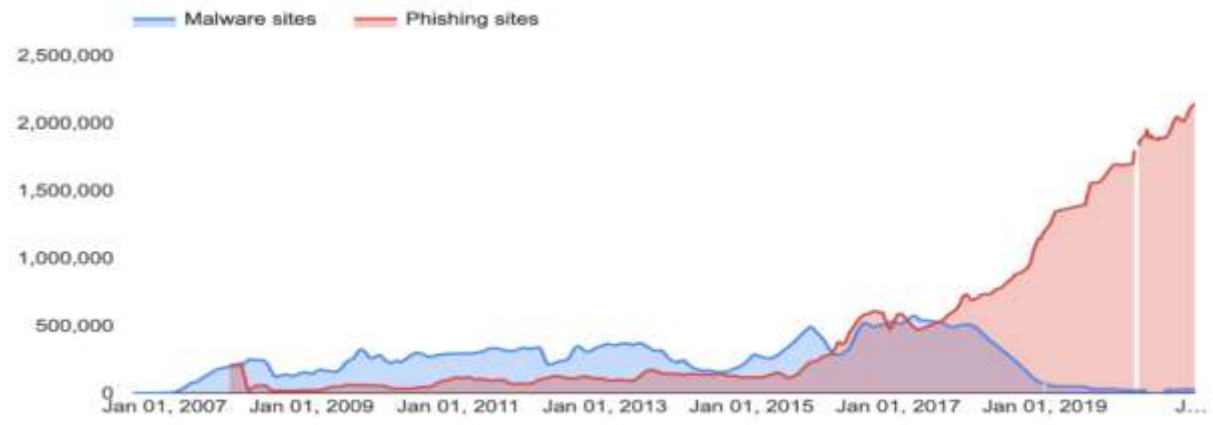
Phishing Website Detection

Udit Joshi & Kristoffer Larsen
MA5790
April 2020

Abstract:

Phishing is one of the familiar attacks that trick users to access malicious content in order to gain their information. As of September 2021, the number of unique active phishing sites has been steadily increasing pressing the need for solutions. In terms of website interface and uniform resource locator (URL), the majority of phishing webpages look identical to the imitated webpages. Currently, blacklists—a giant list of websites that are deemed dangerous, such as Google Safe Browsing—are used to prevent users from visiting these fraudulent websites. An alternative approach was proposed in this paper using a heuristic-based method that considers features from these phishing websites in order to discern phishing or legitimate status. Website features will be explored for their relationship between predictors and the response (phishing status) to find the best subset of predictors. Using predictive modeling, linear and nonlinear classification models will be trained and validated to select the best overall model. Proceeding this, variable importance will be generated to investigate features of interest to consider for distinguishing website status.



This chart - pulled from Google Safe Browsing - shows the steep increase in the number of websites deemed unsafe between January 2016 and January 2021.

Table of Contents

Background

Phishing attacks target the user's computer or attempt to direct victims to malicious websites. In turn tricking users into divulging personal and financial information, such as passwords, usernames, account IDs, and credit card numbers. Phishing attacks are the most common type of cyber-attacks used to obtain sensitive information and have been on the rise, especially as a result of COVID-19 where there is a greater presence of online living and working. Attacks not only affect individuals, but also disrupt organizations across the globe to gain trade secrets, company login, and data. For example, large multinational companies like Sony Pictures, Google, and Facebook have all suffered hundreds of millions of dollars in losses as a result of phishing attacks.

The first approach to detecting phishing websites uses a blacklist, where the given URL is compared with the URLs contained within the blacklist. The downside of this approach is based on the limitation that the blacklist can not contain all possible phishing websites. Malicious users can simply make new phishing websites and trick numerous individuals before they are eventually blacklisted. This results in a sizable lag for phishing attacks to occur within. Additionally, this approach is not efficient when hundreds of thousands of phishing websites are generated every day meaning to compare a single given website to the blacklist produces new problems. Common and well-used blacklists include APWG, Google Safe Browsing API, and PhishTank archive.

A more efficient strategy referred to as a heuristic-based method involves collecting certain features from a website in order to distinguish it as phishing or legitimate. In doing so, the attributes that make up the website can be examined in real-time without the need to consult a blacklist. This approach has the added benefit that even if the fraudulent website was launched relatively recently (and has yet to be added to a blacklist) its status can still be probabilistically discerned.

Variable Introduction and Definitions

The dataset was accessed through the UCI Repository. Alternatively, the dataset can be accessed through the University of Huddersfield Repository and Mendeley Data. The data itself was collective from blacklists, mainly PhishTank archive, MillerSmiles archive, and Google collaborated searching operators. The target variable is categorical with two levels denoting the status of the website either legitimate or phishing. There are 11,055 website samples and 30 categorical predictor variables corresponding to the attributes of the website—22 predictors having only two levels and 8 predictors having three levels. The following is a list of the variables, their respective names within the analysis, and their corresponding description with respect to the specific website attributes.

Variables	**Description**
1) Address Bar Based Features -	
having_IP_Address	URL contains IP address?
URL_Length	URL length greater than or equal to 54?
Shortining_Service	Uses “TinyURL”?
having_At_Symbol	URL contains “@”?
double_slash_redirecting	URL contains “//”?
Prefix_Suffix	URL domain contains “-”?
having_Sub_Domain	URL domain contains “.”?
SSLfinal_State	Uses HTTPS and has certificate?
Domain_registeration_length	Domain expires in less than one year?
Favicon	Favicon loaded from an external domain?
port	Port # is of the Preffesered Status?
HTTPS_token	Uses “HTTPS” token in the domain section?
2) Abnormal Based Features -	
Request_URL	% of requested URL less than 22%? Or bagged between 61%? Or greater?
URL_of_Anchor	% of URL of anchor “<a>” less than 31%? Or bagged between 67%? Or greater?
Links_in_tags	% of Links in “<Meta>”, “<Script>”, and “<Link>” less than 17%?
SFH	SFH contains an empty string or “about:blank”? Or does SFH refer to a different domain entirely?
Submitting_to_email	Uses “mail()” or “mailto:” function to submit user information?
Abnormal_URL	Hostname included in URL?

3) <u>HTML and JavaScript-based Features</u> -

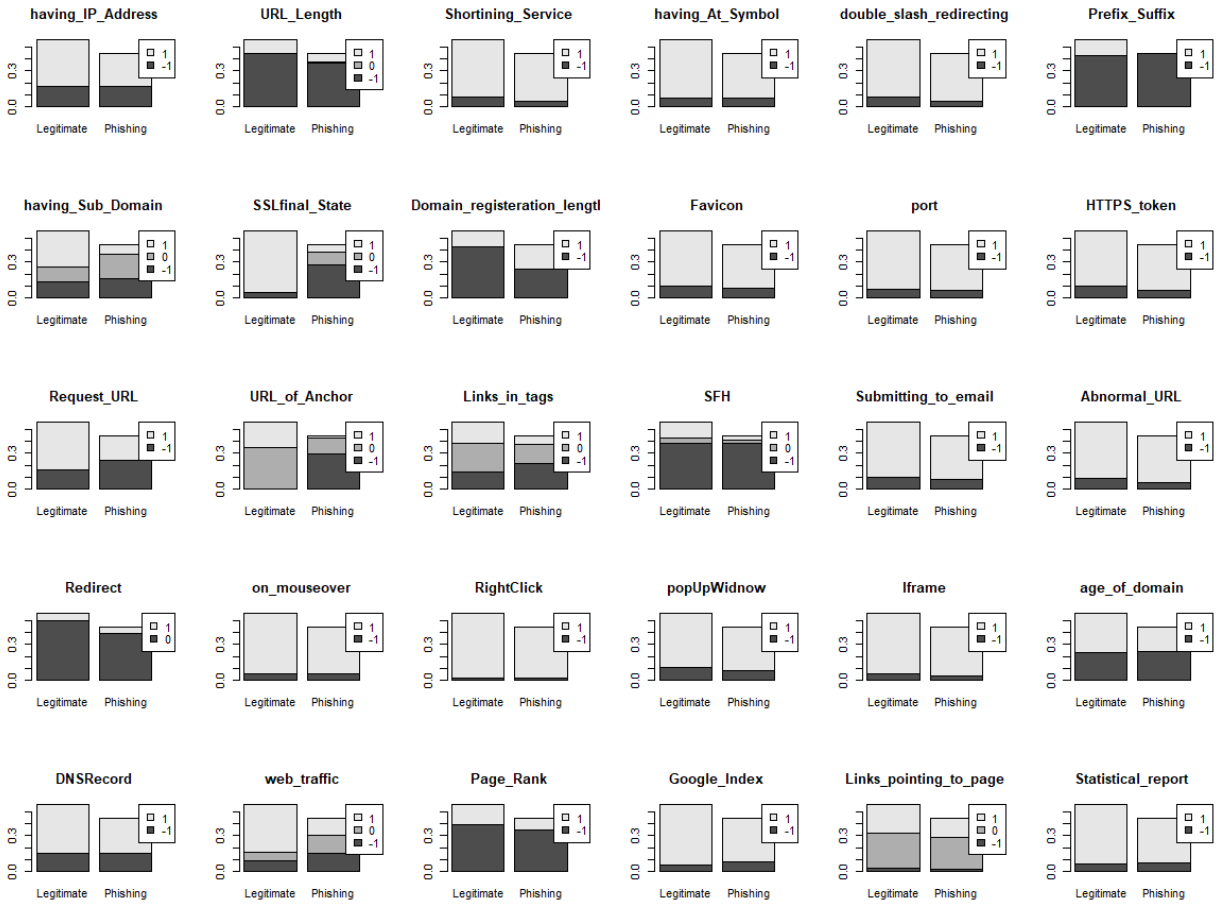
Redirect	# of redirect pages less than one? Or between two and four? Or greater than 4?
on_mouseover	onMouseOver changes the status bar?
RightClick	Right click disabled?
popUpWidnow	Pop-up window contains text fields?
Iframe	Uses iframe?

4) <u>Domain-based Features</u> -

age_of_domain	Age of domain greater than six months?
DNSRecord	Exists DNS record for the domain?
web_traffic	Website traffic rank below 100,000?
Page_Rank	Website PageRank below 0.2?
Google_Index	Webpage indexed by google?
Links_pointing_to_page	Number of links pointing to the webpage equals zero? Or bagged between two? Or greater than two?
Statistical_report	Host belongs to the top ten phishing IPs/domains?

Preprocessing of the Predictors

Before building predictive models to classify the status of a website as legitimate or phishing it is necessary to preprocess the data so that it will be clean. The data itself had no missing values meaning imputation or complete case analysis was not demanded; however, because all the predictors were categorical it was imperative to add dummy variables. From the original 30 predictors, 22 predictors (two levels) were turned into dummy variables resulting in 44 new predictors, but for each original feature, only one dummy variable needed to be retained resulting in 22 dummy variables. From the original 30 predictors, 8 predictors (three levels) were turned into dummy variables resulting in 24 new predictors, but for each original feature, only two dummy variables needed to be retained resulting in 16 dummy variables. This results in a grand total of 38 (dummy variable) predictors.

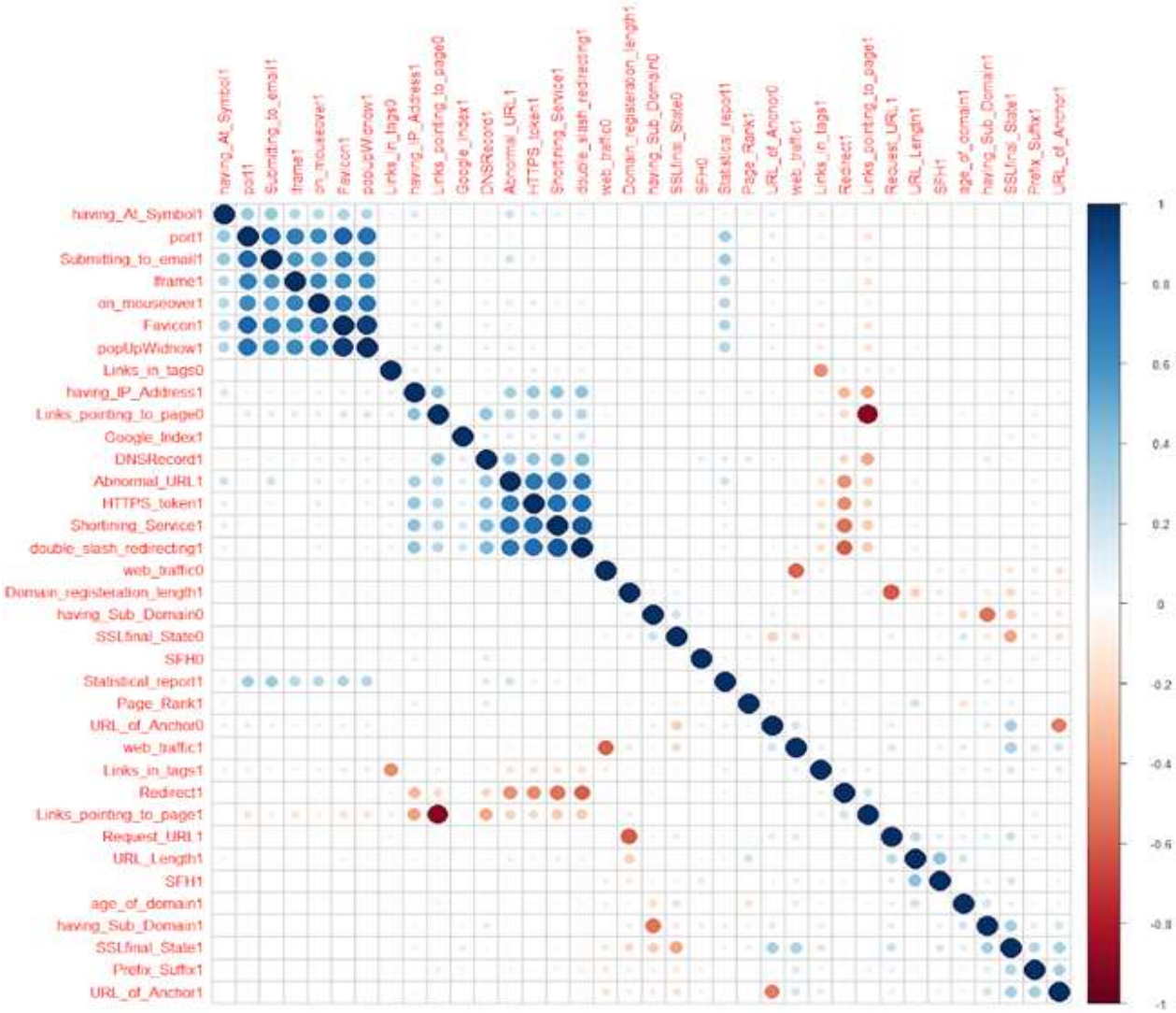


Above is a grouping of boxplots from the original data showing the distribution of the categorical predictors with respect to the response level either legitimate or phishing. For the majority of predictors, there is not an easy distinction between the proportion of one level with either response i.e. a predictor level has a large split in the proportion between levels. However, some predictors like “web_traffic”, “having_Sub_Domain”, and “SSLFinal_State” do have some visual distinction

between classes. In the case of "SSLFinal_State", most legitimate websites have a level of 1 which corresponds with the website using HTTPS which makes sense considering it is a security protocol widely implemented in Web servers.

After dummy variables were added, it became necessary to remove predictors which had near-zero variance in order to remove degenerate predictors which would not be useful in contributing to the models. Two predictors were removed in this fashion, a dummy variable relating to "URL_Length" and "RightClick".

Correlation is the next step. Predictors with greater than 0.90 correlations were removed from the predictor space. In the correlation plot below, a high correlation is shown in the dark blue representing high positive correlations, and dark red represents high negative correlations. From this step two predictor were removed, a dummy variable relating to "age_of_domain" and "Statistical_report".



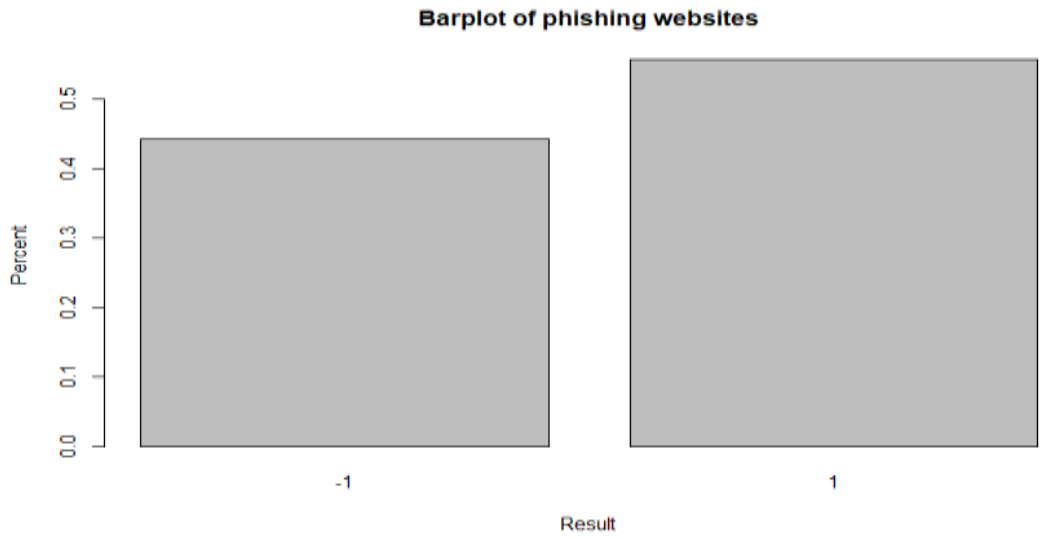
In grand total after preprocessing, there are 11,055 samples in the data with 34 features in the reduced dataset (correlation removed) and 36 features in the full dataset (non-correlation removed) which will be used in models such as partial least-squares discriminant analysis (PLS-DA).

The remaining dummy variable predictors in the predictor space are shown below. Those highlighted are ones who are excluded in the reduced dataset but are retained in the full dataset.

```
 [1] "having_IP_Address1"          "URL_Length1"                  "Shortining_Service1"
 [4] "having_At_Symbol1"           "double_slash_redirecting1"    "Prefix_Suffix1"
 [7] "having_Sub_Domain0"          "having_Sub_Domain1"           "SSLfinal_State0"
[10] "SSLfinal_State1"             "Domain_registeration_length1" "Favicon1"
[13] "port1"                       "HTTPS_token1"                 "Request_URL1"
[16] "URL_of_Anchor0"              "URL_of_Anchor1"               "Links_in_tags0"
[19] "Links_in_tags1"              "SFH0"                         "SFH1"
[22] "Submitting_to_email1"        "Abnormal_URL1"                "Redirect1"
[25] "on_mouseover1"               "popUpWidnow1"                 "Iframe1"
[28] "age_of_domain1"              "DNSRecord1"                   "web_traffic0"
[31] "web_traffic1"                "Page_Rank1"                   "Google_Index1"
[34] "Links_pointing_to_page0"     "Links_pointing_to_page1"      "Statistical_report1"
```

Data Splitting, Resampling, and Metrics

The status of a website can be classified into two levels: phishing (-1) or legitimate (1). However, these two classes are not completely balanced.



That means to properly split the data into training and testing split, a stratified random sampling method is employed. Using an 80%/20% training/testing ratio with stratified random sampling, both the train and test set will have equivalent proportions for each class. Following this method results in 8,845 samples in the training set and 2,210 samples in the testing set.

For the data resampling method, a simple 10-fold cross-validation was considered due to the sample size. It offered quick computation for the relatively large sample size. This method also provides overfitting reduction and allows for tuning parameter selection.

The model metric used to assess model performance is Cohen's Kappa used to measure the agreement between the observed accuracy and expected accuracy according to chance. This statistic is useful for unbalanced classes while also taking into account chance agreement. While the data itself is not completely balanced, in the real world it would be expected that the proportion of phishing websites compared to the proportion of legitimate websites is very small, i.e. although phishing websites are prominent, users are still more likely to come across legitimate websites due to the sheer amount of legitimate websites.

Model Fitting

For this problem statement, we are training different kinds of linear classification models such as Logistic Regression, Linear Discriminant Analysis (LDA), Partial Least Squares Discriminant Analysis (PLSDA), and Penalized Model and nonlinear classification models such as Regularized Discriminant Analysis (RDA), Mixture Discriminant Analysis (MDA), Neural Networks, Flexible Discriminant Analysis (FDA), Support Vector Machines (SVM), K Nearest Neighbors (KNN), and Naive Bayes. And supplemental figures are shown in the appendix for any models with tuning parameters.
Below are the results for these models:

Summary of the Linear Classification Models

Models	Training	Testing
Logistic Regression	Accuracy - 0.9418, Kappa - 0.8819	Accuracy - 0.9294, Kappa - 0.8567
LDA	Accuracy - 0.9315, Kappa - 0.8608	Accuracy - 0.9127, Kappa - 0.8224
PLSDA	Accuracy - 0.9282, Kappa - 0.8542	Accuracy - 0.9136, Kappa - 0.8242
Penalized Model	Accuracy - 0.9395, Kappa - 0.8771	Accuracy - 0.9271, Kappa - 0.8520

In the linear models, first, we train the model on the training data and then evaluate it on test data based on the Kappa value. And finally, we selected the top two models i.e., Logistic Regression and Penalized Model (shown in the highlighted color in the above summary table).

Summary of the Nonlinear Classification Models

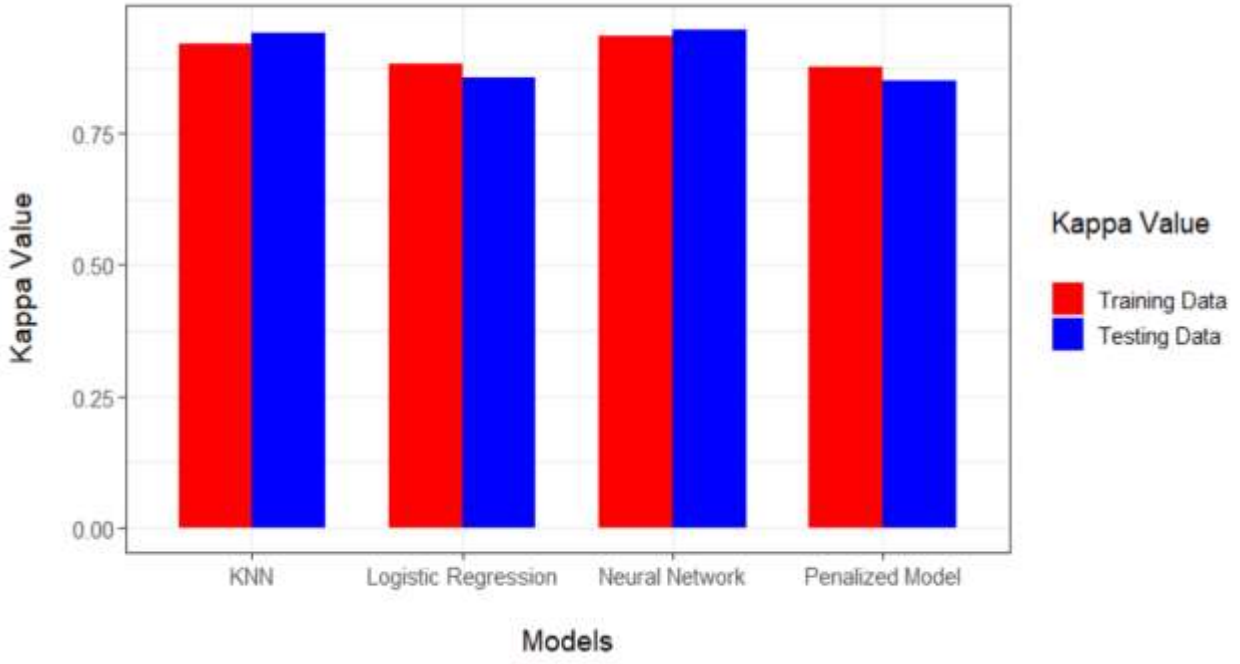
Models	Training	Testing
RDA	Accuracy - 0.9464, Kappa - 0.8914	Accuracy - 0.9362, Kappa - 0.8707
MDA	Accuracy - 0.9313, Kappa - 0.8603	Accuracy - 0.9127, Kappa - 0.8224
Neural Networks	Accuracy - 0.9688, Kappa - 0.9367	Accuracy - 0.9742, Kappa - 0.9476
FDA	Accuracy - 0.927, Kappa - 0.8524	Accuracy - 0.9195, Kappa - 0.8369
SVM	Accuracy - 0.9629, Kappa - 0.9248	Accuracy - 0.9670, Kappa - 0.9330
KNN	Accuracy - 0.9608, Kappa - 0.9204	Accuracy - 0.9715, Kappa - 0.9422
Naive Bayes	Accuracy - 0.9027, Kappa - 0.8001	Accuracy - 0.8968, Kappa - 0.788

In the nonlinear models also, first, we train the model on the training data and then evaluate it on test data based on the Kappa value. And finally, we selected the top two models i.e., Neural Networks and KNN (shown in the highlighted color in the above summary table).

Compare the Best Linear and Nonlinear Model

Models	Training	Testing
Best Linear Model		
Logistic Regression (Tuning Parameter - NA)	Accuracy - 0.9418, Kappa - 0.8819	Accuracy - 0.9294, Kappa - 0.8567
Penalized Model (Tuning Parameter - alpha = 0.1 and lambda = 0.01)	Accuracy - 0.9395, Kappa - 0.8771	Accuracy - 0.9271, Kappa - 0.8520
Best Nonlinear Model		
Neural Networks (Tuning Parameter - size = 20 and decay = 0.48)	Accuracy - 0.9688, Kappa - 0.9367	Accuracy - 0.9742, Kappa - 0.9476
KNN (Tuning Parameter - k = 1)	Accuracy - 0.9608, Kappa - 0.9204	Accuracy - 0.9715, Kappa - 0.9422

Comparison of training and testing Kappa values for the best models



After building the models, now we are comparing the best linear and nonlinear models with their testing Kappa value and find that nonlinear boundaries separate the binary classification problem much more accurately compared to the linear model as we can see in the above summary table, linear models had approx. 85% kappa value and nonlinear models had approx. 94% kappa value which is more than 9%.
Both nonlinear models such as Neural Network and KNN give approx. same kappa value, but KNN is much faster in terms of computation. And while looking into the confusion matrix, we find that KNN classifies the Phishing website more accurately compared to the Neural Network.
So, our best model for this problem statement is **K Nearest Neighbors (KNN).**

Summary

For this problem statement, we have tried different kinds of linear and nonlinear models, out of which we found that the Neural Network and KNN are giving the best Kappa value on the test data which is approximately the same. So, we decided to check the confusion matrix of both models.

Neural Network

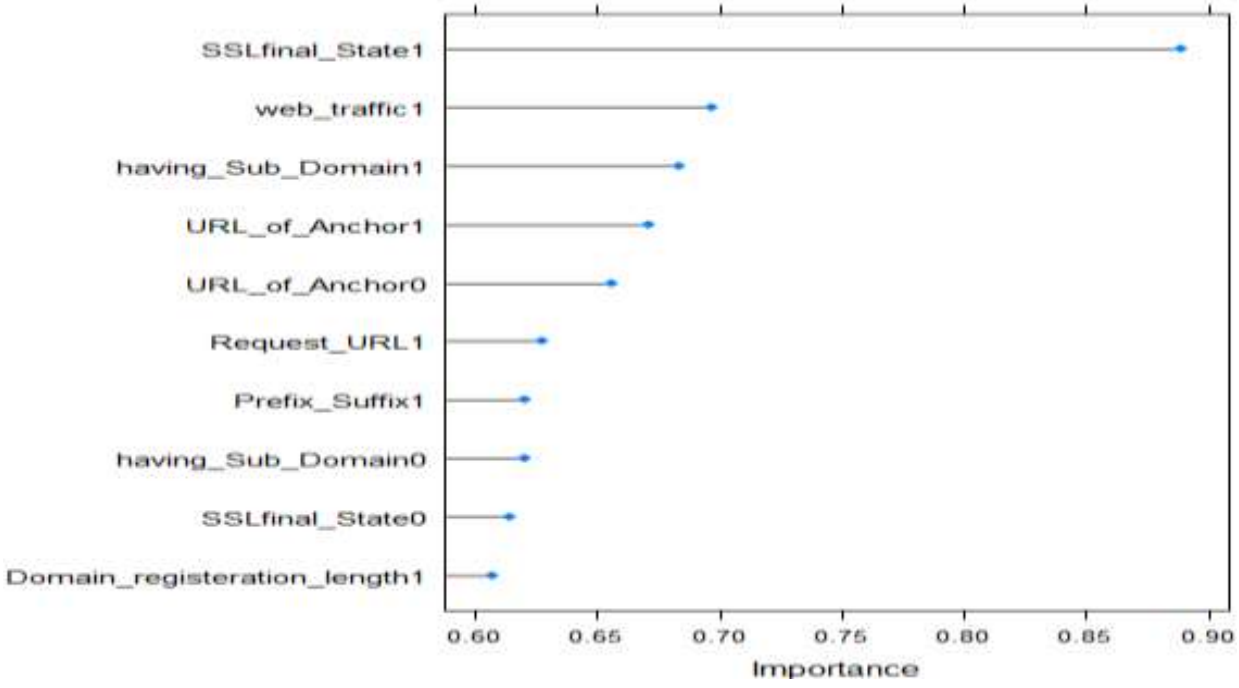
	Reference	
Prediction	Phishing	Legitimate
Phishing	940	18
Legitimate	39	1213

KNN

	Reference	
Prediction	Phishing	Legitimate
Phishing	945	29
Legitimate	34	1202

Based on the confusion matrix, we have chosen the KNN as the best model as we can see in the above table KNN predicts the phishing website more accurately compared to Neural Network. Out of 979 phishing websites, KNN correctly predicts 945, and the Neural network correctly predicts 940 which is less than the KNN model. And Computation is also cheap in KNN compared to Neural Network. So, our final selected model is KNN.

Important Variables based on KNN Model



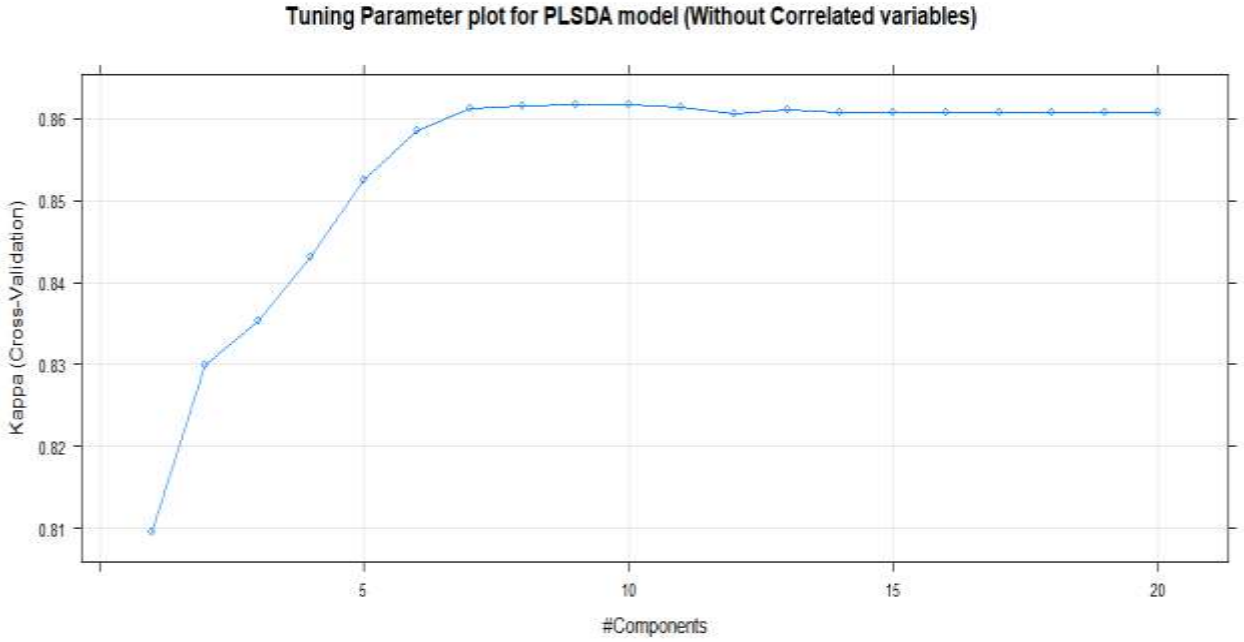
Based on the variable important plot above, by far the most important predictor is “SSLfinal_State”. As noted in the preprocessing when analyzing the bar plot, “SSLfinal_State” very clearly has a distinction between its levels and the response for the website status, i.e., most legitimate websites make use of HTTPS making it a very accurate predictor of website status. This means when examining a website to discern if it is possibly a phishing website, examine the URL to find “https://” as this generally is a good indicator that the website is legitimate, but this is not always the case. Second in importance is “web_traffic” which makes sense because legitimate websites tend to draw in more users for regular reasons, such as frequent visitors. While on the other hand, websites that are phishing tend to have lower traffic because they are generally sent through something like email which limits the number of users visiting the website. This is not to say that websites that have low traffic are phishing, but that if a website is phishing, they tend to have low traffic which also makes sense when a phishing website is eventually detected it is blacklisted preventing more users from visiting it causing its traffic to remain low.

Appendix 1: Linear Classification Supplemental Material

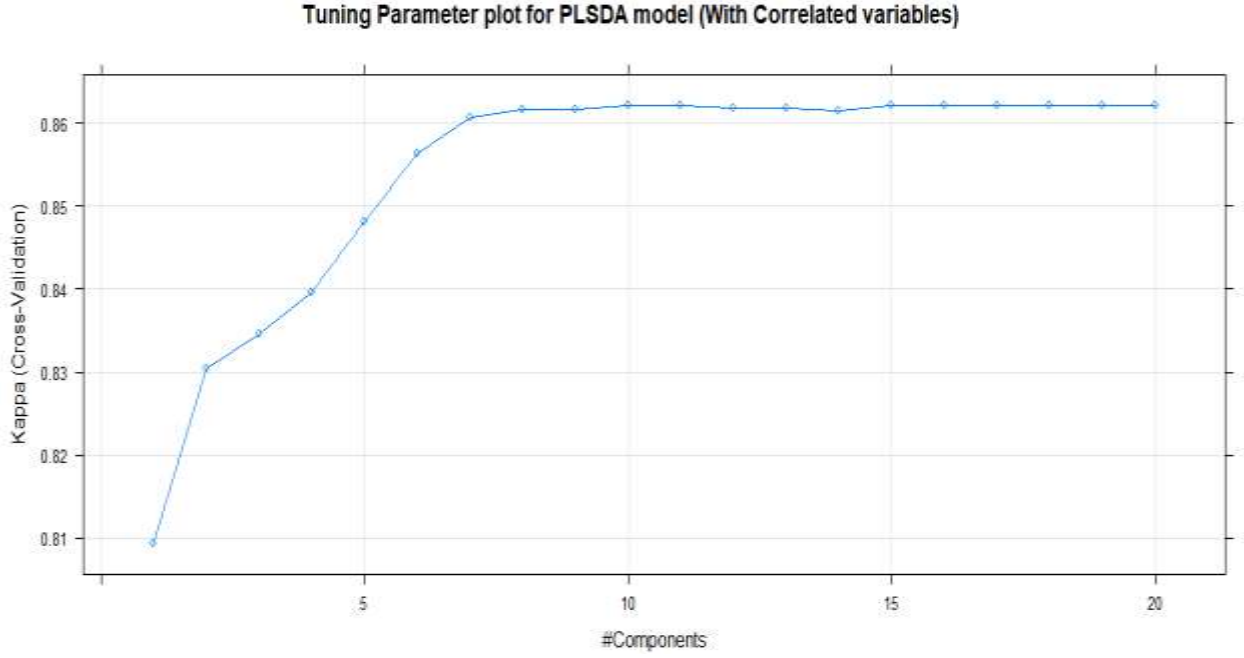
The linear model trained with any tuning parameters was the Partial Least Squares Discriminant Analysis (PLSDA), and Penalized Model.

1) **Partial Least Squares Discriminant Analysis (PLSDA)-**

After removing highly correlated variables, the optimal number of components selected as shown below is 9.

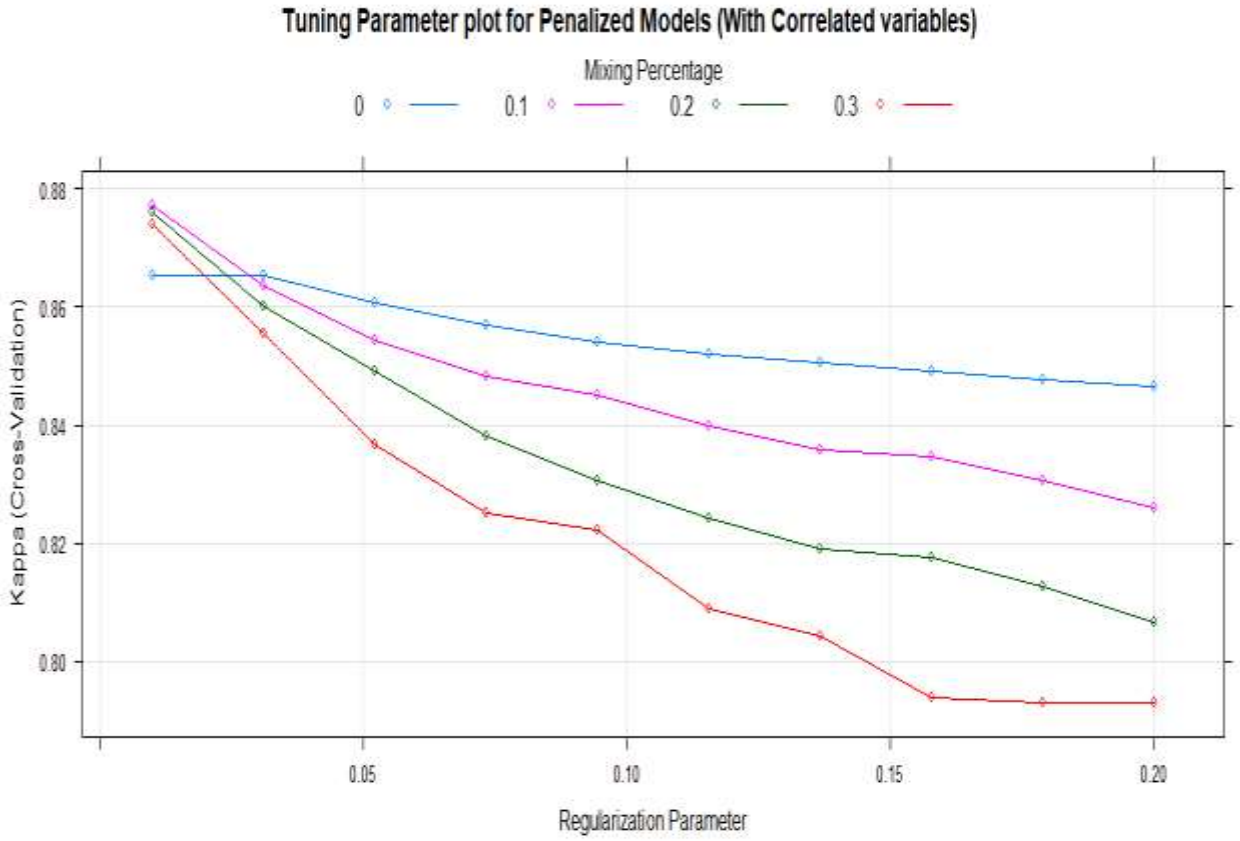


With consideration of highly correlated variables, the optimal number of components selected as shown below is 10.



2) **Penalized Model-**

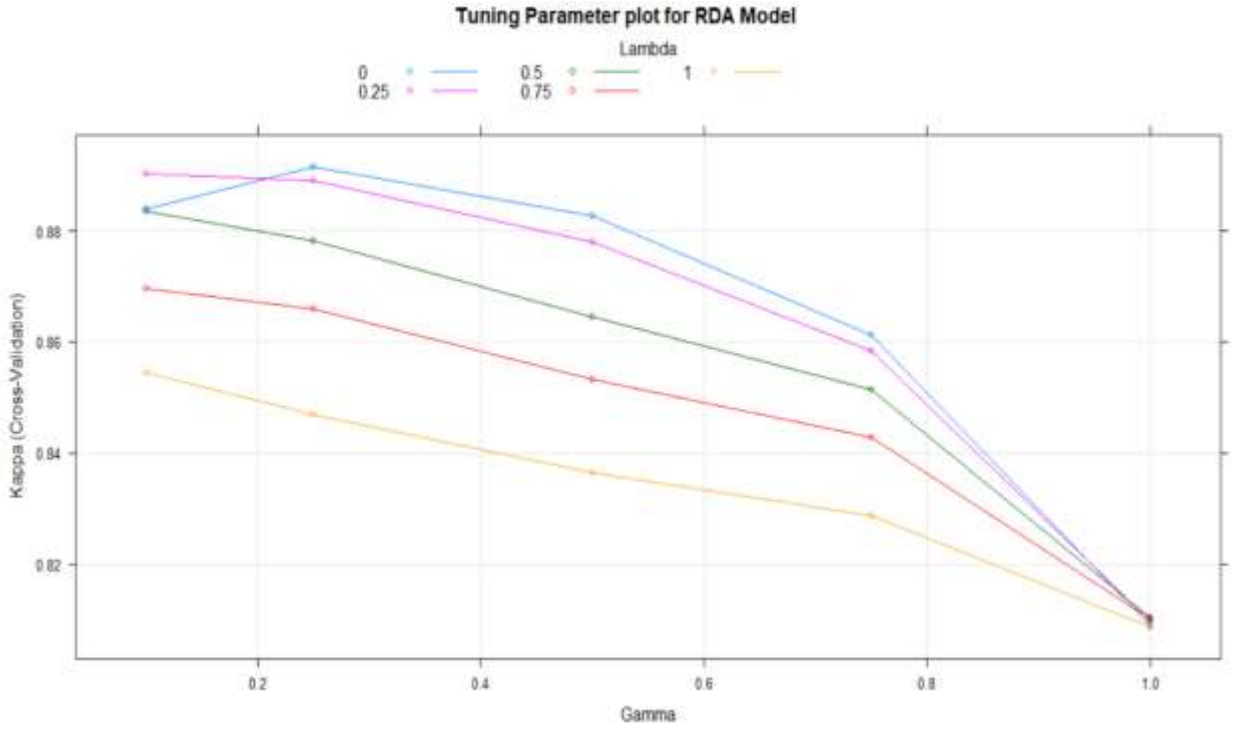
In penalized model, there are 2 tuning parameter - alpha = 0.1 and lambda = 0.01.



Appendix 2: Nonlinear Classification Supplemental Material

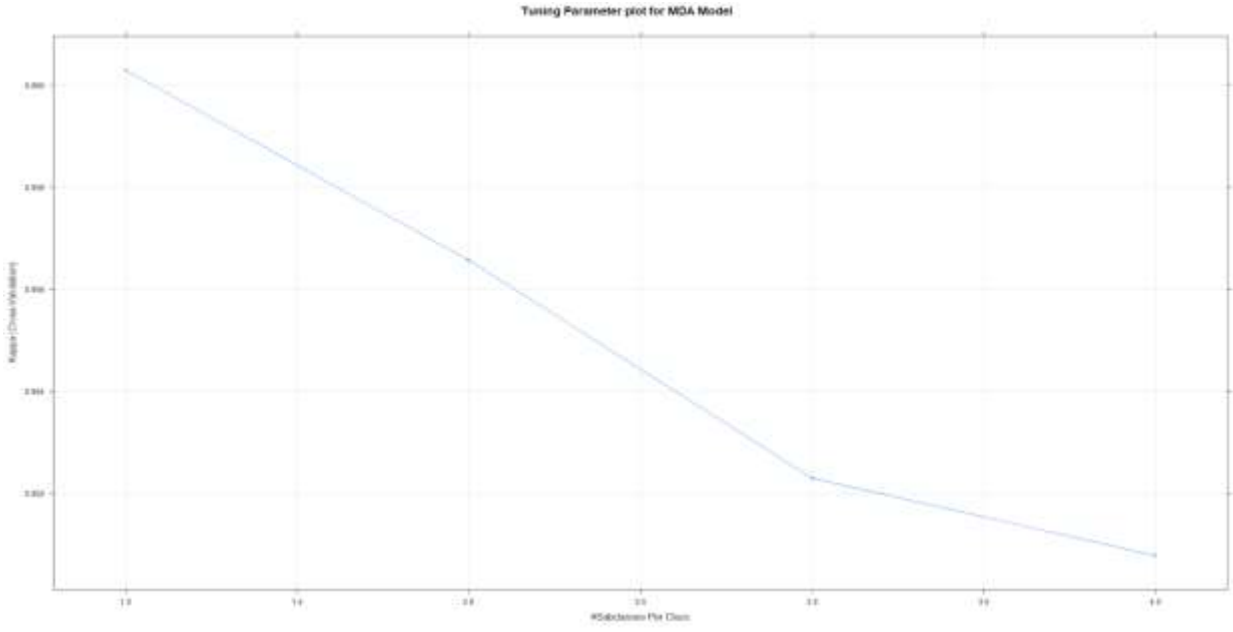
1) **Regularized Discriminant Analysis (RDA)**-

In RDA model, there are 2 tuning parameter - gamma = 0.25 and lambda = 0.



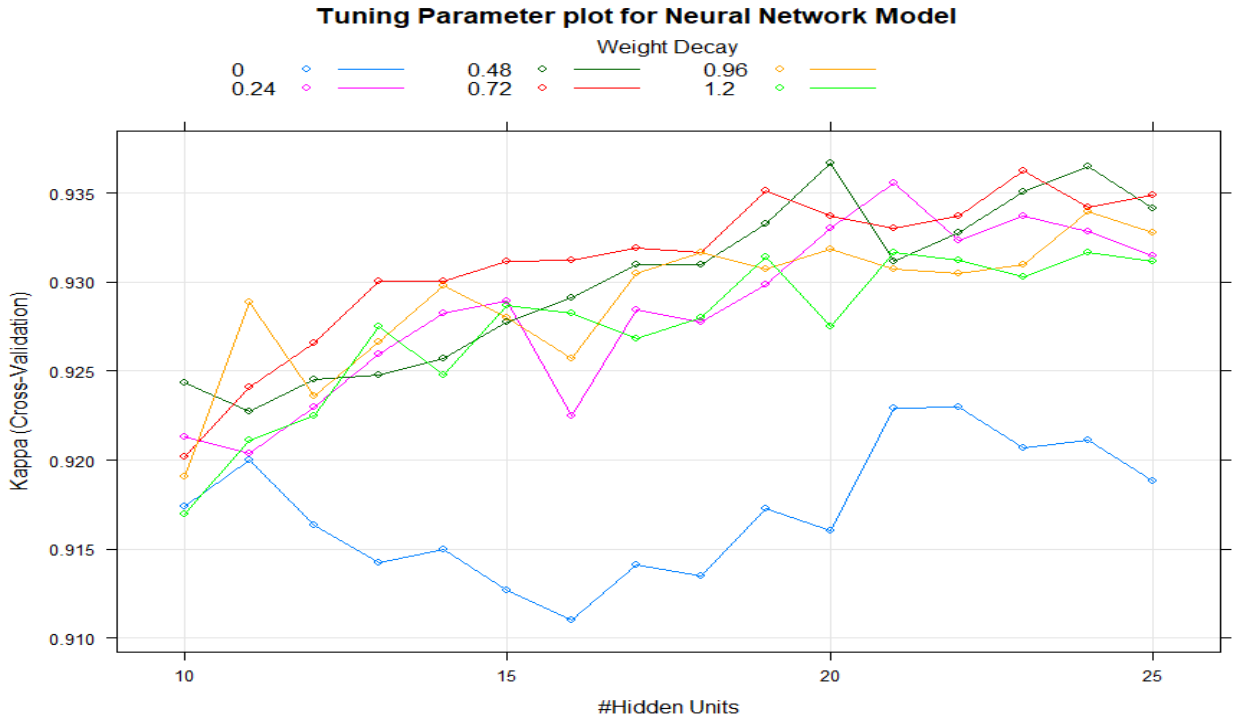
2) **Mixture Discriminant Analysis (MDA)**-

In the MDA model, there is 1 tuning parameter - subclasses = 1.



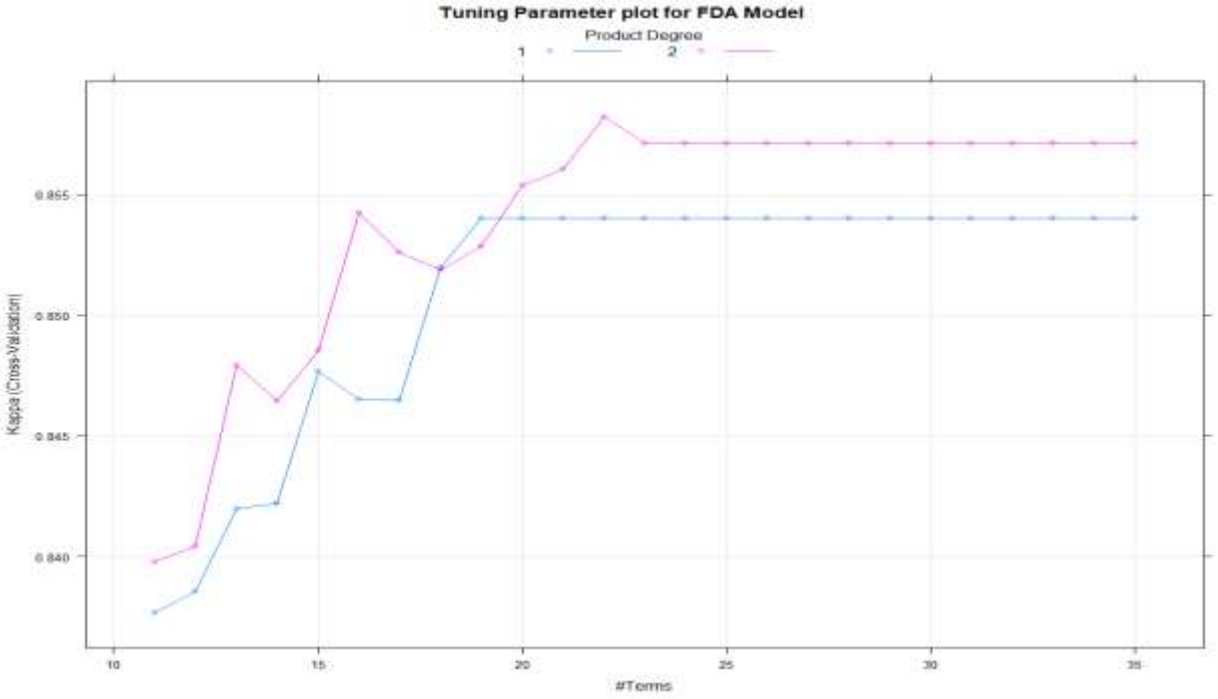
3) **Neural Networks-**

In Neural Networks model, there are 2 tuning parameter - size = 20 and decay = 0.48.



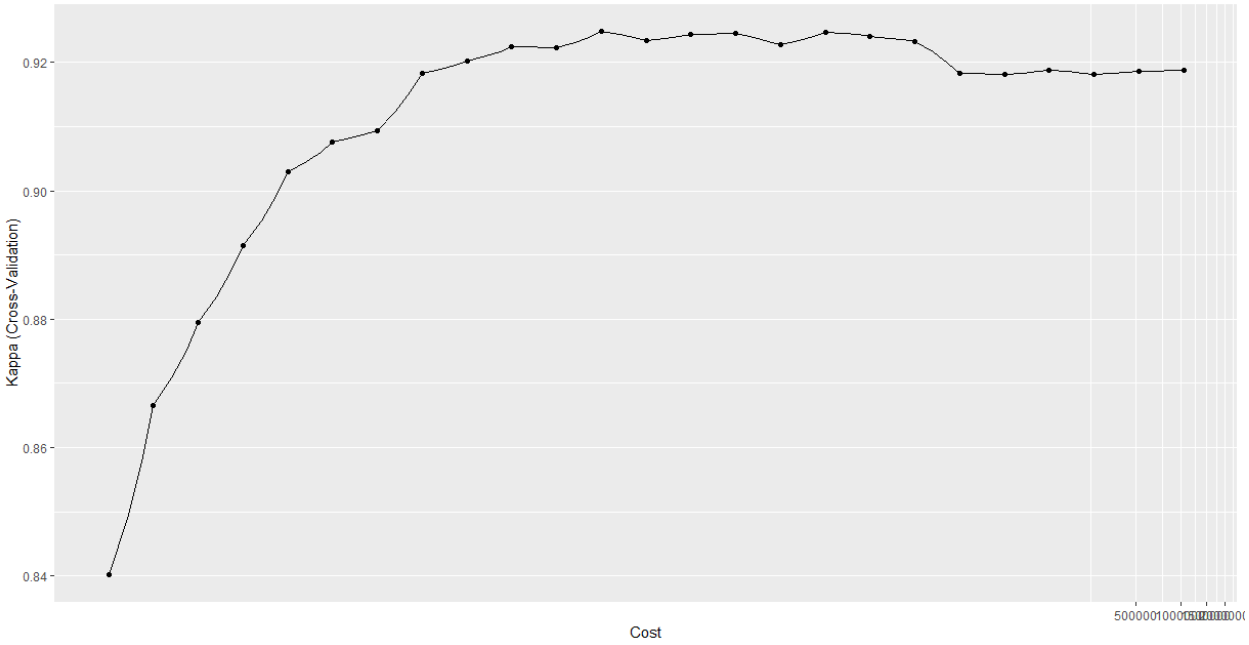
4) **Flexible Discriminant Analysis (FDA)-**

In FDA model, there are 2 tuning parameter - degree = 2 and nprune = 22.



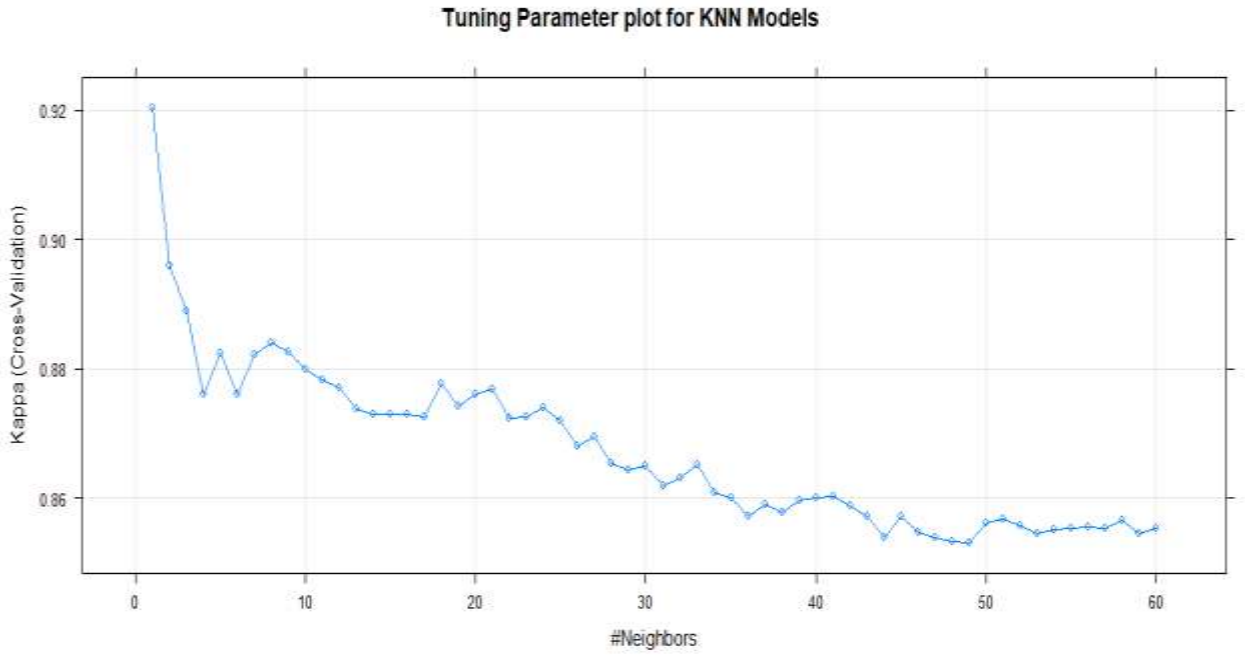
5) **Support Vector Machines (SVM)-**

In SVM model, there are 2 tuning parameter - sigma = 0.009246614 and C = 128.



6) **K Nearest Neighbors (KNN)-**

In the KNN model, there is 1 tuning parameter - k = 1.



R Code

```
################### MA5790 Phishing Website Detection Project ###################
Packages <- c('caret','dplyr','corrplot','earth','Formula','plotmo','plotrix',
             'TeachingDemos','kernlab','klaR')
lapply(Packages, library, character.only = TRUE)

library(doParallel)
cl <- makePSOCKcluster(5)
registerDoParallel(cl)


## data ##
full_data <- read.csv('Training_Dataset.csv')
response <- full_data[,31]
response <- ifelse(response == -1, 'Phishing', 'Legitimate')
response <- as.factor(response)
data <- full_data[,-31]


## dummy variables ##
data[] <- lapply(data, as.character)
dummyRes <- dummyVars("~.", data, fullRank = T); dummyRes
data <- data.frame(predict(dummyRes, newdata = data))
head(data)

## pre-proc ##
barplot(prop.table(table(response)), xlab = "Result (-1 = phishing, 1 =
      legitimate)", main = 'Barplot of Response', ylab= 'Count', horiz = F)

par(mfrow = c(5,6))
for(i in 1:30){
counts <- table(data[,i], response)
barplot(prop.table(counts), main = names(data[i]), legend.text =  T,
      args.legend = list(x = "topright",
                         inset = c(- 0.15, 0)))
}
par(mfrow=c(1,1))

# NZV #
nzv <- nearZeroVar(data)
nzv
data <- data[, -nzv]
str(data)

#Correlation
```

```
corr <- cor(data); corr
corrplot(corr, order = "hclust")
corrplot(corr, order = "FPC")

highCorr <- findCorrelation(corr, cutoff = .90); highCorr
colnames(data[,highCorr])
data_corr_RM <- data[,-highCorr]
colnames(data)
colnames(data_corr_RM)
######### Models #########
a <- 'cv'; b <- 10

#set seed for models
i <- 6428; j <- 3493

# CV
set.seed(j)
trainingRows <- createDataPartition(response, p = .80, list= FALSE)
subset_train <- data[trainingRows,]
subset_test <- data[-trainingRows,]
#nrow(subset_train)
subset_train_corr_rm <- data_corr_RM[trainingRows,]
subset_test_corr_rm <- data_corr_RM[-trainingRows,]

resp_train <- response[trainingRows]
resp_test <- response[-trainingRows]


#set test and training set
c <- subset_train_corr_rm
k <- subset_train
d <- resp_train

e <- subset_test_corr_rm
m  <- subset_test
f <- resp_test

################################ Linear Models ################################
ctrl <- trainControl(method = a,
                     classProbs = TRUE,
                     savePredictions = TRUE)

##### 1) Logistic Regression Model #####
set.seed(i)
lrFull <- train(x = c, y = d,
                method = "glm",
```

```
            metric = 'Kappa',
            #preProcess = c("center","scale"),
            trControl = ctrl)
lrFull
confusionMatrix(data = lrFull$pred$pred, reference = lrFull$pred$obs,
            positive = 'Phishing')

#prediction results
lrpred1 <- predict(lrFull, e)
postResample(pred = lrpred1, obs = f)
confusionMatrix(data = lrpred1, reference = f, positive = 'Phishing')


##### 2) Linear Discriminant Analysis (LDA) #####
set.seed(i)
LDAFull <- train(x = c, y = d,
             method = "lda",
             metric = "Kappa",
             #preProcess = c("center","scale"),
             trControl = ctrl)
LDAFull
confusionMatrix(data = LDAFull$pred$pred, reference = LDAFull$pred$obs,
            positive = 'Phishing')

#prediction
LDApred1 <- predict(LDAFull, e, type = 'raw')
postResample(pred = LDApred1, obs = f)
confusionMatrix(data = LDApred1, reference = f, positive = 'Phishing')


##### 3) Partial Least Squares Discriminant Analysis (PLSDA) #####
set.seed(i)
PLSFull <- train(x = c, y = d,
             method = "pls",
             metric = 'Kappa',
             tuneGrid = expand.grid(.ncomp = 1: 20 ),
             #preProc = c('center', 'scale'),
             trControl = ctrl)
PLSFull
plot(PLSFull,main = "Tuning Parameter plot for PLSDA model
    (Without Correlated variables)")
confusionMatrix(data = PLSFull$pred$pred, reference = PLSFull$pred$obs,
            positive = 'Phishing')

#prediction
PLSpred1 <- predict(PLSFull, e)
```

```
#PLSpred <- predict(PLSFull, m, type = 'prob')
postResample(pred = PLSpred1, obs = f)
confusionMatrix(data = PLSpred1, reference = f, positive = 'Phishing')


##### 4) Penalized Models #####
set.seed(i)
PENFull <- train(x = k, y = d,
                 method = "glmnet",
                 metric = 'Kappa',
                 tuneGrid = expand.grid(.alpha = c(0,.1, .2, .3),
                                        .lambda = seq(.01, .2, length = 10)),
                 preProc = c('center', 'scale'),
                 trControl = ctrl)
PENFull
plot(PENFull, main = "Tuning Parameter plot for Penalized Models
    (With Correlated variables)")
confusionMatrix(data = PENFull$pred$pred, reference = PENFull$pred$obs,
                positive = 'Phishing')

#prediction
PENpred1 <- predict(PENFull, m)
postResample(pred = PENpred1, obs = f)
confusionMatrix(data = PENpred1, reference = f, positive = 'Phishing')


################################ Non-linear Models ################################
##### 1) Regularized Discriminant Analysis(RDA) #####
set.seed(i)
RDAFull <- train(x = c, y = d,
                 method = "rda",
                 metric = "Kappa",
                 tuneGrid = expand.grid(.gamma = c(0.1, 0.25, 0.5, 0.75, 1),
                                        .lambda = seq(0, 1, length = 5)),
                 #preProcess = c("center","scale"),
                 trControl = ctrl)
RDAFull
plot(RDAFull,main = "Tuning Parameter plot for RDA Model")
confusionMatrix(data = RDAFull$pred$pred, reference = RDAFull$pred$obs,
                positive = 'Phishing')

#prediction
RDApred <- predict(RDAFull, e)
postResample(pred = RDApred, obs = f)
confusionMatrix(data = RDApred,  reference = f, positive = 'Phishing')
```

```
##### 2) Mixture Discriminant Analysis(MDA) #####
set.seed(i)
MDAFull <- train(x = c, y = d,
                 method = "mda",
                 metric = "Kappa",
                 tuneGrid = expand.grid(.subclasses = 1:4),
                 #preProcess = c("center","scale"),
                 trControl = ctrl)
MDAFull
plot(MDAFull,main = "Tuning Parameter plot for MDA Model")
confusionMatrix(data = MDAFull$pred$pred, reference = MDAFull$pred$obs,
                positive = 'Phishing')

#prediction
MDApred <- predict(MDAFull, e)
postResample(pred = MDApred, obs = f)
confusionMatrix(data = MDApred, reference = f, positive = 'Phishing')


##### 3) Neural Network #####
set.seed(i)
nnetGrid <- expand.grid(.size = 10:25, .decay =  seq(0, 1.2, length = 6))
maxSize <- max(nnetGrid$.size)
numWts <- (maxSize * (ncol(c) + 1) + (maxSize + 1)*2)

set.seed(i)
NNETFull <- train(x = c, y = d,
                  method = "nnet",
                  metric = 'Kappa',
                  tuneGrid = nnetGrid,
                  preProc = c('center', 'scale'),
                  trControl = ctrl,
                  trace = F,
                  maxit = 2000,
                  MaxNWts = numWts)
NNETFull
plot(NNETFull,main = "Tuning Parameter plot for Neural Network Model")
confusionMatrix(data = NNETFull$pred$pred, reference = NNETFull$pred$obs,
                positive = 'Phishing')

#prediction
NNETpred1 <- predict(NNETFull, e)
NNETpred <- predict(NNETFull, e, type = 'prob')
postResample(pred = NNETpred1, obs = f)
confusionMatrix(data = NNETpred1, reference = f, positive = 'Phishing')
```

```
##### 4) Support Vector Machines (SVM) #####
set.seed(i)
sigmaRangeReduced <- sigest(as.matrix(c))
SVMGrid <-expand.grid(.sigma = sigmaRangeReduced[1], .C = 2^(seq(-4, 20)))

set.seed(i)
SVMFull <- train(x = c,  y = d,
                 method = "svmRadial",
                 metric = 'Kappa',
                 tuneGrid = SVMGrid,
                 preProc = c('center', 'scale'),
                 trControl = ctrl,
                 trace = F)
SVMFull
ggplot(SVMFull)+coord_trans(x='log2')
#plot(SVMFull,main = "Tuning Parameter plot for SVM model")

confusionMatrix(data = SVMFull$pred$pred, reference = SVMFull$pred$obs,
                positive = 'Phishing')

#prediction
SVMpred1 <- predict(SVMFull, e)
SVMpred <- predict(SVMFull, e, type = 'prob')
postResample(pred = SVMpred1, obs = f)
confusionMatrix(data = SVMpred1, reference = f, positive = 'Phishing')


##### 5) Flexible Discriminant Analysis (FDA) #####
set.seed(i)
marsGrid <- expand.grid(.degree = 1:2, .nprune = 11:35)

set.seed(i)
FDAFull <- train(x = c,  y = d,
                 method = "fda",
                 metric = 'Kappa',
                 tuneGrid = marsGrid,
                 #preProcess = c("center","scale"),
                 trControl = ctrl)
FDAFull
plot(FDAFull,main ="Tuning Parameter plot for FDA Model")
confusionMatrix(data = FDAFull$pred$pred,  reference = FDAFull$pred$obs,
                positive = 'Phishing')

#prediction
```

```
FDApred1 <- predict(FDAFull, e, type = 'raw')
postResample(pred = FDApred1, obs = f)
confusionMatrix(data = FDApred1, reference = f, positive = 'Phishing')


##### 6) K Nearest Neighbors(KNN) #####
set.seed(i)
KNNFull <- train(x = c, y = d,
                 method = "knn",
                 metric = 'Kappa',
                 tuneGrid = expand.grid(.k = 1:60),
                 preProc = c('center', 'scale'),
                 trControl = ctrl)
KNNFull
plot(KNNFull,main = "Tuning Parameter plot for KNN Models")
confusionMatrix(data = KNNFull$pred$pred, reference = KNNFull$pred$obs,
                positive = 'Phishing')

#prediction
KNNpred1 <- predict(KNNFull, e)
KNNpred <- predict(KNNFull, e, type = 'prob')
postResample(pred = KNNpred1, obs = f)
confusionMatrix(data = KNNpred1, reference = f, positive = 'Phishing')


##### 7) Naive Bayes #####
set.seed(i)
NBFull <- train(x = c, y = d,
                method = "nb",
                metric = 'Kappa',
                tuneGrid = expand.grid(.fL = 2, .usekernel = T, .adjust = T),
                preProc = c('center', 'scale'),
                trControl = ctrl)
NBFull
#plot(NBFull,main = "Tuning Parameter plot for Naive Bayes Models")
confusionMatrix(data = NBFull$pred$pred,  reference = NBFull$pred$obs,
                positive = 'Phishing')

#prediction
NBpred1 <- predict(NBFull, e)
NBpred <- predict(NBFull, e, type = 'prob')
postResample(pred = NBpred1, obs = f)
confusionMatrix(data = NBpred1, reference = f, positive = 'Phishing')


#### Barplot of the models.####
```

```
best_model <- c('Logistic Regression','Penalized Model','Neural Network','KNN')
kappa_trainging <- c(0.8819,0.8386,0.9272,0.8673)
kappa_testing <- c(0.8567,0.8520,0.9476,0.9422)
df_best_models <- data.frame(best_model, kappa_trainging,kappa_testing)
names(df_best_models) <- c('Models','Training Data', 'Testing Data')
df_best_models

library(reshape2)
# reshape your data into long format
best_models <- melt(df_best_models, id=c("Models"))
# make the plot
ggplot(best_models) +  geom_bar(aes(x = Models, y = value, fill = variable),
                        stat="identity", position = "dodge", width = 0.7) +
                        scale_fill_manual("Kappa Value\n", values = c("red","blue"),
                        labels = c('Training Data', 'Testing Data')) +
                        labs(x="\nModels",y="Kappa Value\n") +  theme_bw(base_size = 14)


#### Importance variables from the Best model.####
nnet_varImp <- varImp(NNETFull, scale = FALSE)
nnet_varImp
plot(nnet_varImp, top = 10, scales = list(y = list(cex = .95)))
```

References

Huss, Nick, et al. “How Many Websites Are There in the World? [2022].” *Siteefy*, 3 Jan. 2022, https://siteefy.com/how-many-websites-are-there/.

Nsanzimana Gilbert. “Phishing Statistics and Facts for 2019–2021.” *Comparitech*, 31 Jan. 2022, https://www.comparitech.com/blog/vpn-privacy/phishing-statistics facts/.

“9 Popular Phishing Scams (Be Aware).” *BroadbandSearch.net*, https://www.broadbandsearch.net/blog/popular-email-phishing-scams.

Alkhalil, Zainab, et al. “Phishing Attacks: A Recent Comprehensive Study and a New Anatomy.” *Frontiers*, Frontiers, 1 Jan. 1AD, https://www.frontiersin.org/articles/10.3389/fcomp.2021.563060/full.

Büber, Ebubekir. “Phishing URL Detection with ML.” *Medium*, Towards Data Science, 21 May 2018, https://towardsdatascience.com/phishing-domain-detection-with-ml 5be9c99293e5.

Monctonpubliclibrary.files.wordpress.com. https://monctonpubliclibrary.files.wordpress.com/2022/01/dottodoteasy_activitybo kletpointapointfacile_livretactivites.pdf.

“Phishing Attacks (Article) | Cyber Attacks.” *Khan Academy*, Khan Academy, https://www.khanacademy.org/computing/computers-and internet/xcae6f4a7ff015e7d:online-data-security/xcae6f4a7ff015e7d:cyber attacks/a/phishing-attacks

“Phishing Statistics (Updated 2022) - 50+ Important Phishing Stats.” *Tessian*, 8 Feb. 2022, https://www.tessian.com/blog/phishing-statistics 2020/#:~:text=When%20asked%20about%20the%20impact,organizations%20we

e%20infected%20with%20ransomware.

"Sans ISC: Hunting for Phishing Sites Masquerading as Outlook Web Access Sans Site Network Current Site sans Internet Storm Center Other sans Sites Help Graduate Degree Programs Security Training Security Certification Security Awareness Training Penetration Testing Industrial Control Systems Cyber Defense Foundations DFIR Software Security Government Onsite Training Infosec Handlers Diary Blog." *SANS Internet Storm Center*, https://isc.sans.edu/diary/27974.

"What Iss Phishing: Attack Techniques & Scam Examples: Imperva." *Learning Center*, 17 June 2020, https://www.imperva.com/learn/application-security/phishing-attack scam/.

Mohammad, Rami. "Phishing Websites Data Set." *UCI Machine Learning Repository: Phishing Websites Data Set*, https://archive.ics.uci.edu/ml/datasets/Phishing+Websites.