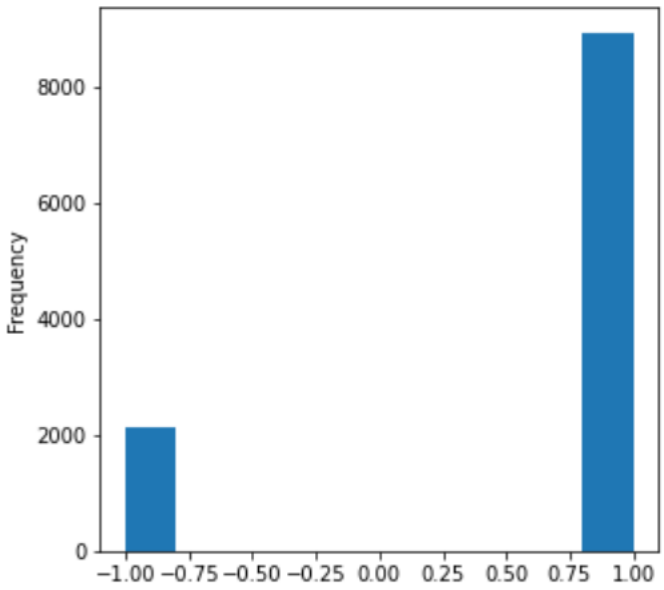

Figure 3.43: Frequency values of Using Pop-up Window

- IFrame Redirection: If the webpage is using iframe then it is classified as “Phishing”, otherwise “Legitimate”.

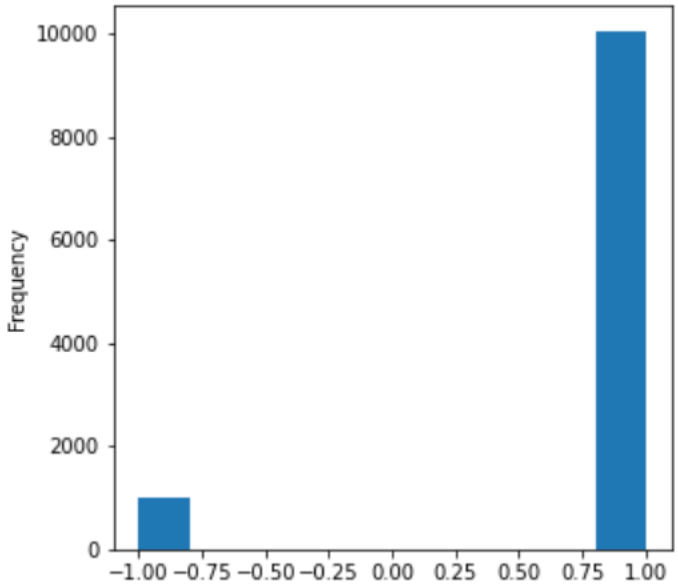

Figure 3.44: Frequency values of IFrame Redirection

Table 3.11: What the values of the HTML and JavaScript based features mean and under what conditions they are determined.

Feature	Condition	Value	Meaning
Website Forwarding	of Redirect Page ≤ 1	1	Legitimate
	2 ≤ of Redirect Page < 4	0	Suspicious
	Otherwise	-1	Phishing
Status Bar Customization	onMouseOver Changes Status Bar	-1	Phishing
	Otherwise	1	Legitimate
Disabling Right Click	Right Click Disabled	-1	Phishing
	Otherwise	1	Legitimate
Using Pop-up Window	Pop up Window Contains Text Fields	-1	Phishing
	Otherwise	1	Legitimate
IFrame Redirection	Using iframe	-1	Phishing
	Otherwise	1	Legitimate

D. Domain Based Features

- Age of Domain: If the domain age is less than six months, it is “legitimate”, otherwise it is “phishing.”

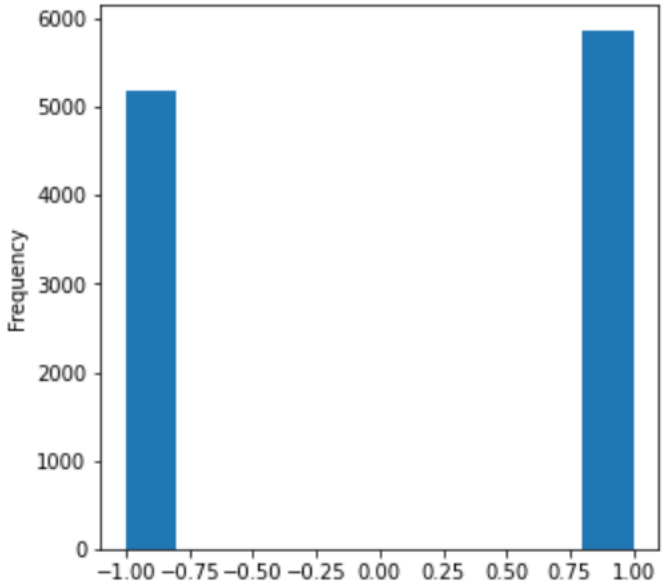

Figure 3.45: Frequency values of Age of Domain

- DNS Record: DNS record is full or if any "legitimate", else "phishing."

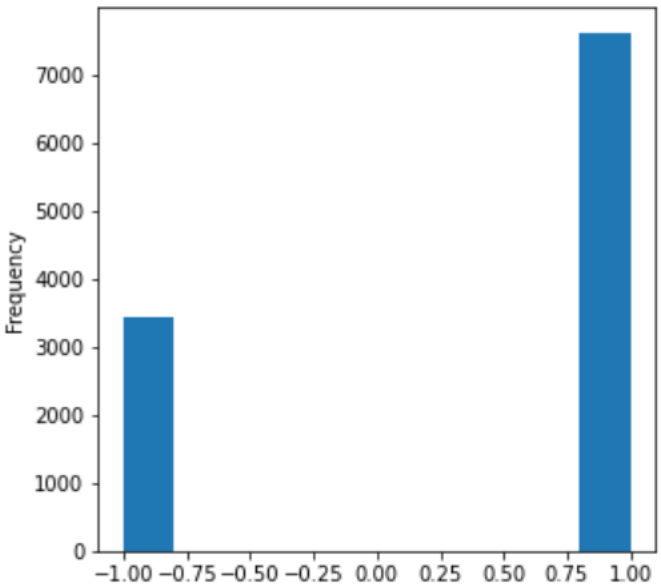

Figure 3.46: Frequency values of DNS Record

- Website Traffic: It is classed as “Phishing” if the domain has no traffic or is not recognized by the Alexa database; it is categorized as “Suspicious” if the website rank is greater than 100.000; otherwise, it is labeled as “Legitimate.”

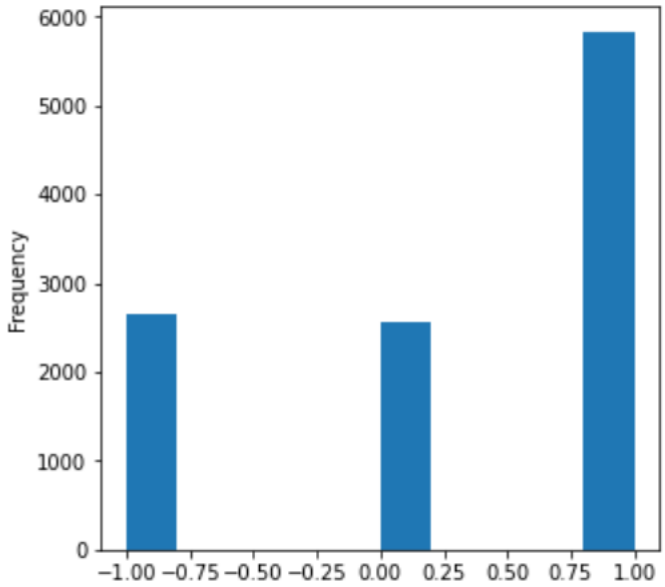

Figure 3.47: Frequency values of Website Traffic

- PageRank: If the webpage's PageRank is less than 0.2, it is considered “Phishing,” else “Legitimate.”

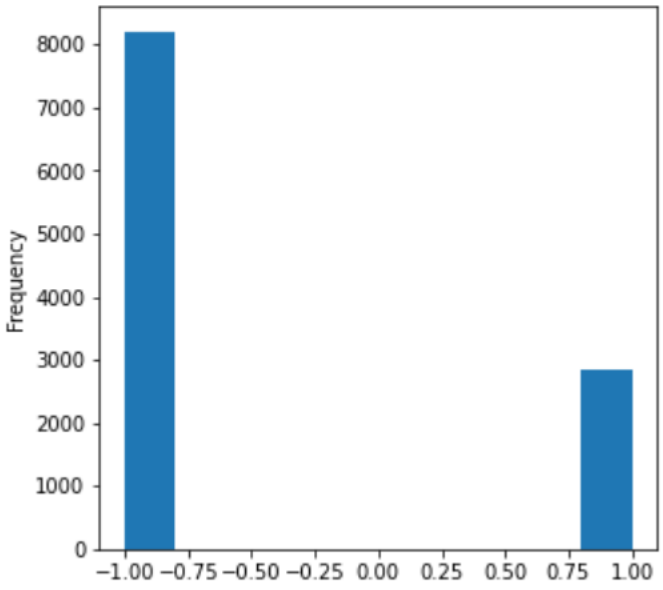

Figure 3.48: Frequency values of PageRank

- Google Index: If Google indexes the webpage, it is regarded as “Legitimate,” else it is labeled as “Phishing.”

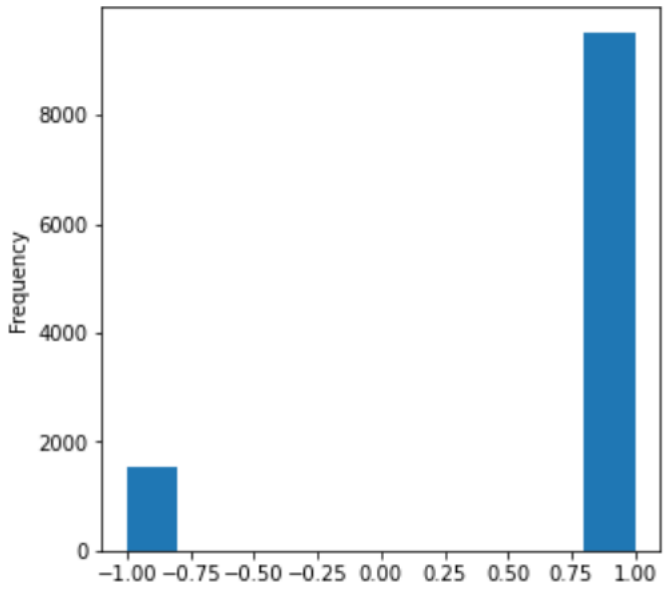

Figure 3.49: Frequency values of Google Index

- Number of Links Pointing to Page: If there are no links going to the webpage, it is categorised as "Legitimate," if there are more than 0 but less than 2, it is

categorized as "Suspicious," and if there are more than 0 but less than 2, it is categorized as "Phishing."

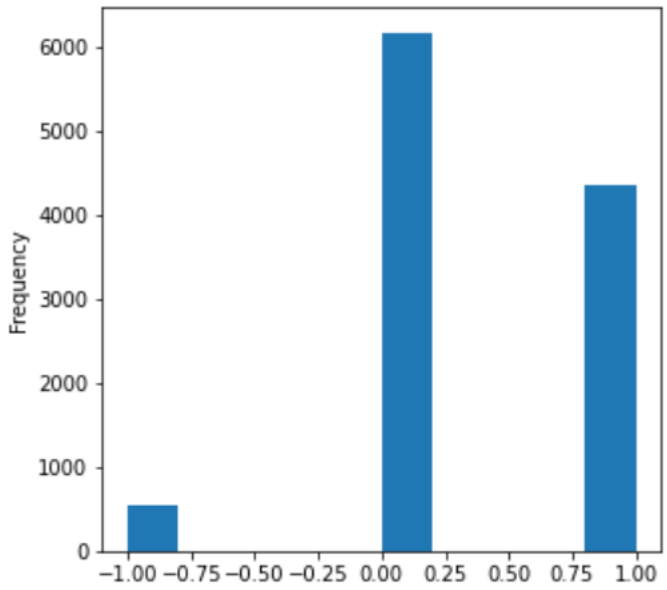

Figure 3.50: Frequency values of Number of Links Pointing to Page

- Statistical-Reports Based Feature: If the webpage's host is one of the top phishing IPs or domains, it is classed as “Phishing,” else “Legitimate.”

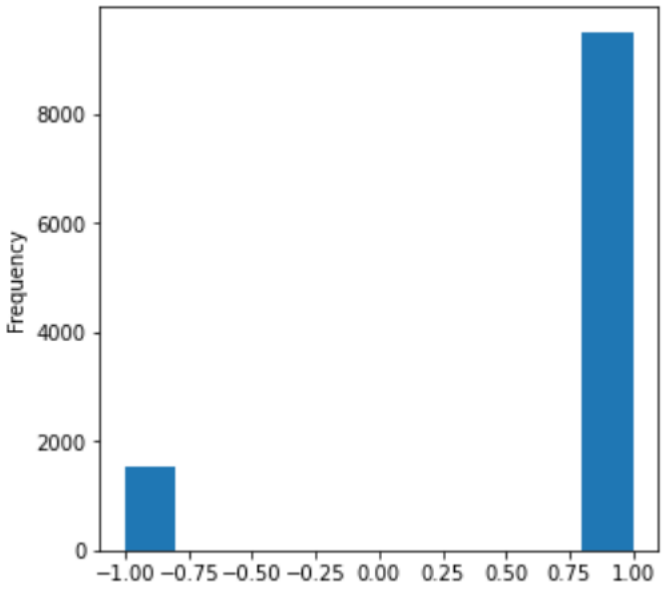

Figure 3.51: Frequency values of Statistical-Reports Based Feature

Table 3.12: What the values of the Domain based features mean and under what conditions they are determined.

Feature	Condition	Value	Meaning
Age of Domain	Age of Domain ≥ 6 months	1	Legitimate
	Otherwise	-1	Phishing
DNS Record	No DNS Record for The Domain	-1	Phishing
	Otherwise	1	Legitimate
Website Traffic	Website Rank < 100,000	1	Legitimate
	Website Rank > 100,000	0	Suspicious
	Otherwise	-1	Phishing
PageRank	PageRank < 0.2	-1	Phishing
	Otherwise	1	Legitimate
Google Index	Webpage Indexed by Google	1	Legitimate
	Otherwise	-1	Phishing
Number of Links Pointing to Page	Of Link Pointing to The Webpage = 0	-1	Phishing
	0 < Of Link Pointing to The Webpage ≤ 2	0	Suspicious
	Otherwise	1	Legitimate
Statistical-Reports Based Feature	Host Belongs to Top Phishing IPs or Top Phishing Domains	-1	Phishing
	Otherwise	1	Legitimate

3.2. Tools and Technology

We used the Python programming language on the Jupyter Notebook. Our Python version was 3.8.5. Jupyter Notebook is a web-based platform that references Julia, Python, and R programming languages. In our project, we used Sklearn, Tensorflow, Keras, Numpy, Pandas, and Matplotlib libraries used in the Python programming language. The software library that performs machine learning functions is called Sklearn. The software library used for differentiable programming and data flow is called Tensorflow. The software library that performs DL functions is called Keras. The software library that performs mathematical operations is called Numpy. Pandas is a

software library that performs data analysis functions. Matplotlib is a plotting library that visualizes data. We have used all of these programs on an HP brand laptop with 8 GB RAM, Nvidia GeForce 940MX graphics card, Intel i5 7th generation processor, and Windows 10 operating system installed.

3.2.1. ML Techniques

A. K-Nearest Neighbors (KNN)

Searching for points closest to the new point is called KNN. Representing the closest neighbors of the unknown point is called K. We also choose a K value to predict the results. The nonparametric method used for classification and regression is called KNN. The class membership in the classification is the output. The property value of the object in the regression is the output. KNN is the simplest of all ML techniques. In the KNN algorithm, we first select a K value and then calculate its distance. According to the calculated distance, K's nearest neighbor is taken. By counting the number of data points, recent data point is assigned to the category with the maximum number of neighbors. As a result of these steps, the model is ready. Eq. (1) is euclidean formula, eq. (2) is Manhattan formula, eq. (3) is minkowski formula, eq. (4) is mahalanobis formula, and eq. (5) is Chebyshev formula.

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{1}$$

$$\sum_{i=1}^{k}|x_i - y_i| \tag{2}$$

$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{1/p} \tag{3}$$

$$D(x) = \sqrt{(x - \mu)^T S^{-1}(x - \mu)} \tag{4}$$

$$\max\left(|x_2 - x_1|, |y_2 - y_1|\right) \tag{5}$$

Using these distance formulas, we choose K's nearest neighbor.

B. Random Forest

The community of many decision trees is called RF. Each tree gives a prediction in Random Forest, and we use the class with the most votes as our model. The basic concept in the Random Forest is strength out of unity. Eq. (6) is the norm formula, and eq. (7) is Random Forest classification formula. X is the set of all features. Y is the set of all trees. Also, T is the total number of trees.

$$norm\varphi_i = \frac{\varphi_i}{\sum_{j \epsilon X} \varphi_j} \tag{6}$$

$$RF\varphi_i = \frac{\sum_{j \epsilon Y} norm\varphi_{ij}}{T} \tag{7}$$

C. Support Vector Machine (SVM)

The ML algorithm preferred to distinguish between two data classes is called SVM. Hyper planes are determined for this. Eq. (8) is a hyperplane formula. Eq. (9), eq. (10), and eq. (11) are the steps of the SVM classification formula. x and y_i are two classes that do the classification process [13]. w is a vector. n is the number of training dataset of points of the form. Also, b is a scalar quantity.

$$w^T x - b \tag{8}$$

$$w^T x - b \geq 1, if\ y_i = 1 \tag{9}$$

$$w^T x - b \leq -1, if\ y_i = -1 \tag{10}$$

$$\left[\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(w^T x - b)\right)\right] + \lambda \|w\|^2 \tag{11}$$

Advantages:

- It is effective in higher dimensional spaces.
- Kernels are used for decision functions.

- It is effective if number of size is greater than number of samples.
- Uses memory efficiently.

SVM is divided into linear and nonlinear.

In Linear SVM, when data belonging to two classes are distributed linearly, they are separated from each other by a decision function obtained by using the training data of these classes. The aim is to determine the best decision line. Also, usually (-1, +1) class tags are used.

Since nonlinear SVM cannot draw a hyperplane, kernels are used. Through the kernels, ML increases significantly in nonlinear data. RBF and polynomial are the most preferred kernels.

D. Decision Tree

The ML technique preferred for classification and regression problems is called the DT. There are two nodes in the decision tree, and they are the decision node and the leaf node. The decision node is used to make the decision based on its name. The leaf node is the output of this decision. The decision tree starts with the root node and contains the entire data set. The best attribute in the dataset is then found using ASM. The root node is divided into subsets containing the possible values for the best attributes. Finally, new decision trees are created iteratively using subsets of the data set and is repeated until leaf nodes are obtained. To build the decision tree, you need to know some formulas. If an attribute exists, the Entropy formula is applied. If two attributes exist, the Entropy formula is applied. If more than two attributes exist, the Gini formula is applied. Eq. (12) is the entropy formula used when it is the only attribute. Eq. (13) is the entropy formula used when it is the only attribute. Eq. (14) is the Gini formula, and eq. (15) is the Information Gain formula. S is the current state and p_i is the probability.

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{12}$$

$$E(T, X) = \sum_{c \in X} P(c) E(c) \tag{13}$$

$$Gini = 1 - \sum_{i=1}^{c} (p_i)^2 \tag{14}$$

$$IG(T,X) = E(T) - \sum_{j=1}^{K} E(j,X) \tag{15}$$

E. Naïve Bayes

ML algorithm based on Bayes' theorem is called NB. It is mainly preferred for text classifications containing a higher dimensional education dataset. Naïve Bayes is a fast predictive, simple, and effective method. In Naïve Bayes, the data set is first converted into a frequency table. Then the probabilities of the properties of the data are extracted. Finally, the bayes theorem is applied. Eq. (16) is the bayes theorem formula. Let Y be a class of the X instance. Eq. (17) is the Naïve Bayes classification formula.

$$P(Y|X) = \frac{P(Y)P(X|Y)}{P(X)} \tag{16}$$

$$P(Y|X) = P(Y)\prod_{i=1}^{n} P(X_i|Y)P(X) \tag{17}$$

Gaussian assumes that properties want a normal distribution. If the estimators take continuous values, they are sampled from the Gaussian distribution. Eq. (18) is the Gaussian formula. Let μ_k be the mean of the values in x associated with class C_k, and let σ_k^2 be the variance of the values in x associated with class C_k. Suppose we add some observation value v.

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \tag{18}$$

Bernoulli is used when data is distributed in multiple terms. In Bernoulli, predictor variables are independent boolean values. Eq. (19) is the Bernoulli formula. If x_i is a

boolean expressing the occurrence or absence of the i 'th term from the vocabulary, the probability of a document with class C_k is given as.

$$p(x|C_k) = \prod_{i=1}^{n} p_{k_i}^{x_i} (1 - p_{k_i})^{(1-x_i)} \tag{19}$$

3.2.2. DL Techniques

A. Recurrent Neural Networks (RNN)

RNN is an ANN using sequential data. RNN uses training data to learn. It gathers information from previous entries to influence the current entry and exit. The output of RNN depends on previous items. RNN has the same weight parameter in each layer. These weights are adjusted in back propagation and gradient descent processes to facilitate reinforcement learning. RNN uses the BPTT algorithm to determine gradients. BPTT calculates errors in each layer, thus allowing us to set and fit the parameters of the model accordingly. Eq. (20) is the activation formula, and eq. (21) is the RNN formula. Also, eq. (22) is the BPTT formula. t is the time step. a^t is activation. y^t is output. In addition, g_1 and g_2 are activation functions.

$$a^t = g_1(W_{aa}a^{t-1} + W_{ax}x^t + b_a) \tag{20}$$

$$y^t = g_2(W_{ya}a^t + b_y) \tag{21}$$

$$\frac{\partial \mathcal{L}^T}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^T}{\partial W}\Big|_t \tag{22}$$

B. Deep Neural Networks (DNN)

ANN with many layers among output and input layers is called a DNN. DNN have components as neurons, synapses, weights, biases, and functions. DNN creates models from mixed nonlinear relationships. DNN makes a map of virtual neurons. It then randomly assigns weights to the links between them. Then the weights are multiplied by the inputs and give an output between 0-1. For the loss function, the MSE formula is used.

Eq. (23) is the MSE formula, and eq. (24) is formula for dot product. Eq. (25), and eq. (26) are steps of the gradient descent formula. W is weight matrix and X is feature matrix. b is intercept.

$$MSE = \frac{1}{n}\sum_{i}\left(y_i - (mx_i + b)\right)^2 \tag{23}$$

$$f(x) = b + W^T X \tag{24}$$

$$\nabla J(W) = \left(\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \ldots, \frac{\partial J}{\partial w_n}\right) \tag{25}$$

$$W := W - \alpha \nabla J(W) \tag{26}$$

C. Convolutional Neural Networks (CNN)

The CNN is made up of numerous layers of neurons. Artificial neurons compute the weighted sum of multiple inputs. Also, artificial neurons give an activation value. Each of the CNN layers creates several activation maps. The activation map highlights the relevant property of the data. Each of the neurons multiplies and adds the input by their weight. It then runs them with the help of the activation function. The process of multiplication and addition is called convolution. Recent layer output is the subsequent layer input. The classification layer gets recent convolution layer's output as input. The classification layer determines the result with values between 0-1. Eq. (27) is the dimension formula, and eq. (28) is the convolutional formula. n_h is the size of the height, n_w is the size of the width and n_c is the number of channels.

$$\dim(image) = (n_h, n_w, n_c) \tag{27}$$

$$conv(I,K)_{x,y} = \sum_{i=1}^{n_h}\sum_{j=1}^{n_w}\sum_{k=1}^{n_c} K_{i,j,k} I_{x+i-1,y+j-1,k} \tag{28}$$

3.2.3. Comparison Methods

We used precision, accuracy, F1-score, and recall compare ML and DL techniques.

The ratio of correctly categorized samples' total number to samples' total number is called accuracy [28]. Eq. (29) is an accuracy formula.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{29}$$

The ratio of phishing's number detected by the model to phishing's total number is called precision [28]. Eq. (30) is the precision formula.

$$Precision = \frac{TP}{TP + FP} \tag{30}$$

The ratio of how accurately the model predicted is called the recall [28]. Eq. (31) is the recall formula.

$$Recall = \frac{TP}{TP + FN} \tag{31}$$

The ratio of twice the products of recall and precision to their sum is called F1-score [28]. Eq. (32) is the F1-score formula.

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{32}$$

3.3. Proposed Approach

In this paper, our goal was to compare different machine learning algorithms while detecting phishing websites. Different parameter combinations of KNN, GNB, DT, BNB, RF, SVM, DNN, RNN and CNN algorithms are tested with a dataset of phishing websites from UCI Machine Learning Repository [32]. We tried values between 15% and 35% as test size in increments of five. Precision, accuracy, F1-score, and recall were evaluated to compare DL and ML techniques. For machine learning techniques, accuracy metric showed that DT, RF, KNN and SVM performed accuracy above 98%. The F1-Score metric showed that Random Forest, Decision Tree, and KNN performed F1-score

above 98%. The precision metric showed that Random Forest, Gaussian Naïve Bayes, Decision Tree, and KNN performed precision above 98% but although Gaussian Naïve Bayes performed good precision it has one of the lowest accuracies. The recall metric showed that, Random Forest, Decision Tree, and KNN are above 98%. For deep learning algorithms, the accuracy metric showed that CNN models performed accuracy lower than 92% but DNN and RNN models performed accuracy above 95% and the highest accuracy, 97.20%, is achieved by RNN Model 2. CNN models performed f1-score below 92% but DNN and RNN models performed f1-score above 96% and the highest f1-score, 97.32%, is achieved by RNN Model 2. CNN models again performed precision and recall below 92% but DNN and RNN models performed precision above 96% and the highest precision, 97.69%, is achieved by RNN Model 2 whereas the highest recall, %97.60, is achieved by RNN Model 1. The overall results of the comparison metrics on deep learning, and machine learning models we obtained that Random Forest performs the best over the machine learning models and RNN Model 2 performs the best over deep learning models.

4. EXPERIMENTAL RESULTS

The experiments are done by using Tensorflow, Keras and Sklearn packages with Python programming language. The details about the deep learning models are shown in Table 4.1. The dataset [32] is used to train the models. 70% of the data for the train set is chosen at random, whereas 30% of the data for the test set is chosen at random. With test data, accuracy, F1 score, recall, precision, and confusion matrix techniques are preferred for comparing models and evaluating their efficiency. Experiments that are done on our data set shows that the best performed deep learning architecture is RNN, the worst performed deep learning architecture is CNN and the DNN architecture is in the middle of these two architectures. The results also showed that training our model by the features that are turned into categorical features performed really badly turning features into categorical ones decreases the accuracy.

Table 4.1: Parameters, training and test accuracy values, epoch values, and test sizes used in models.

Model	Parameters	Train Accuracy	Test Accuracy	Epoch	Test Size
RNN Model 1	Tanh/Binary Cross Entropy/Adamax	98.81%	95.99%	150	30%
RNN Model 2	Sigmoid/Binary Cross Entropy/Adamax	98.31%	97.20%	150	30%
RNN Model 3	Sigmoid/Binary Cross Entropy/SGD	96.85%	96.47%	150	30%
RNN Model 4	Sigmoid/Binary Cross Entropy/SGD	96.21%	95.90%	150	30%
DNN Model 1	Sigmoid/Binary Cross Entropy/SGD	55.84%	55.35%	150	30%
DNN Model 2	Sigmoid/Adamax/Huber	98.84%	96.53%	150	30%
CNN Model 1	Tanh/Huber/Adamax	93.19%	90.59%	150	30%
CNN Model 2	Tanh/Binary Cross Entropy/Adam	92%	89.63%	150	30%
Combined Model 1	Softplus/Binary Cross Entropy/Adamax	91.23%	90.02%	150	30%
Combined Model 2	Softplus/Binary Cross Entropy/Adamax	98.48%	96.29%	150	30%
Combined Model 3	Softplus/Binary Cross Entropy/Adamax	98.54%	96.26%	150	30%

Data set is preferred to train models for machine learning algorithms. 15%, 20%, 25%, 30% and 35% of the data is randomly selected as a test set. Support Vector Machine algorithm is tested over different parameter combinations. 25% test size and “kernel: RBF, C: 1, degree: 2, gamma: scale, DFS: ovo” is performed 98.37% accuracy as the best out of 2400 different models. Linear Support Vector Machine algorithm is tested over different parameter combinations. 15% test size and “loss: squared hinge, C: 1, multi class: ovr” is performed 93.97% accuracy as the best out of 100 different models. K-Nearest Neighbors algorithm is tested over different parameter combinations. 25% test size and “K: 5, weight: distance, algorithm: auto, metric: Manhattan” is performed 98.91% accuracy as the best out of 1440 different models. Gaussian Naïve Bayes is tested with 5 different test sizes. 35% test size performed 61.34% accuracy as the best.

Bernoulli Naïve Bayes algorithm is tested over different parameter combinations. 15% test size and “alpha: 0, binarize: 0, fit prior: false” is performed 91.86% accuracy as the best out of 810 different models. Random Forest algorithm is tested over different parameter combinations. 25% test size and “class weight: balanced subsample, max features: auto, criterion: gini, warm start: true” is performed 99.09% accuracy as the best out of 240 different models. Decision Tree algorithm is tested over different parameter combinations. 25% test size and “class weight: none, max features: none, criterion: entropy, splitter: best” is performed 98.76% accuracy as the best out of 160 different models. Experiments that are done on our data set shows that the best performed machine learning algorithm is Random Forest and the worst performed algorithm is Linear Support Vector Machine.

The best performed combinations of machine learning models is tested with 10 fold cross validation, Table 4.2, to get a more accurate accuracy on our dataset. SVM performed 94.74 average accuracy. Linear SVM performed 92.65% average accuracy. Random Forest performed 97.24 average accuracy. Decision Tree performed 96.43% average accuracy. KNN performed 96.63 average accuracy. Bernoulli Naïve Bayes performed 90.40% average accuracy. Gaussian Naïve Bayes performed 60.38% average accuracy.

Although the performance of the models was very close to each other, we obtained that Random Forest performs better than other 15 different models by experimenting the models with our data set using the metrics in Table 4.3.

Table 4.2: Comparison of accuracy values and cross-validation accuracy values according to ML algorithms

Model	Accuracy	Cross–Validation Accuracy
SVM	98.37%	94.74%
Bernoulli Naïve Bayes	91.86%	90.40%
KNN	98.91%	96.63%
Linear SVM	93.97%	92.65%
Gaussian Naïve Bayes	61.34%	60.38%
Decision Tree	98.76%	96.43%
Random Forest	99.09%	97.24%

Table 4.3: F1 score, precision, recall values of ML and DL techniques used.

Model	F1 Score	Precision	Recall
SVM	0.9852	0.9798	0.9907
Linear SVM	0.9470	0.9303	0.9644
KNN	0.9901	0.9875	0.9927
Gaussian Naïve Bayes	0.4544	0.9983	0.2941
Decision Tree	0.9888	0.9894	0.9881
Bernoulli Naïve Bayes	0.9267	0.9333	0.9202
Random Forest	0.9917	0.9895	0.9940
CNN Model 2	0.9061	0.9080	0.9041
CNN Model 1	0.9156	0.9092	0.9221
DNN Model 2	0.9688	0.9643	0.9733
DNN Model 1	0.7125	0.5535	1.0
RNN Model 4	0.9631	0.9574	0.9689
RNN Model 3	0.9681	0.9689	0.9673
RNN Model 2	0.9732	0.9769	0.9694
RNN Model 1	0.9642	0.9526	0.9760
Combined Model 1	0.9087	0.9025	0.9150
Combined Model 2	0.9661	0.9579	0.9744
Combined Model 3	0.9656	0.9635	0.9678

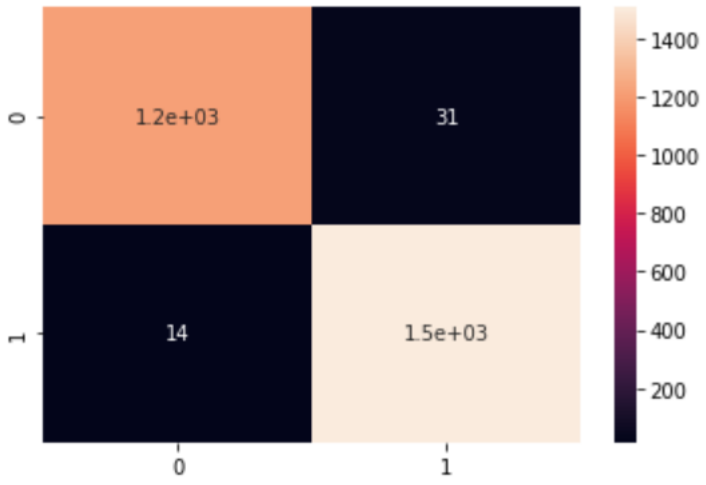

Figure 4.1: SVM's Confusion Matrix

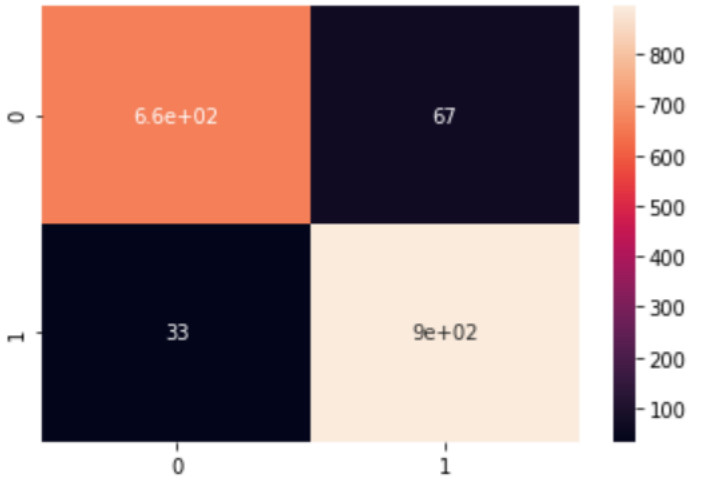

Figure 4.2: Linear SVM's Confusion Matrix

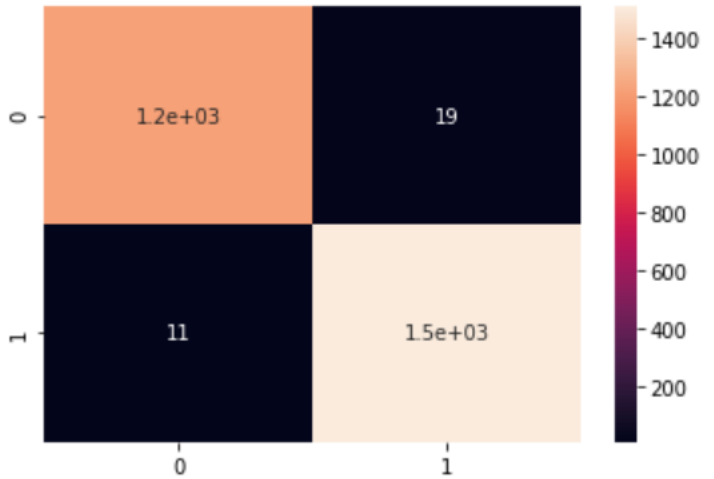

Figure 4.3: KNN's Confusion Matrix

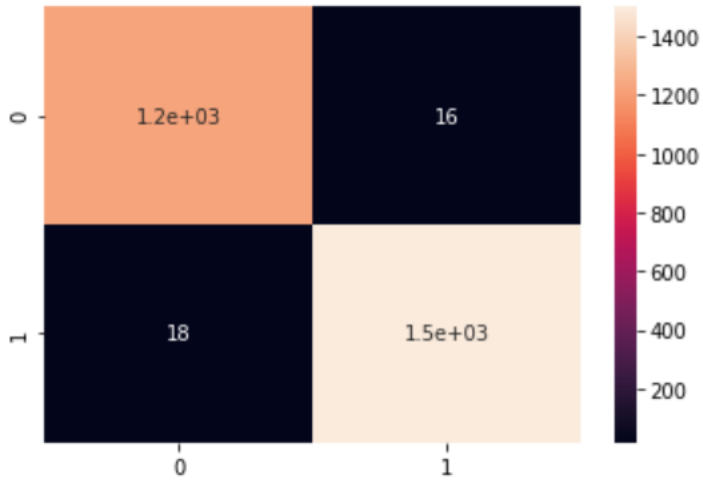

Figure 4.4: DT's Confusion Matrix

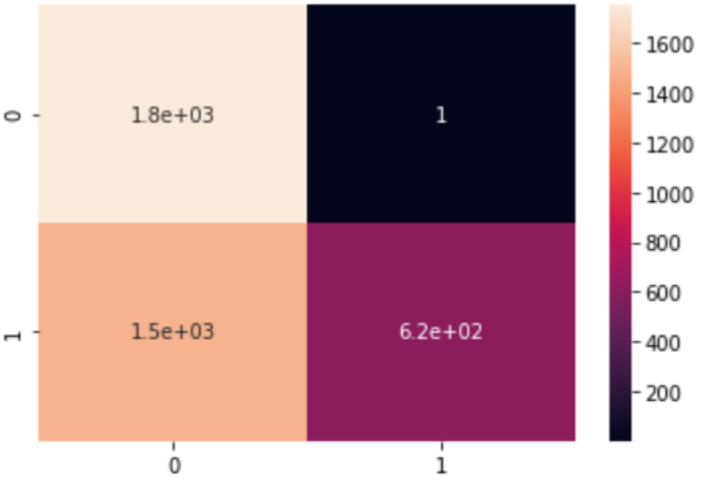

Figure 4.5: GNB's Confusion Matrix

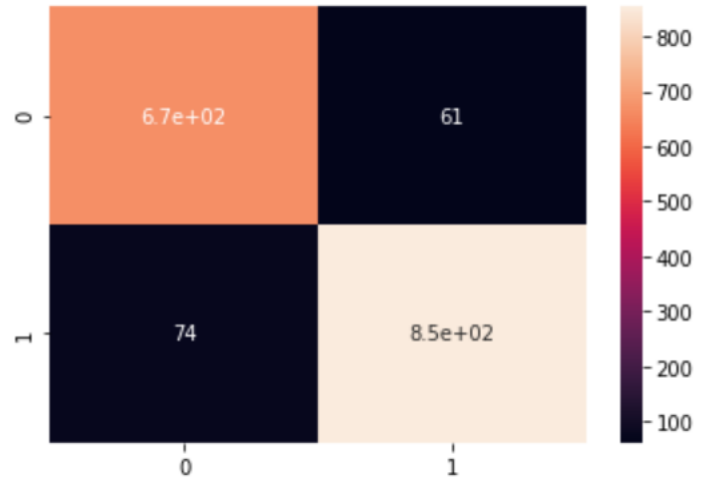

Figure 4.6: BNB's Confusion Matrix

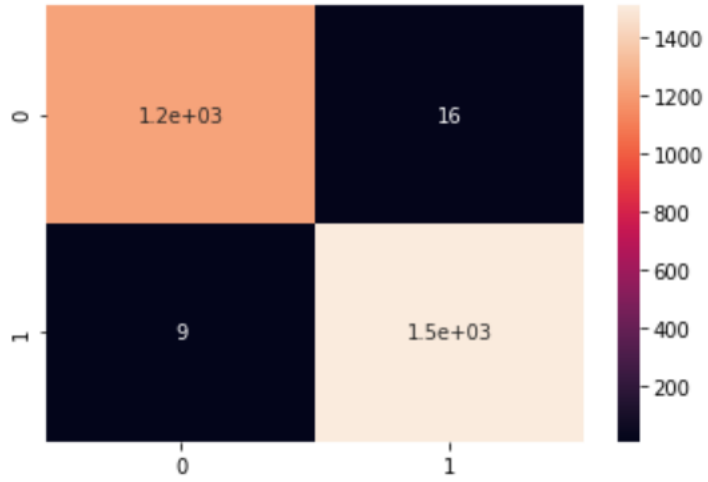

Figure 4.7: RF's Confusion Matrix

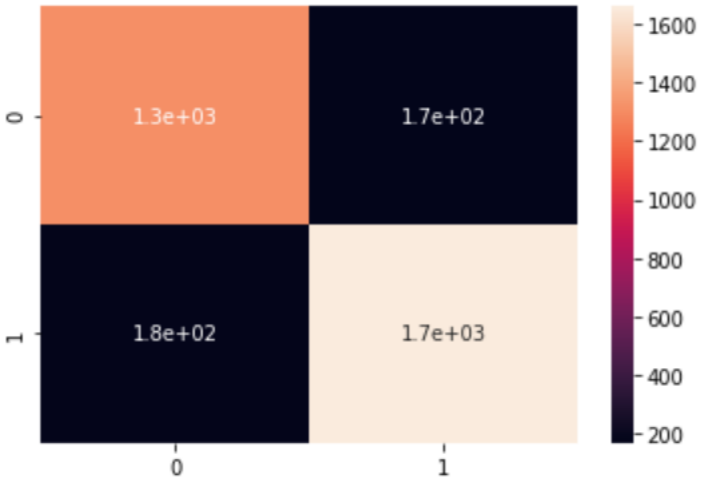

Figure 4.8: CNN Model 2's Confusion Matrix

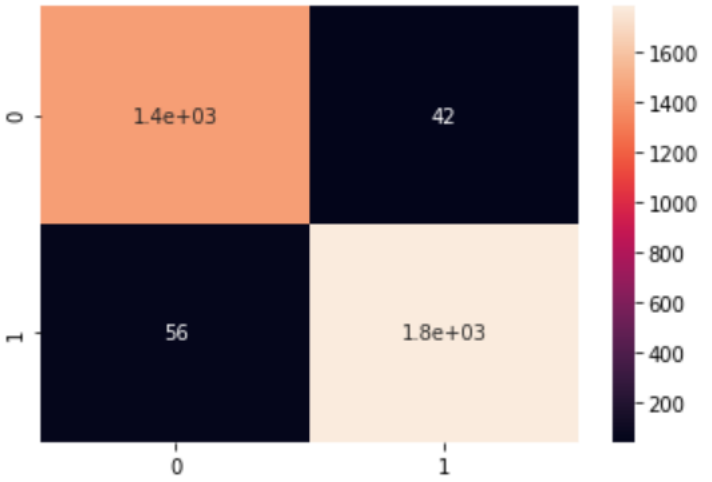

Figure 4.9: CNN Model 1's Confusion Matrix

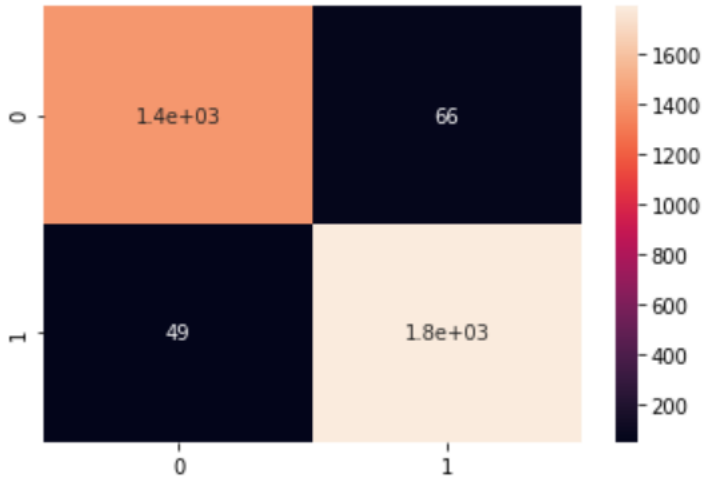

Figure 4.10: RNN Model 4's Confusion Matrix

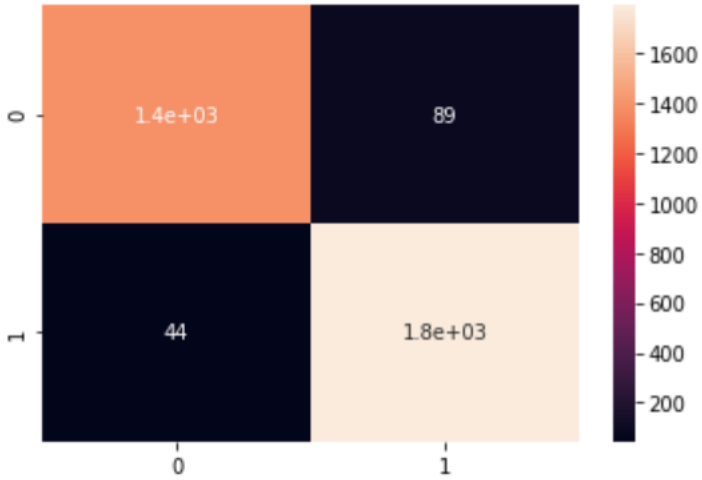

Figure 4.11: RNN Model 3's Confusion Matrix

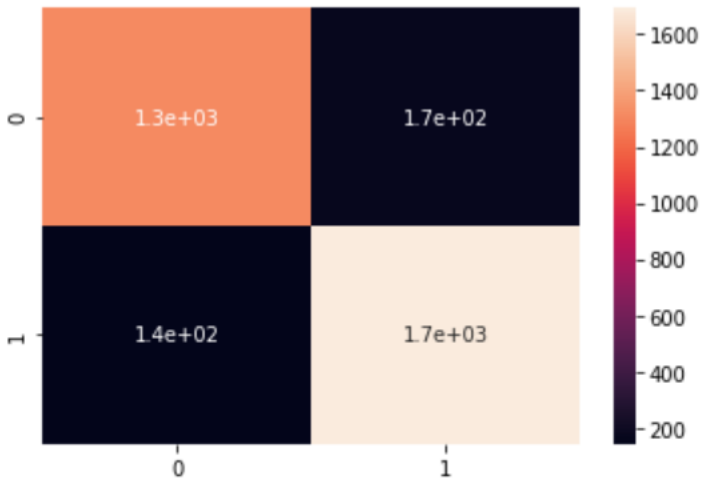

Figure 4.12: RNN Model 2's Confusion Matrix

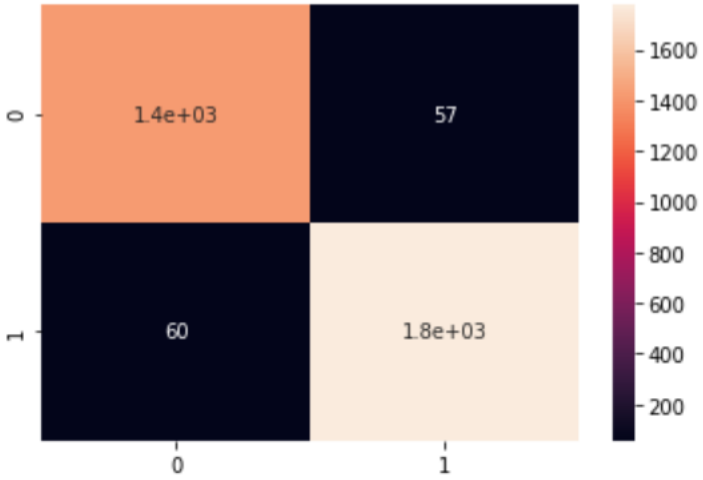

Figure 4.13: RNN Model 1's Confusion Matrix

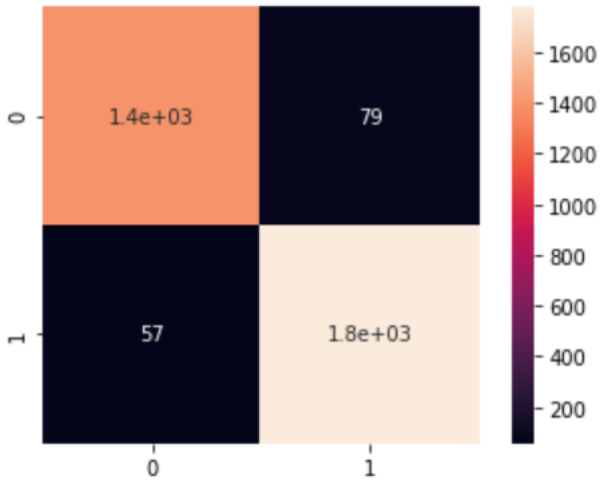

Figure 4.14: DNN Model 2's Confusion Matrix

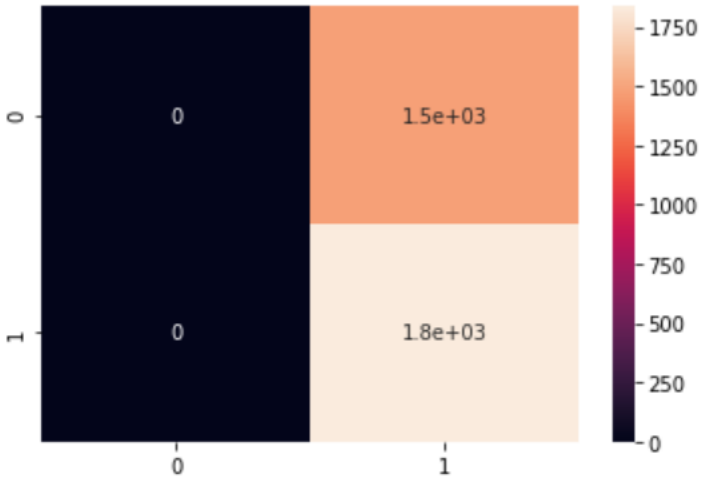

Figure 4.15: DNN Model 1's Confusion Matrix

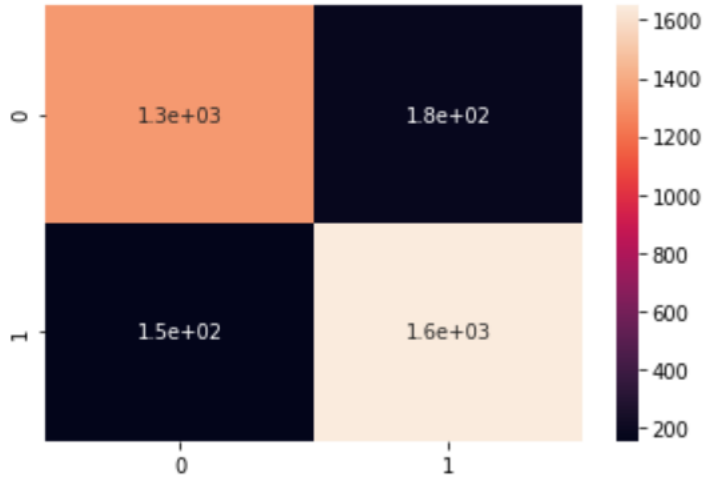

Figure 4.16: Combined Model 1's Confusion Matrix

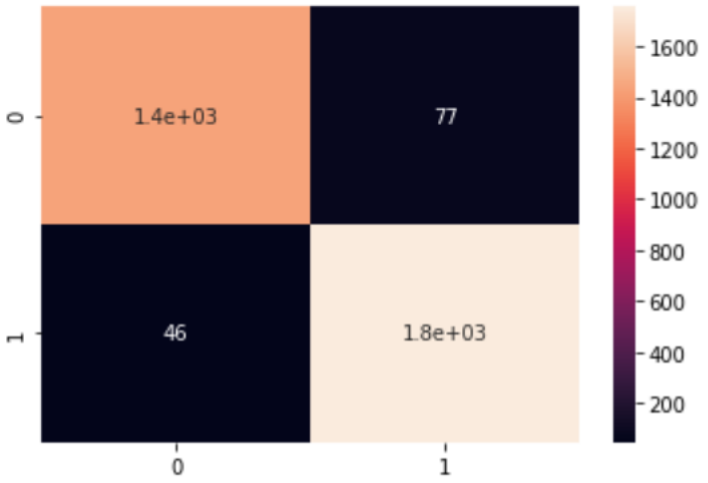

Figure 4.17: Combined Model 2's Confusion Matrix

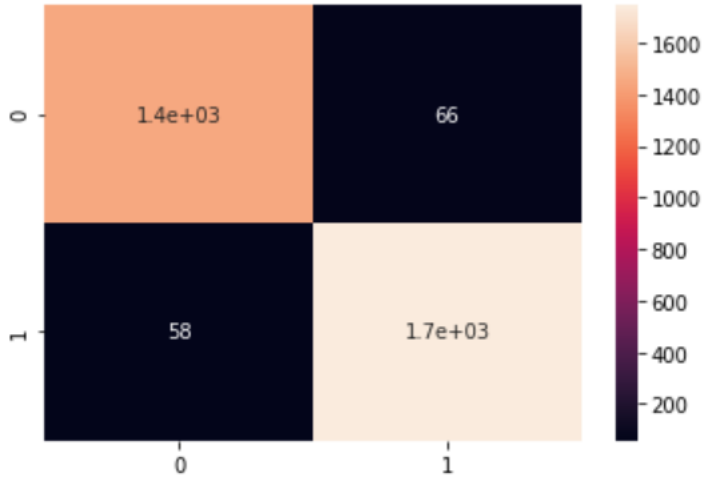

Figure 4.18: Combined Model 3's Confusion Matrix

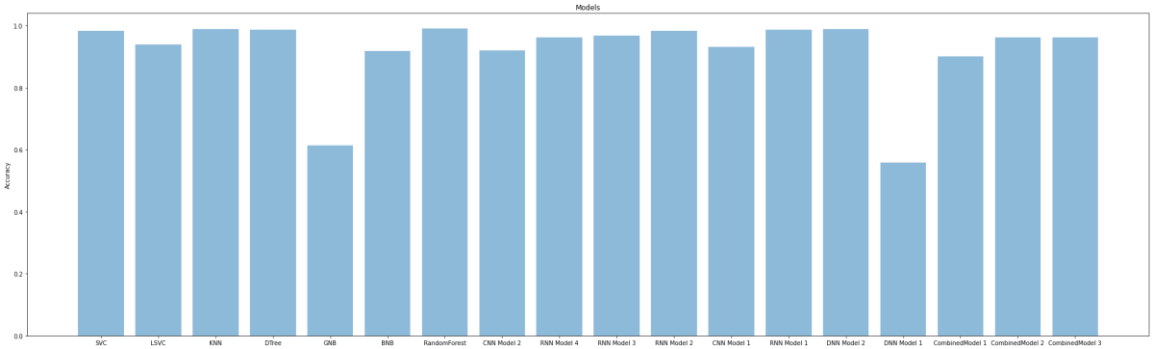

Figure 4.19: Models' best accuracy values

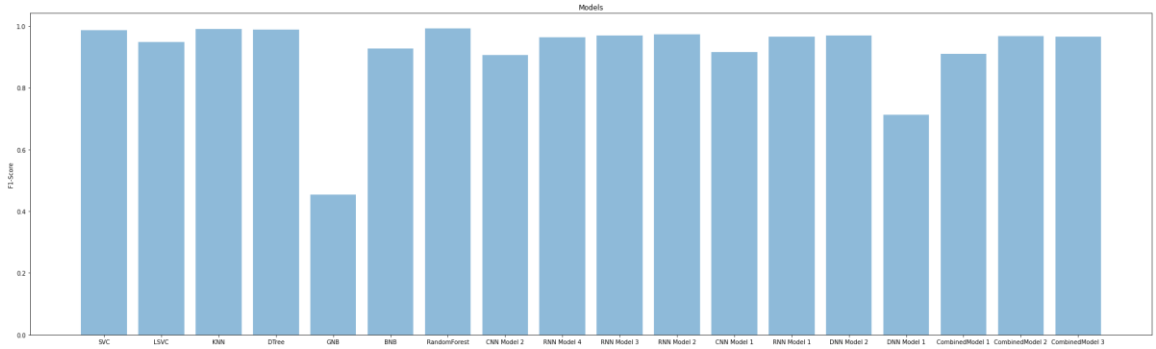

Figure 4.20: Models' best F1 score values

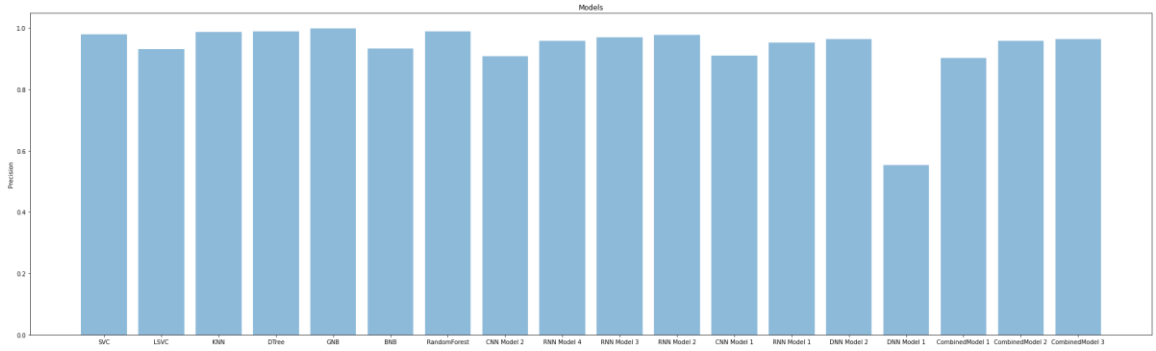

Figure 4.21: Models' best precision values

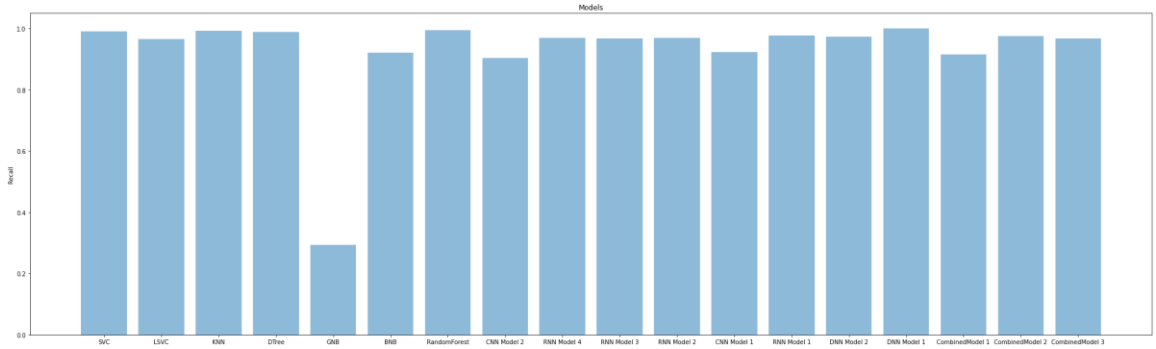

Figure 4.22: Models' best recall values

5. DISCUSSION

Which Machine Learning Algorithm performs better? and how can we compare them using model evaluation metrics? We aimed to answer these two questions.

To understand which machine learning algorithm achieves better accuracy we first set some parameters to tune every algorithm we use and get the best out of them. We also evaluated estimator performance using 10 folds cross validation technique and the results showed that the best performed models performed the best again and there was not much change in accuracy. Most of the machine learning algorithms performs accuracy above 90% but one of our findings was Naïve Bayes performs the worst performance. Also, we set five different test sizes to understand if there is a major change in accuracy and our findings were although most of the machine learning algorithms performed their best accuracy when the test size was 25%, we obtained that changing the test size does not affect our accuracy a lot. Instead of tuning the test size, tuning the hyper parameters may cause more increase in accuracy. While tuning the parameters of the deep learning models we obtained that turning our features into categorical features caused a huge decrease in our models. We have seen that training the same model with features that are turned into categorical performs accuracy below 70% but the same model achieves accuracy between 90% and 98% when training with features without turning them into categorical. So, we obtained that turning features into categorical causes a decrease in accuracy for DL models. We also obtained that RNN performs better than DNN and CNN. CNN performs the worst in deep learning models. We also obtained that when training deep learning models hyper parameter tuning affects major change in accuracy. For example, when we train the model with the FTRL optimizer model can perform accuracy below 60% accuracy but when training the model with SGD optimizer we can get accuracy above 90% easily which showed us hyper parameter tuning matters.

To compare the machine learning algorithms, we used f-measure, precision, recall, accuracy, and confusion matrix. Apart from the Naïve Bayes model and DNN model that is trained with categorical features the models perform good accuracy. But one of our findings is that even though deep learning models perform good accuracy they were not

good at other metrics like f-measure, precision, and recall. So deep learning models may be good at accuracy, but they are not good at other metrics. Random Forest, Decision Tree, KNN, and SVM performed f-measure, precision and recall above 98%. But Random Forest performed better than other models for the model evaluation metrics. We think this is because of our data set since its features are easy to build trees.

6. CONCLUSIONS

Phishing aims to deceive innocent users' credentials by serving fake web pages that impersonate their legitimate targets. There are many problems in phishing such as people, security vulnerabilities, and attackers constantly updating themselves. We propose a model that is successful against phishing attacks using ML and DL techniques. ML techniques we use; SVM, KNN, Random Forest, Gaussian Naive Bayes, Decision Tree, and Bernoulli Naive Bayes. The DL techniques we use are DNN, RNN and CNN. To compare the ML and DL algorithms, we used f-measure, precision, recall, accuracy, and confusion matrix. DL algorithms have good accuracy values, but other metrics such as precision, F1-score and recall did not give good results. For this reason, we did not recommend DL models. Nevertheless, the DL algorithm that worked the best was RNN. Although Random Forest, Decision Tree, SVM, KNN give good results in all comparison methods, we recommended it because Random Forest gave the best result. Since it is easy to create trees in the Random Forest, we concluded that the models that give the best results are in the Random Forest. Of course, we think that the reason why this is so likely to be due to the structure of our data set. As a result, the model we propose will be successful against phishing attacks. Of course, in the future, considering that the attackers are constantly updating the attacks, we will have to refresh our data set and test it to see if the model still works.

REFERENCES

[1] Wei, B., Hamad, R.A., Yang, L., He, X., Wang, H., Gao, B., & Woo, W.L. (2019, September). A Deep-Learning-Driven Light-Weight Phishing Detection Sensor (pp. 1-13). Basel, Switzerland: MDPI.

[2] Mohammed, R.M., Thabtah, F., & McCluskey, L. (2012, December). An Assessment of Features Related to Phishing Websites using an Automated Technique. In The 7th International Conference for Internet Technology and Secured Transactions (pp. 492-497). IEEE, London, UK.

[3] Vrbančič, G., Fister, I. Jr., & Podgorelec, V. (2020, October). Datasets for phishing websites detection (pp. 1-7). Maribor, Slovenia: Elsevier

[4] Adebowale, M.A., Lwin, K.T., & Hossain, M.A. (2019, August). Deep Learning with Convolutional Neural Network and Long Short-Term Memory for Phishing Detection. In 13th International Conference on SKIMA (pp. 1-8). IEEE, Island of Ukulhas, Maldives.

[5] Korkmaz, M., Sahingoz, O.K., & Diri, B. (2020, July). Detection of Phishing Websites by Using Machine Learning-Based URL Analysis. In 11th ICCCNT (pp. 1-7). IEEE, Kharagpur, India.

[6] Razaque, A., Frej, M.B.H., Sabyrov, D., Shaikhyn, A., Amsaad, F., & Oun, A. (2020, October). Detection of Phishing Websites using Machine Learning. In Cloud Summit (pp. 103-107). IEEE, Harrisburg, PA, USA.

[7] Opara, C., Wei, B., & Chen, Y. (2020, July). HTMLPhish: Enabling Phishing Web Page Detection by Applying Deep Learning Techniques on HTML Analysis. In IJCNN (pp. 1-8). IEEE, Glasgow, UK.

[8] Folorunso, S.O., Ayo, F.E., Abdullah, K-K.A., & Ogunyinka, P.I. (2020, January). Hybrid vs Ensemble Classification Models for Phishing Websites (pp. 3387-3396). Baghdad, Iraq: Iraqi Journal of Science.

[9] Mohammad, R.M., Thabtah, F. & McCluskey, L. (2014). Intelligent rule-based phishing websites classification (pp. 153-160). Elsevier.

[10] Kumar, J., Santhanavijayan, A., Janet, B., Rajendran, B., & Bindhumadhava, B.S. (2020, January). Phishing Website Classification and Detection Using Machine Learning. In ICCCI (pp. 1-6). IEEE, Coimbatore, India.

[11] Bai, W. (2020, August). Phishing Website Detection Based on Machine Learning Algorithm. In International Conference on CDS (pp. 293-298). IEEE, Stanford, CA, USA.

[12] Yang, P., Zhao, G., & Zeng, P. (2019, January). Phishing Website Detection Based on Multidimensional Features Driven by Deep Learning (pp. 15196-15209). IEEE.

[13] Anupam, S., & Kar, A.K. (2020, November). Phishing website detection using support vector machines and nature-inspired optimization algorithms (pp. 17-32). New Delhi, India: Springer.

[14] Ibrahim, D.R., & Hadi, A.H. (2017, October). Phishing Websites Prediction Using Classification Techniques. In ICTCS (pp. 133-137). IEEE, Amman, Jordan.

[15] Abdelhamid, N., Ayesh, A., & Thabtah, F. (2014). Phishing detection based Associative Classification data mining (pp. 5948-5959). Elsevier.

[16] Yang, S. (2020, June). Research on Website Phishing Detection Based on LSTM RNN. In 4th ITNEC (pp. 284-288). IEEE, Chongqing, China.

[17] Vrbančič, G., Fister, I. Jr., & Podgorelec, V. (2018, June). Swarm Intelligence Approaches for Parameter Setting of Deep Learning Neural Network: Case Study on Phishing Websites Classification. In International Conference on Web Intelligence, Mining and Semantics (pp. 1-8). ACM, New York, USA.

[18] Hasan, K.M.Z., Hasan Md. Z., & Zahan, N. (2019, July). Automated Prediction of Phishing Websites Using Deep Convolutional Neural Network. In IC4ME2 (pp. 1-4). IEEE, Rajshahi, Bangladesh.

[19] Vaitkevicius, P., & Marcinkevicius, V. (2020). Comparison of Classification Algorithms for Detection of Phishing Websites (pp. 143-160). Vilnius, Lithuania: Vilnius University.

[20] Zaman, S., Deep, S.M.U., Kawsar, Z., Ashaduzzaman, Md., & Pritom, A.I. (2019, December). Phishing Website Detection Using Effective Classifiers and Feature Selection Techniques. In 2nd ICIET (pp. 1-6). IEEE, Dhaka, Bangladesh.

[21] Mohammad, R.M., & Thabtah, F. (2013, November). Predicting phishing websites based on self-structuring neural network (pp. 443–458). London, UK: Springer.

[22] Tan, C.L. (2018, March). Phishing Legitimate Full. URL: https://data.mendeley.com/

[23] Abdelhamid, N., Thabtah, F., & Abdel-jaber, H. (2017, July). Phishing detection: A recent intelligent machine learning comparison based on models content and features. In International Conference on ISI (pp. 72-77). IEEE, Beijing, China.

[24] Subasi, A., Molah, E., Almkallawi, F., & Chaudhery, T. J. (2017, November). Intelligent phishing website detection using random forest classifier. In ICECTA (pp. 1-5). IEEE, Ras Al Khaimah, United Arab Emirates.

[25] Basit, A., Zafar, M., Liu, X., Javed, A.R., Jalil, Z., & Kifayat, K. (2021). A comprehensive survey of AI-enabled phishing attacks detection techniques (pp. 139-154). Springer.

[26] Rasymas, T., & Dovydaitis, L. (2020, September). Detection of Phishing URLs by Using Deep Learning Approach and Multiple Features Combinations (pp. 471-483). Baltic Journal of Modern Computing.

[27] Bahnsen, A.C., Bohorquez, E.C., Villegas, S., Vargas, J., & Gonzalez, F.A. (2017, April). Classifying Phishing URLs Using Recurrent Neural Networks. In APWG Symposium on Electronic Crime Research (pp. 1-8). IEEE, Scottsdale, AZ, USA.

[28] Wang, W., Zhang, F., Luo, X., & Zhang, S. (2019, October). PDRCNN: Precise Phishing Detection with Recurrent Convolutional Neural Networks (pp. 1-16). Wiley-Hindawi.

[29] Saxe, J., & Berlin, K. (2017, February). eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys (pp. 1-18). New York, USA: arXiv.

[30] Le, H., Pham, Q., Sahoo, D., & Hoi, S.C.H. (2018, March). URLNet: Learning a URL Representation with Deep Learning for Malicious URL Detection (pp. 1-13). New York, USA: arXiv.

[31] Sahingoz, O.K., Baykal, S.I., & Bulut, D. (2018). Phishing Detection from URLs by Using Neural Network (pp. 41-54). İstanbul, Turkey: IKU.

[32] Mohammad, R.M., McCluskey, L., & Thabtah, F. (2015, March). Training Dataset. URL: https://archive.ics.uci.edu/ml/index.php

[33] Abdelhamid, N. (2016, November). Phishing Data. URL: https://archive.ics.uci.edu/ml/index.php

[34] Banik, S. (2021, January). Dataset. URL: https://www.kaggle.com/

[35] Hannousse, A., & Yahiouche, S. (2020, September). dataset B 05 2020. URL: https://data.mendeley.com/

[36] Adebowale, M. (2019, December). Text Frame Image Features. URL: https://data.mendeley.com/

[37] Mamun, M.S.I., Rathore, M.A., Lashkari, A.H., Stakhanova, N., & Ghorbani, A.A. (2016). Phishing. URL: https://www.unb.ca/