Detecting Phishing Website Using Data Mining Techniques

Nasim Tavakkoli, Furkan Reha Tutaş, Melis Tuvana Sarioglu

September 27, 2021

1 Introduction

Phishing is a cyber-attack which tries to steal sensitive information of victim using misleading e-mails and websites. Phishing websites appear frequently in recent years as one of the cyber security threats, which have resulted in massive damage to online financial services and data security. Moreover, during the past year, the COVID-19 pandemic has amplified the use of technology in every zone, which resulted in switching activities such as organizing official meetings, attending classes, shopping, and payments from physical to online. Therefore, these days phishers have more opportunities to carry out attacks in order to impact the victims financially.
In a phishing attack, the attacker directs victims to fake websites and aims to steal their sensitive and personal information including passwords of accounts, bank details, atm pin-card data, etc. Hence protecting sensitive information from web phishing is difficult since the attacker utilizes obfuscated URLs, link redirections, and manipulated links in order to mislead the victims. Phishing website URLs have distinct features contrasting with legal URLs. These features can be separated to four groups: Address bar features,

Abnormal features, HTML and JavaScript features and Domain features. By way of example, in the case that a website URL contain short address, IP address or special characters it can be categorized as a phishing website. As another example, if a web page contains considerable amount of requested links on the page, hyperlinked tags, mailbox information or special HTML or Java tags it can be classified as malicious.

Detection of a phishing attack with high accuracy is a challenging research issue. The utilization of Data Mining algorithms is one of the approaches to prevent phishing attacks. Several articles have been published related to the prediction of phishing websites that employed recent data mining techniques. SMOTE(Synthetic Minority Over-sampling Technique) [1] proposes a solution to scarcity issue of actual phishing website data in comparison with reliable website data in the training data-set. SMOTE is applied to phishing website data-set in order to handle the class imbalance problem. This work applies the Support Vector Machine, Random Forests, and XGBoost algorithms on both balanced and imbalanced data-set and proves that using SMOTE on cyber security data mining results in higher accuracy. Bohacik, Skula, and Zabovsky [2] propose a plugin for web-browsers using a decision tree model named C4.5. In [3] a system is proposed to detect phishing URLs with no historical data. The system extracts the features of the website through URL while the user is visiting it. The obtained features are used as test data for the model and a cloud-based classification model is employed which is trained using the Random Forest Algorithm.

In this project, we aimed to examine different data mining classifiers that are applied to the phishing data-set in the literature. The goal of this project is to determine effective data mining algorithms in terms of classification performance.

2 General Framework

In our project, we have decided to design our project in the Python programming language since python is the most common programming language among our teammates. Python is a language that can be used for a wide variety of applications, it is a dynamic, object-oriented, high-level programming language that is very popular in business and academic life.
Since we need to load and process the data set, we have used Pandas library. Pandas is a software library that developed for the Python programming

language to be able to manipulate, process and analyze the data sets. Since we also need some transformation of data set values, we have used Numpy library as a helper tool for Pandas.
In addition to loading and processing the data set we also need data visualization procedures to be able to analyze the data set properties further. For this purpose, we have used Seaborn/Mathplotlib libraries. Seaborn is a software library developed for the Python programming language that provides various methods to draw informative statistical graphics for data sets based on Mathplotlib library.
After loading, processing, visualizing and analyzing the data set, we also need machine learning algorithms to detect phishing websites using the data set. For this reason, we have used Sklearn and XgBoost libraries. Scikit-Learn (Sklearn) is a library implemented for Python that consists of a variety of methods for machine learning and statistics. It supports a variety of classification, regression and statistical algorithms. Since XgBoost machine learning algorithms are not implemented in Sklearn, we also have decided to use XgBoost library to both create a model to detect phishing websites as well as to select features in the data set.

3 Related Work

In current project we will investigate the method proposed in OFS-NN [4]. In this paper, OFS-NN is proposed as an effective phishing websites detection model created based on a neural network that is widely used to detect phishing attacks and the optimal feature selection method. The aim was to solve the over-fitting problem which prevents effective detect phishing websites. To appraise the impact of sensitive features on phishing websites detection feature, a validity value (FVV) is introduced.

$$FVV = P(Ax = 1 \& y = 1) + P(Bx = -1 \& y = -1) \tag{1}$$

By using the FVV index, an algorithm is designed to select the optimal features from phishing websites. According to [4], for feature A with the value 1, if the sample is also predicted as label 1 by the OFS-NN, A contributes to choice of label 1. On the other hand, for feature A with the value -1, if the sample is also predicted as label -1 by the OFS-NN, A contributes to choice of label -1. As a matter of fact, only features with value 1 and features with

value -1 can contribute the decision of a classifier. In the phishing data sets,

$$P(Ax = 1 \& y = 1)$$

indicates the probability that the feature value is represented as a phishing website and the detection result by the feature A, is also the phishing one;

$$P(Bx = -1 \& y = -1)$$

indicates the probability that the feature value is a legitimate website and the detection result by the feature A, is also the legitimate. This algorithm could deal with problems of the over-fitting problem of the underlying neural network by selecting the optimal features in the data-set. The paper concludes that the OFS-NN model is an accurate and stable model for detecting several types of phishing websites.

4 Data-set

The csv and txt versions of the data-set used in the project received from Kaggle [5]. The data-set provides collection of 11000+ website URLs. Each website instance contains 30 parameters and has a class tag that identifies whether it is a phishing website or not. Website URLs tagged as 1 or -1 for phishing and reliable website respectively. There are 30 features for each URL in the data-set where each feature is assigned with a number. At most three values is assigned to each feature in the data-set, these values are -1, 0 and 1. We divided features to four categories as address bar features, abnormal features, HTML and JavaScript features and domain features.

HTML and JavaScript Features	
Feature Name	Value
WebsiteForwarding	-1 , 0 , 1
StatusBarCust	-1 , 1
DisableRightClick	-1 , 1
UsingPopupWindow	-1 , 1
IframeRedirection	-1 , 1

Address Bar Features	
Feature Name	Value
UsingIP	-1 , 1
LongURL	-1 , 0 , 1
ShortURL	-1 , 1
Symbol@	-1 , 1
Redirecting//	-1 , 1
PrefixSuffix-	-1 , 1
SubDomains	-1 , 0 , 1
HTTPS	-1 , 0 , 1
DomainRegLen	-1 , 1
Favicon	-1 , 1
NonStdPort	-1 , 1
HTTPSDomainURL	-1 , 1
Abnormal Features	
Feature Name	Value
RequestURL	-1 , 1
AnchorURL	-1 , 0 , 1
LinksInScriptTags	-1 , 0 , 1
ServerFormHandler	-1 , 0 , 1
InfoEmail	-1 , 1
AbnormalURL	-1 , 1
Domain Features	
Feature Name	Value
AgeofDomain	-1 , 1
DNSRecording	-1 , 1
WebsiteTraffic	-1 , 0 , 1
PageRank	-1 , 1
GoogleIndex	-1 , 1
LinksPointingToPage	-1 , 0 , 1
StatsReport	-1 , 1

5 Pre-Processing and Analyzing the Data Set

Before implementing machine learning algorithms to detect phishing websites, we analyzed our data set to have a further understanding of the data set.

Firstly, to check whether the data set is balanced or unbalanced, we counted the number of 1(phishing website) and -1(not phishing website) labels in the data set. The bar graph of the label counts is shown in the figure below.

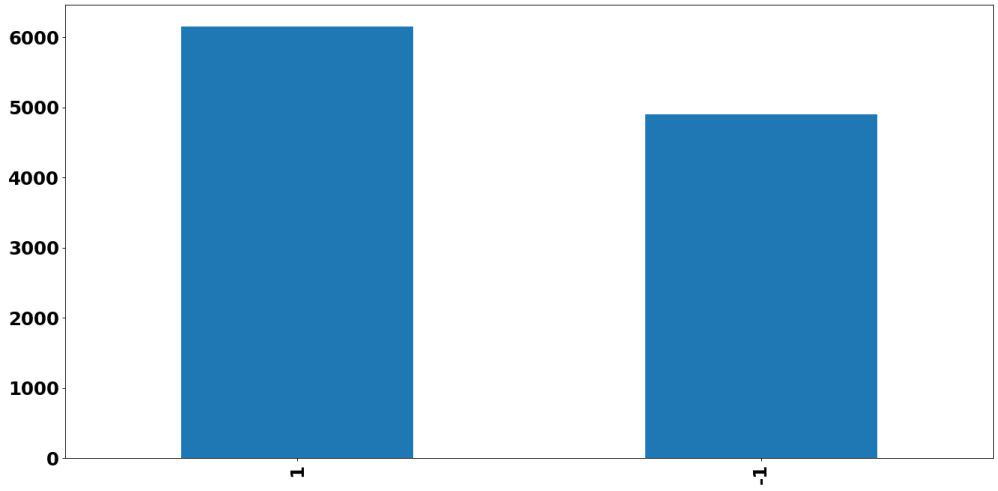

Figure 1: Label Counts-Bar Plot

As it can be observed from the figure above, the number of label 1 is around 6000 and the number of label -1 is around 5000. Although the number of labels is not equal to each other, if we look at the percentages of labels (1 is around 55 percent, whereas -1 is around 45 percent); we can claim that the data set is nearly balanced and doesn't require any additional pre-processing to make the data set more balanced. Additionally, label 1 has 55 percent ratio means that if we apply majority vote (select 1 for all test/validation data sets) we would have around 55 percent accuracy. In this project with the help of various machine learning algorithms, we want to achieve a much higher accuracy rate than the majority vote.
Secondly, in order to check which features are more important than the others for predicting the labels, we used the XgBoost classifier from the XgBoost library with default settings alongside its feature importance method.

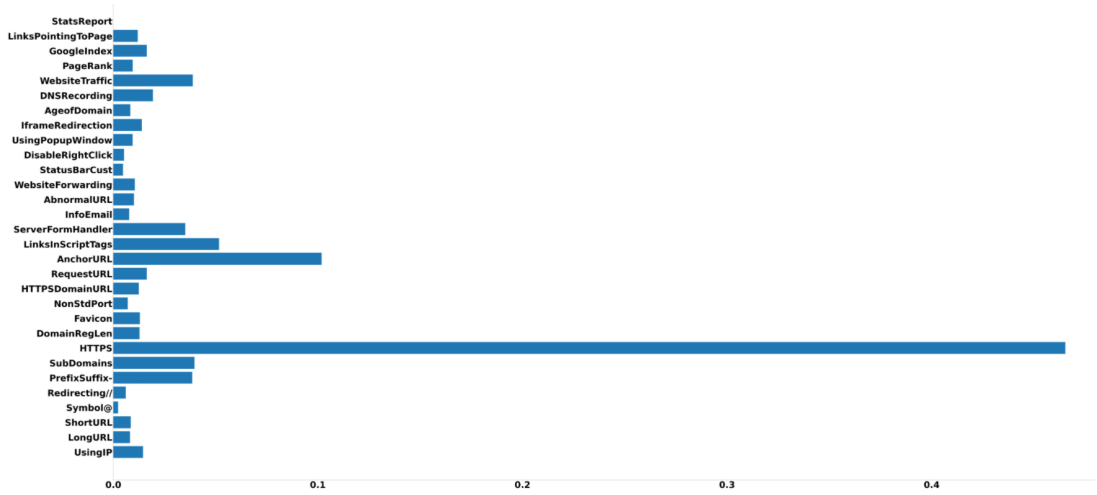

Figure 2: Feature Importance for Detecting Phishing Websites by XgBoost Library

As it can be observed from the figure above, HTTPs is the most important feature to detect phishing websites by a world of distance according to XgBoost Classifier. Additionally, AnchorUrl and LinksInScriptTags are the second and third most important features respectively. On the other hand, StatsReport is the least important feature and may be excluded in the feature selection process.

However, we cannot just use one single algorithm to decide which features will be in the training data set because the default XgBoostClassifier may under-fit or over-fit the data set and resulting in not using StatsReport (or other features with small importance score) in the decision trees. For this reason, we also calculated the correlations among the pairs of features to have further understanding. Sake of simplicity in tables, we only included features with high correlations (> 0.5 or < -0.5).

IframeRedirection-UsingPopupWindow: 0.62	Favicon-NonStdPort: 0.80	NonStdPort-StatusBarCust :0.62	Favicon-StatusBarCust :0.70	ShortURL-AbnormalURL :0.73	AbnormalURL-Redirecting// :0.72
ShortURL-HTTPSDomainURL :0.75	StatusBarCust-UsingPopupWindow :0.73	Redirecting//-ShortURL :0.84	AbnormalURL-Redirecting// :0.72	ShortURL-Redirecting// :0.84	InfoEmail-Favicon :0.66
IframeRedirection-StatusBarCust :0.65	Redirecting//-HTTPSDomainURL :0.76	HTTPSDomainURL-AbnormalURL :0.71	InfoEmail-StatusBarCust :0.53	InfoEmail-NonStdPort :0.79	Favicon-StatusBarCust :0.70
HTTPSDomainURL-Redirecting// :0.76	NonStdPort-InfoEmail :0.79	ShortURL-AbnormalURL :0.73	Favicon-IframeRedirection :0.62	IframeRedirection-NonStdPort :0.68	IframeRedirection-DisableRightClick :0.65
NonStdPort-IframeRedirection :0.68	UsingPopupWindow-Favicon :0.93	Favicon-InfoEmail :0.66	NonStdPort-Favicon :0.80	IframeRedirection-Favicon :0.62	Favicon-UsingPopupWindow :0.93
StatusBarCust-NonStdPort :0.62	StatusBarCust-IframeRedirection :0.65	Favicon-NonStdPort :0.80	HTTPSDomainURL-ShortURL :0.75	InfoEmail-UsingPopupWindow :0.62	Redirecting//-AbnormalURL :0.72
HTTPS-AnchorURL :0.53	UsingPopupWindow-InfoEmail :0.62	AbnormalURL-HTTPSDomainURL :0.71	UsingPopupWindow-StatusBarCust :0.73	InfoEmail-IframeRedirection :0.57	UsingPopupWindow-IframeRedirection :0.62
IframeRedirection-UsingPopupWindow :0.62	AbnormalURL-ShortURL :0.73	StatusBarCust-Favicon :0.70	IframeRedirection-InfoEmail :0.57	Redirecting//-WebsiteForwarding: -0.59	RequestURL-DomainRegLen: -0.60
WebsiteForwarding-ShortURL: -0.53	ShortURL-WebsiteForwarding: -0.53	DomainRegLen-RequestURL: -0.60	WebsiteForwarding-Redirecting//: -0.59		

Figure 3: Highest Correlations among Feature Pairs

As the table states there are some feature pairs with strong correlations. One of two features in some of these pairs can be eliminated from the training data set to avoid over-fitting the training data in the next steps of our project.

Finally, in order to select features for our machine learning algorithms, we developed the algorithm FVV proposed in the paper. Since we mentioned the FVV algorithm in the section[4], you can refer to the details in that section.

As the below FVV table states, HTTPs is the most successful feature to detect phishing websites very similar to what XgBoost feature importance method states. In addition to that, Website Forwarding is the less successful feature to detect phishing websites, but this doesn't mean that it is not important for the machine learning algorithms. Since there are only two labels (-1 and 1), if we flip the value of Website Forwarding (1 to -1 or -1 to 1), it'll have a very good score (probability to detect phishing websites). To conclude, when the FSS value of a feature is more distant than 0.5 (Pure Guessing Probability), the more that feature potentially contributes to machine learning algorithms to detect phishing websites.

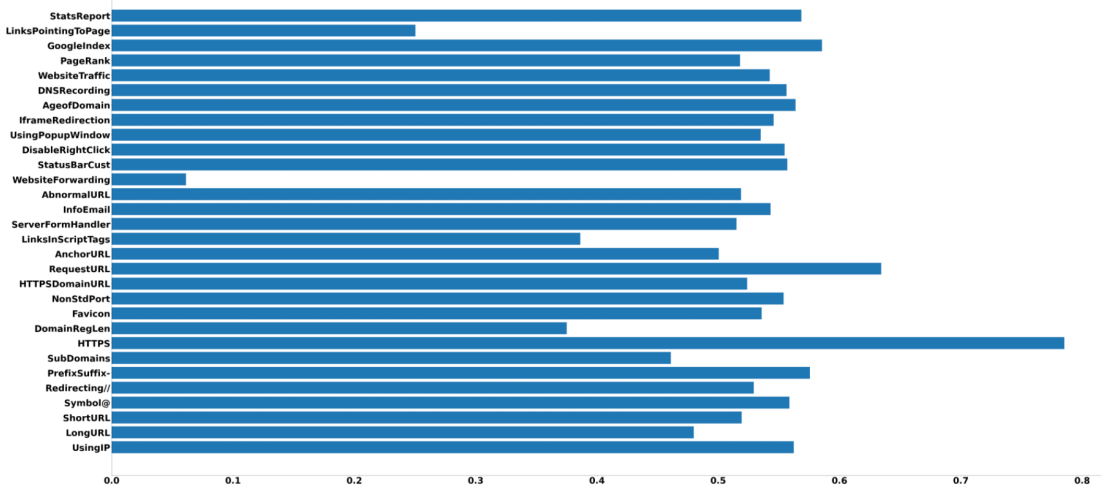

Figure 4: FVV Probabilities for each Feature

6 Feature Selection

In order to find the optimal features to be used in the machine learning algorithms, the following feature selection algorithm is used.

Algorithm 1: Feature Selection

Result: Best feature setting to be used in the ML Algorithms
Run ML Algorithms with all features;
Calculate validation accuracies;
Go back Analysis of the Dataset; Ex(In)clude some features;
while *Not Terminating* **do**
    Run ML Algorithms with new feature setting;
    Calculate validation accuracies and compare with old ones;
    Go back Analysis of the Dataset; Ex(In)clude some features;
end
Report the best feature setting in terms of validation accuracy;

By using the feature selection algorithm, StatsReport and Symbol@ features are eliminated to have better validation accuracies in the ML algorithms. ML algorithms and the performance of these algorithms are explained in the next section[7].

7 Machine Learning Algorithms

7.1 Neural Networks

The first ML algorithm developed is Neural Networks as proposed in OFS-NN[4]. Although hyper-parameter tuning is executed for most of the hyper-parameters of NN, only hidden layers hyper-parameter has increased the validation accuracy rate significantly. The following plot shows the change in validation accuracy rate versus the number of hidden layers in NN.

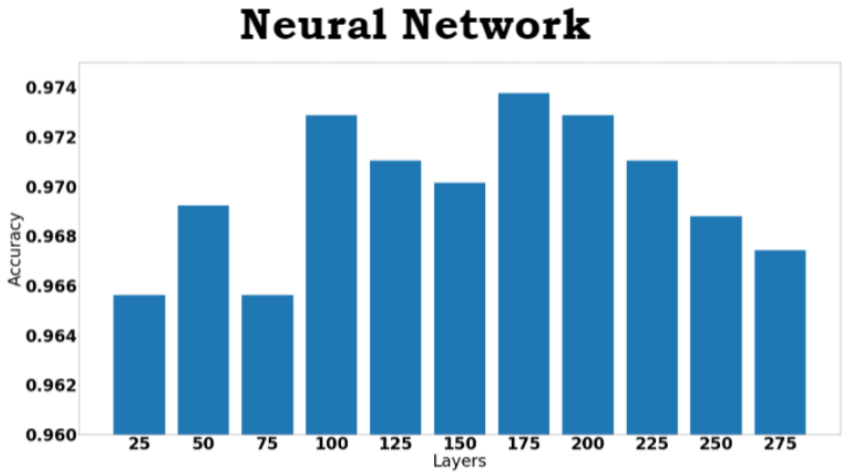

Figure 5: Validation Accuracy versus Numbers of Hidden Layers in NN

As a result, 0.974 validation accuracy achieved using NN.

7.2 XGBoost

We applied a tree-based ensemble method XGBoost on our data-set that aims to provide a scable, portable and distributed gradient boosting library. XGBoost is an implementation of gradient boosted decision trees designed for speed and performance. Gradient boosting is an approach where new models are created which predict the residuals or errors of prior models and then added together to make the final prediction. In order to build this model we split the training set into training-validation set with $0.7 : 0.3$ portion. First trial of training XGBoost is over-fitted the training set (%98 training

accuracy and %93 test accuracy) with training 2 minutes training time. Although this model is over-fitted, training time was promising. Considering this observation, we tuned the parameters of XGBoost and increased the accuracy of testing to%95. The tuned model is build according to the settings in Table below:

XGBoost Parameter Tuned	
Parameter	Value
n_estimators	1000
rate	0.1
gamma	1
max_depth	6
reg_lambda	0.15
reg_alpha	0.15
max_bin	1024

	precision	recall	f1-score	support
-1	0.96	0.94	0.95	1469
1	0.96	0.97	0.96	1848
accuracy			0.96	3317
macro avg	0.96	0.96	0.96	3317
weighted avg	0.96	0.96	0.96	3317

Figure 6: XGBoost Classification Report after Parameter Tuning

We built a model with respect to the importance of the feature that noticeable change in the performance did not be observed. Considering evaluation set as training-validation set, and evaluation metrics as "error " and "log loss", we monitor the training loss and validation loss.It is obvious in figures below that after almost 75 epoch the model tends to over-fit. Accordingly, by setting an early stopping equal to 75, we achieve %0.96 accuracy.

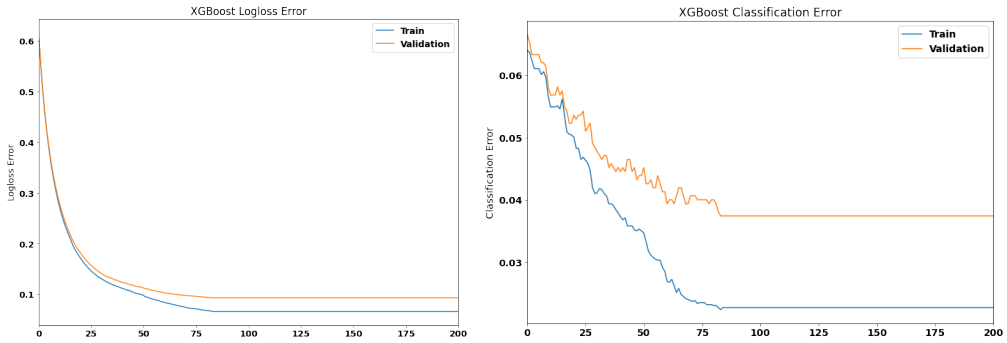

Figure 7: Monitoring the Training Loss and Validation Loss of Model

	precision	recall	f1-score	support
-1	0.97	0.95	0.96	720
1	0.96	0.97	0.97	828
accuracy			0.96	1548
macro avg	0.96	0.96	0.96	1548
weighted avg	0.96	0.96	0.96	1548

Figure 8: XGBoost Classification Report with Early Stop = 75

7.3 Gradient Boosting - LightGBM

LightGBM uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value while XGBoost uses pre-sorted algorithm and Histogram-based algorithm for computing the best split. In order to build this model we split the training set into training-validation set with 0.7 : 0.3 portion. Setting the objective and boosting of LightGBM as "binary" and "Gradient Boosting Decision Tree (gbdt)", initial model was built. the observation was promising since it had lower training time and performance similar to XGBoost. Considering this observation, we tuned the related parameters of LightGBM and we get %0.98 accuracy. The tuned parameters are denoted in Table below:

LightGBM Parameter Tuned	
Parameter	Value
n_estimators	1000
extra_trees	"True"
objective	'binary'
learning_rate	0.05
max_depth	10
feature_fraction	0.9
bagging_freq	8
bagging_fraction	0.8
bagging_seed	15
reg_alpha	0.15
reg_lambda	0.15
save_binary	True
random_state	42

	precision	recall	f1-score	support
-1	0.98	0.97	0.98	720
1	0.97	0.99	0.98	828
accuracy			0.98	1548
macro avg	0.98	0.98	0.98	1548
weighted avg	0.98	0.98	0.98	1548

Figure 9: LightGBM Classification Report after Parameter Tuning

7.4 SVM

Support vector machines are one of the most robust prediction methods, being based on statistical learning frameworks. We applied SVM method on the data-set with Linear and RBF values for kernel.

7.4.1 Linear Kernel

We tuned the parameter 'C' : [0.0001, 0.001, 0.01, 0.1, 1, 10, 100], and the best parameter set found on the development set as 'C' = 0.1 with %0.925

accuracy.

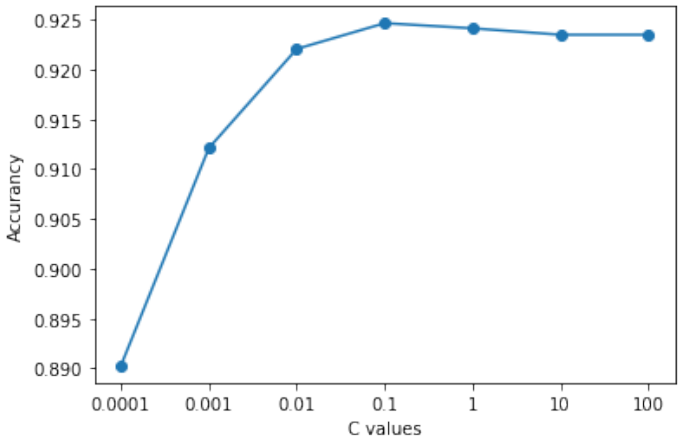

Figure 10: Linear Kernel 'C' Parameter Tuning Result

	precision	recall	f1-score	support
-1	0.94	0.90	0.92	1469
1	0.92	0.95	0.94	1848
accuracy			0.93	3317
macro avg	0.93	0.93	0.93	3317
weighted avg	0.93	0.93	0.93	3317

Figure 11: Linear Kernel Classification Report after Parameter Tuning

7.4.2 RBF Kernel

We tuned the parameters 'C' : [0.0001, 0.001, 0.01, 0.1, 1, 10, 100] and 'gamma': [0.0625, 0.125, 0.25, 0.5, 1]. We found the best parameter se on the development set as 'C' = 100 and 'gamma' = 0.125 with %0.957 accuracy.

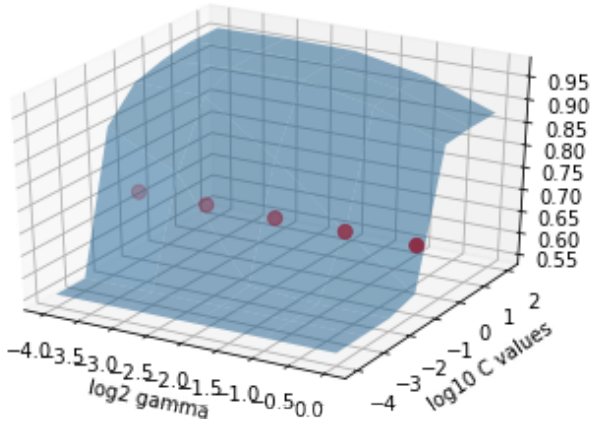

Figure 12: RBF Kernel 'C' and 'gamma' Parameter Tuning Result

	precision	recall	f1-score	support
-1	0.97	0.95	0.96	1469
1	0.96	0.98	0.97	1848
accuracy			0.97	3317
macro avg	0.97	0.97	0.97	3317
weighted avg	0.97	0.97	0.97	3317

Figure 13: RBF Kernel Classification Report after Parameter Tuning

8 Conclusion

From the discussion in Introduction, the critical tasks in this problem is to recognize a phishing website URL. We applied different classification methods on the data-set of 11000 website URLs from kaggle and we compare the result of each of the models we developed in order to find the best model to detect a phishing website. Table below denotes the classification result of each classifier. Since the data-set is balanced, for each of the models, no significant difference among the precision and recall is observed.

Classification Results	
Model	Accuracy
LightGBM	0.98
Neural Network	0.97
XGBoost	0.96
SVM: RBF	0.96
SVM: Linear	0.92

Based on the results it is obvious that LightGBM outperforms the other models performance with accuracy equal to %0.98. Moreover, it has higher training speed and is memory-efficient.

References

[1] M. Ahsan, R. Gomes, and A. Denton, "Smote implementation on phishing data to enhance cybersecurity," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, 2018, pp. 0531–0536.

[2] J. Bohacik, I. Skula, and M. Zabovsky, "Data mining-based phishing detection," in *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2020, pp. 27–30.

[3] M. Thaker, M. Parikh, P. Shetty, V. Neogi, and S. Jaswal, "Detecting phishing websites using data mining," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. IEEE, 2018, pp. 1876–1879.

[4] E. Zhu, Y. Chen, C. Ye, X. Li, and F. Liu, "Ofs-nn: an effective phishing websites detection model based on optimal feature selection and neural network," *IEEE Access*, vol. 7, pp. 73 271–73 284, 2019.

[5] E. Chand, "Phishing website detector," Feb 2020. [Online]. Available: https://www.kaggle.com/eswarchandt/phishing-website-detector?select=phishing.csv