FIT3152_Assignment2

31994695

2024-05-17

My student ID is 31994695. However, there was some unexpected happening with classifier such as every node leads to zero. The lecturer adviced me to change seeds. Therefore, I used my seeds as 319946955.

I didn't use any AI to generate any materials / content in this assignment.

Question 1

```
## Ratio of Class 1 (proportion of phishing sites to legitimate sites) : 0.291
```

There were a total of 2000 rows and 26 columns in the dataset. Among the 2000 sites, 582 were phishing sites and 1418 were legitimate sites. (Phishing sites were represented as 1, and legitimate sites as 0.) Therefore, the proportion of phishing sites is 582/2000, which corresponds to 29.1% of the total.

I could see the mean, and median values of each attributes from the data. However, I couldn't notice about the anything noteworthy in the data. I concluded that it will be available to find noteworthy data only after implementing classifier.

Question 2

```
# Question 2, pre-proessing

PD_filtered <- PD[complete.cases(PD), ]
PD_filtered$Class <- as.factor(PD_filtered$Class)

dim(PD_filtered)
```

```
## [1] 1605   26
```

First, to make accurate predictions, I excluded all columns with missing values (NA). Consequently, the shape of PD_filtered changed to 1605 rows and 26 columns from the original 2000 rows. This indicates that a total of 395 rows were removed due to the exclusion of columns with NA values.

Furthermore, I converted the Class attribute into a factor.

Question 3

I divided my data into a 70% training and 30% test set by adapting the following code. The code is from Assignment instruction, with my student ID seeds.

```
# Question 3
set.seed(319946955)
train.row = sample(1:nrow(PD_filtered), 0.7*nrow(PD_filtered))
PD.train = PD_filtered[train.row,]
PD.test = PD_filtered[-train.row,]
```

Question 4

Implementing a classification model for each techniques.

```
# Question 4

# Decision Tree
set.seed(319946955)
PD.tree = tree(Class ~., data = PD.train)

# Naive Bayes
set.seed(319946955)
PD.nav = naiveBayes(Class ~ . , data = PD.train)

set.seed(319946955)
sub <- sample(1:nrow(PD.train), 750, replace = FALSE)

# Bagging
set.seed(319946955)
PD.bag = bagging(Class ~ ., data = PD.train, mfinal = 10)

# Boosting
set.seed(319946955)
PD.boost <- boosting(Class ~ ., data = PD.train, mfinal = 10)

# Random Forest
set.seed(319946955)
PD.randomforest <- randomForest(Class ~., data=PD.train )
```

Question 5

Decision Tree confusion matrix

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   0   1
##          0 327  70
##          1  29  56
##                                           
##                Accuracy : 0.7946          
##                  95% CI : (0.7557, 0.8298)
##     No Information Rate : 0.7386          
##     P-Value [Acc > NIR] : 0.002491        
##                                           
##                   Kappa : 0.4056          
##                                           
##  Mcnemar's Test P-Value : 5.816e-05       
##                                           
##             Sensitivity : 0.9185          
##             Specificity : 0.4444          
##          Pos Pred Value : 0.8237          
##          Neg Pred Value : 0.6588          
##              Prevalence : 0.7386          
##          Detection Rate : 0.6784          
##    Detection Prevalence : 0.8237          
##       Balanced Accuracy : 0.6815          
##                                           
##        'Positive' Class : 0               
## 
```

Naive Bayes confusion matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 304  76
##          1  52  50
##
##                Accuracy : 0.7344
##                  95% CI : (0.6926, 0.7734)
##     No Information Rate : 0.7386
##     P-Value [Acc > NIR] : 0.60519
##
##                   Kappa : 0.2672
##
##  Mcnemar's Test P-Value : 0.04206
##
##             Sensitivity : 0.8539
##             Specificity : 0.3968
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.4902
##              Prevalence : 0.7386
##          Detection Rate : 0.6307
##    Detection Prevalence : 0.7884
##       Balanced Accuracy : 0.6254
##
##        'Positive' Class : 0
##
```

Bagging confusion matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 337  71
##          1  19  55
##
##                Accuracy : 0.8133
##                  95% CI : (0.7756, 0.8471)
##     No Information Rate : 0.7386
##     P-Value [Acc > NIR] : 7.163e-05
##
##                   Kappa : 0.4421
##
##  Mcnemar's Test P-Value : 7.621e-08
##
##             Sensitivity : 0.9466
##             Specificity : 0.4365
##          Pos Pred Value : 0.8260
##          Neg Pred Value : 0.7432
##              Prevalence : 0.7386
##          Detection Rate : 0.6992
##    Detection Prevalence : 0.8465
##       Balanced Accuracy : 0.6916
##
##        'Positive' Class : 0
##
```

Boosting confusion matrix

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction   0   1
##          0 315  59
##          1  41  67
##                                           
##                Accuracy : 0.7925          
##                  95% CI : (0.7535, 0.8279)
##     No Information Rate : 0.7386          
##     P-Value [Acc > NIR] : 0.003476        
##                                           
##                   Kappa : 0.4367          
##                                           
##  Mcnemar's Test P-Value : 0.089131        
##                                           
##             Sensitivity : 0.8848          
##             Specificity : 0.5317          
##          Pos Pred Value : 0.8422          
##          Neg Pred Value : 0.6204          
##              Prevalence : 0.7386          
##          Detection Rate : 0.6535          
##    Detection Prevalence : 0.7759          
##       Balanced Accuracy : 0.7083          
##                                           
##        'Positive' Class : 0               
## 
```

Random Forest confusion matrix

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 337  75
##          1  19  51
##
##                Accuracy : 0.805
##                  95% CI : (0.7667, 0.8394)
##     No Information Rate : 0.7386
##     P-Value [Acc > NIR] : 0.0003927
##
##                   Kappa : 0.4103
##
##  Mcnemar's Test P-Value : 1.405e-08
##
##             Sensitivity : 0.9466
##             Specificity : 0.4048
##          Pos Pred Value : 0.8180
##          Neg Pred Value : 0.7286
##              Prevalence : 0.7386
##          Detection Rate : 0.6992
##    Detection Prevalence : 0.8548
##       Balanced Accuracy : 0.6757
##
##        'Positive' Class : 0
##
```

```
##      Classifier Accuracy
## 1 Decision Tree   0.7946
## 2   Naive Bayes   0.7344
## 3       Bagging   0.8133
## 4      Boosting   0.7925
## 5 Random Forest   0.8050
```

Question 6

```
# Question 6
PD.tree.confidence <- PD.tree.conf_matrix$byClass
PD.tree.sensitivity <- PD.tree.confidence["Sensitivity"]

PD.pred.tree <- predict(PD.tree,PD.test,type = "vector")
PDT.pred <- prediction(PD.pred.tree[,2], PD.test$Class)
PDT.perf <- performance(PDT.pred,"tpr","fpr")

#AUC - Decision Tree
PD.tree.auc_value <- performance(PDT.pred, "auc")
PD.tree.auc <- PD.tree.auc_value@y.values[[1]]
cat("Deicions Tree AUC: ", PD.tree.auc )
```

```
## Deicions Tree AUC:  0.7499554
```

```
plot(PDT.perf, col = "lightpink")
title("Performance of Different Models")
abline(0,1)

# Naive Bayes
PD.nav.confidence <- PD.nav.conf_matrix_nb$byClass
PD.nav.sensitivity <- PD.nav.confidence["Sensitivity"]

PD.pred.bayes <- predict(PD.nav,PD.test,type = "raw")
PDN.pred <- prediction (PD.pred.bayes[,2], PD.test$Class)
PDN.perf <- performance(PDN.pred,"tpr","fpr")

#AUC - Naive Bayes
PD.nav.auc_value <- performance(PDN.pred, "auc")
PD.nav.auc <- PD.nav.auc_value@y.values[[1]]
cat("Naive Bayes AUC: ", PD.nav.auc)
```

```
## Naive Bayes AUC:  0.7089576
```

```
plot(PDN.perf, add=TRUE, col = "blueviolet")

# Bagging
PD.bag.confidence <- PD.bag.conf_matrix$byClass
PD.bag.sensitivity <- PD.bag.confidence["Sensitivity"]

PDBA.pred <- prediction(PD.bag.predict$prob[,2], PD.test$Class)
PDBA.perf <- performance(PDBA.pred,"tpr","fpr")

#AUC - Bagging
PDBA.auc_value <- performance(PDBA.pred, "auc")
PDBA.auc <- PDBA.auc_value@y.values[[1]]
cat("Bagging AUC: ", PDBA.auc )
```

```
## Bagging AUC:  0.7293785
```

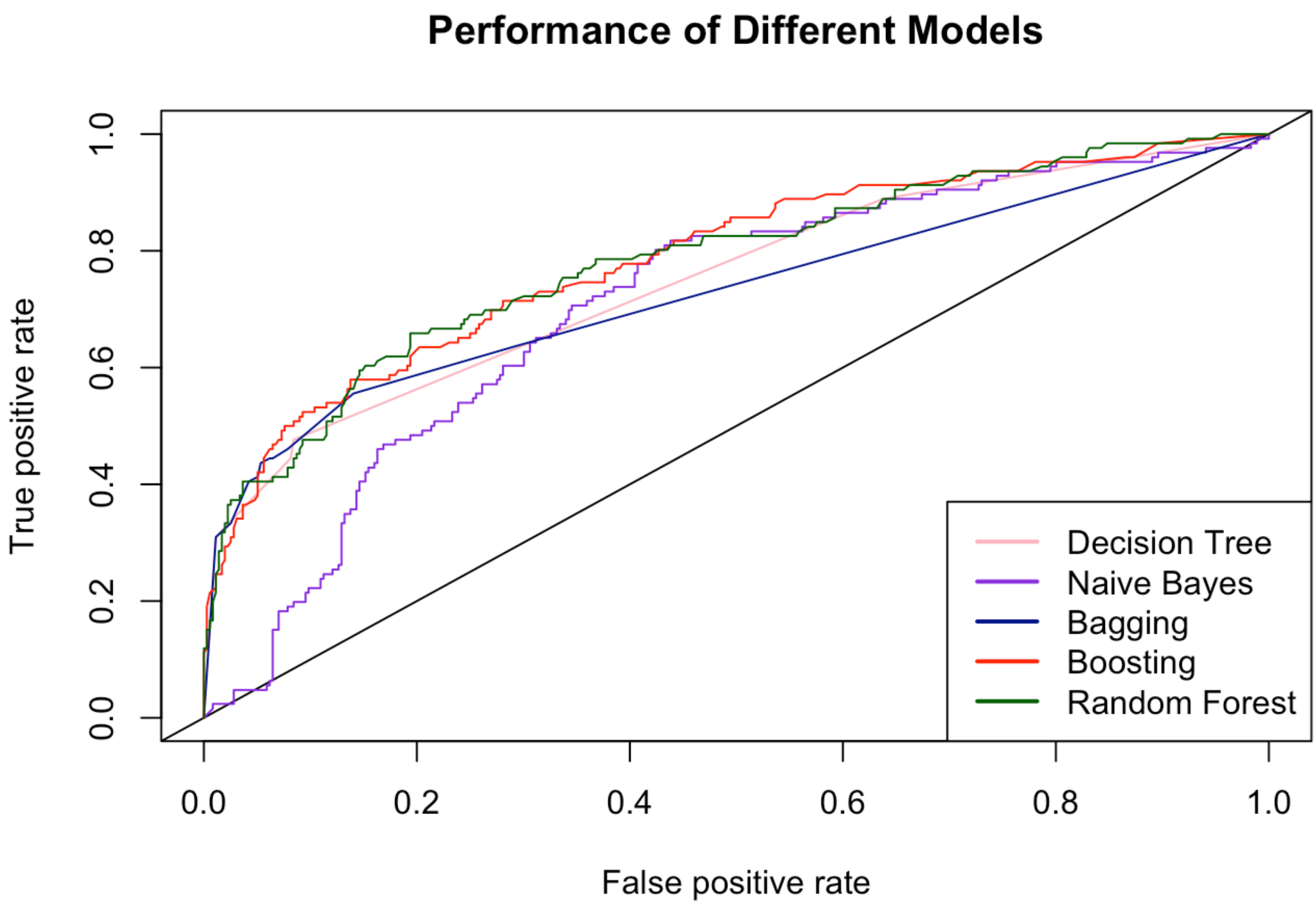
```
plot(PDBA.perf, add=TRUE, col = "darkblue")

# Boosting
PD.boost.confidence <- PD.boost.conf_matrix$byClass
PD.boost.sensitivity <- PD.boost.confidence["Sensitivity"]
PD.pred.boost <- predict.boosting(PD.boost,PD.test, mfinal = 10)
PDBO.pred <- prediction(PD.pred.boost$prob[,2], PD.test$Class)
PDBO.perf <- performance(PDBO.pred,"tpr","fpr")

#AUC - Boosting
PDBO.auc_value <- performance(PDBO.pred, "auc")
PDBO.auc <- PDBO.auc_value@y.values[[1]]
cat("Boosting AUC:" , PDBO.auc )
```

```
## Boosting AUC: 0.7866172
```

```
plot(PDBO.perf, add=TRUE, col = "red")

# Random Forest
PD.randomforest.confidence <- PD.randomforest.conf_matrix$byClass
PD.randomforest.sensitivity <- PD.randomforest.confidence["Sensitivity"]
PD.pred.rf <- predict(PD.randomforest, PD.test, type="prob")
PDRF.pred <- prediction (PD.pred.rf[,2],PD.test$Class)
PDRF.perf <- performance(PDRF.pred, "tpr","fpr")

#AUC - Random Forest
PDRF.auc_value <- performance(PDRF.pred, "auc")
PDRF.auc <- PDRF.auc_value@y.values[[1]]
cat("Random Forest AUC: ", PDRF.auc )
```

```
## Random Forest AUC:  0.783184
```

```
plot(PDRF.perf, add=TRUE, col = "darkgreen")

legend_text <- c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Random Fo
rest")
legend("bottomright", legend=legend_text, col=c("lightpink", "blueviolet", "darkbl
ue", "red", "darkgreen"), lty=1, lwd=2)
```



To calculate the confidence of predicting phishing for each case, I computed the True Positive Rate (TPR), also known as sensitivity or recall.

The results are as follows:

Decision Tree AUC: 0.7499554 Naive Bayes AUC:0.7089576 Bagging AUC:0.7293785 Boosting AUC:0.7866172 Random Forest AUC:0.783184

Decision Tree Confidence (Sensitivity): 0.9185393 Naive Bayes Confidence (Sensitivity): 0.8539326 Bagging Confidence (Sensitivity): 0.9466292 Boosting Confidence (Sensitivity): 0.8848315 Random Forest Confidence (Sensitivity): 0.9466292

Hence, the classifiers with the highest confidence are Random Forest and Bagging.

Additionally, I ploted the ROC curve with different colors for each classifier.

Question 7

```
##      Classifier Accuracy Confidence       AUC
## 1 Decision Tree   0.7946     0.9185 0.7499554
## 2   Naive Bayes   0.7344     0.8539 0.7089576
## 3       Bagging   0.8133     0.9466 0.7293785
## 4      Boosting   0.7925     0.8848 0.7866172
## 5 Random Forest   0.8050     0.9466 0.7831840
```

Through the confusion matrix of each of the 5 classifiers, it is evident that Bagging achieved the highest accuracy of 0.8133. Additionally, Bagging demonstrated the highest confidence at 0.9466 among the other classifiers. However, improving other classifiers may lead to better performance than the improvement seen in Bagging. Therefore, while Bagging can currently be considered the "best" classifier, it cannot be definitively confirmed based solely on accuracy.

Question 8

```
# Decision Tree
summary(PD.tree)
```

```
##
## Classification tree:
## tree(formula = Class ~ ., data = PD.train)
## Variables actually used in tree construction:
## [1] "A01" "A23" "A22"
## Number of terminal nodes:  6
## Residual mean deviance:  0.9047 = 1010 / 1117
## Misclassification error rate: 0.2012 = 226 / 1123
```

Variables actually used in tree construction: "A01" "A23" "A22"

```
# Bagging
PD.bag$importance
```

```
##        A01        A02        A03        A04        A05        A06        A07
## 51.7010388  0.3143095  0.0000000  0.3004152  0.0000000  0.0000000  0.0000000
##        A08        A09        A10        A11        A12        A13        A14
##  5.5686691  0.7937100  0.0000000  0.0000000  0.2823874  0.0000000  0.0000000
##        A15        A16        A17        A18        A19        A20        A21
##  0.0000000  0.0000000  0.4047226  3.6014411  0.0000000  0.0000000  0.0000000
##        A22        A23        A24        A25
##  5.8941845 30.9556961  0.1834259  0.0000000
```

most important variables : A01, 49.5053 | A23, 29.6782 | A22, 08.5829 the variables could be omitted from the data : A03, A04, A05, A07, A09, A10, A11, A13, A15, A16, A19, A20, A21, A25

```
# Boosting
PD.boost$importance
```

```
##        A01        A02        A03        A04        A05        A06        A07 
## 29.1120786  1.3596075  0.0000000  0.0000000  0.0000000  0.0000000  0.0000000 
##        A08        A09        A10        A11        A12        A13        A14 
##  8.5366567  0.5411947  0.0000000  0.3919958  3.8192572  0.0000000  0.4561732 
##        A15        A16        A17        A18        A19        A20        A21 
##  0.2410163  0.0000000  1.3512695 11.3016628  1.1475779  0.6705265  0.3106414 
##        A22        A23        A24        A25 
## 19.9729556 18.2278376  2.5595487  0.0000000
```

most important variables : A01, 31.0082 | A22, 15.8626 | A23, 14.7768 the variables could be omitted from the data : A03, A05, A07, A09, A11, A13, A16, A19

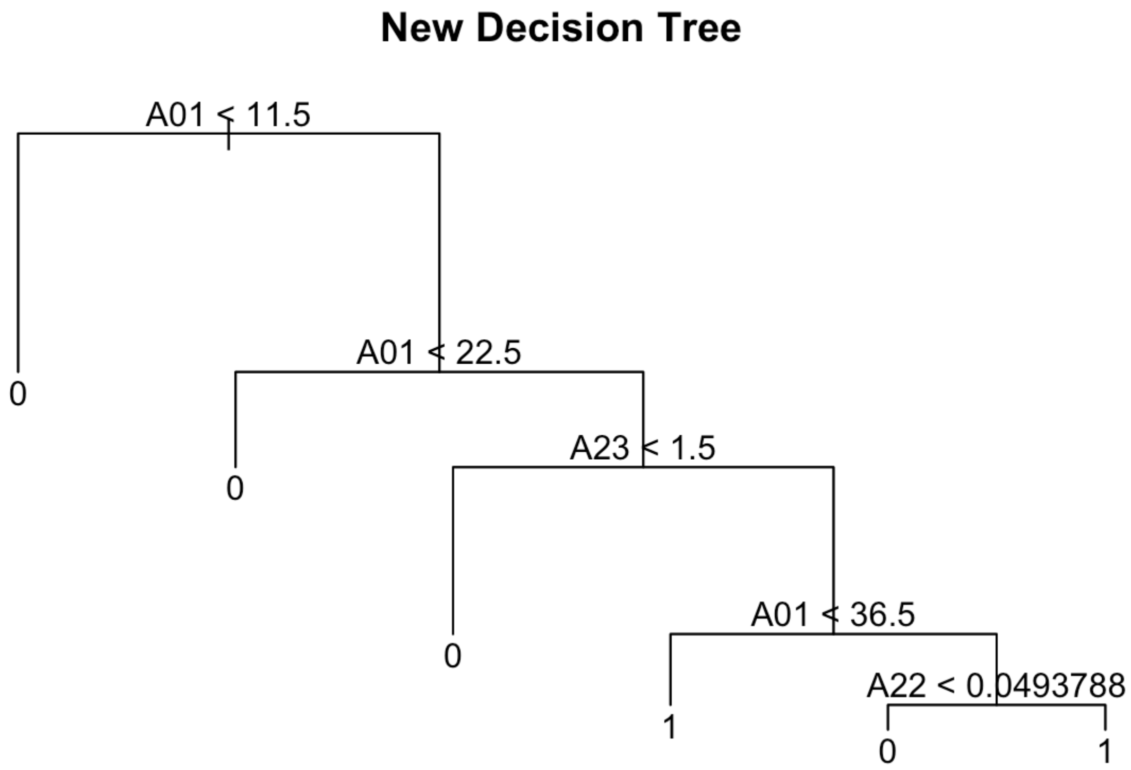
```
# Random Forest
PD.randomforest$importance
```

```
##     MeanDecreaseGini
## A01       90.6239626
## A02        9.2315474
## A03        0.0000000
## A04        7.9915678
## A05        0.6800765
## A06        5.1125548
## A07        0.1130400
## A08       28.2887290
## A09        2.1979800
## A10        1.9636385
## A11        2.0677726
## A12       22.8829649
## A13        0.1796883
## A14        9.0581670
## A15        5.4348011
## A16        3.0515957
## A17        9.8948142
## A18       54.0001083
## A19        5.3256940
## A20        7.3102724
## A21        1.5447744
## A22       63.6113975
## A23       67.8826005
## A24       23.5348557
## A25        0.2508155
```

most important variables : A01, 91.4770 | A23, 68.1270 | A22, 64.2624 the variables could be omitted from the data : A03

The criterion for determining which variables could be omitted considered those with an importance score of 0.

Question 9



```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 327  70
##          1  29  56
##
##                Accuracy : 0.7946
##                  95% CI : (0.7557, 0.8298)
##     No Information Rate : 0.7386
##     P-Value [Acc > NIR] : 0.002491
##
##                   Kappa : 0.4056
##
##  Mcnemar's Test P-Value : 5.816e-05
##
##             Sensitivity : 0.9185
##             Specificity : 0.4444
##          Pos Pred Value : 0.8237
##          Neg Pred Value : 0.6588
##              Prevalence : 0.7386
##          Detection Rate : 0.6784
##    Detection Prevalence : 0.8237
##       Balanced Accuracy : 0.6815
##
##        'Positive' Class : 0
##
```

```
## [1] "NEW Decision Tree Accuracy: 0.794605809128631"
```

I implemented a Decision Tree diagram for Question 4, considering A01, A23, and A22 attributes as they are deemed most crucial for classifying phishing or legitimate sites.

This Decision Tree achieves an accuracy of 0.7946 and a sensitivity of 0.9185 (as referenced in Question 5). Additionally, I confirmed that PD.tree.auc is 0.7499554 in Question 6.

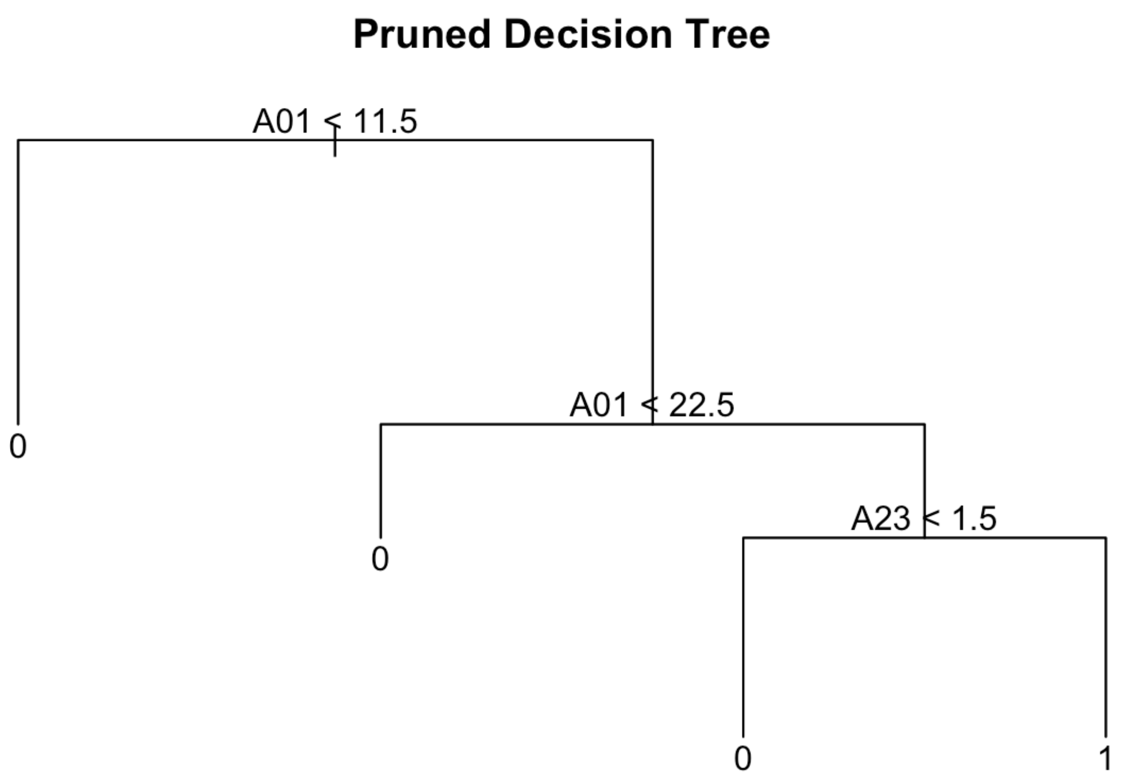
Question 10

```
# Question 10
set.seed(319946955)
test.PD.tree.fit <- cv.tree(PD.tree, FUN = prune.misclass)

test.PD.tree.fit
```

```
## $size
## [1] 6 4 1
##
## $dev
## [1] 237 236 310
##
## $k
## [1]     -Inf  1.00000 35.33333
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Size 6 Tree is same with our original decision tree, so I tried pruning with best= 4.

```
##
## Classification tree:
## snip.tree(tree = PD.tree, nodes = 15L)
## Variables actually used in tree construction:
## [1] "A01" "A23"
## Number of terminal nodes:  4
## Residual mean deviance:  0.9542 = 1068 / 1119
## Misclassification error rate: 0.203 = 228 / 1123
```



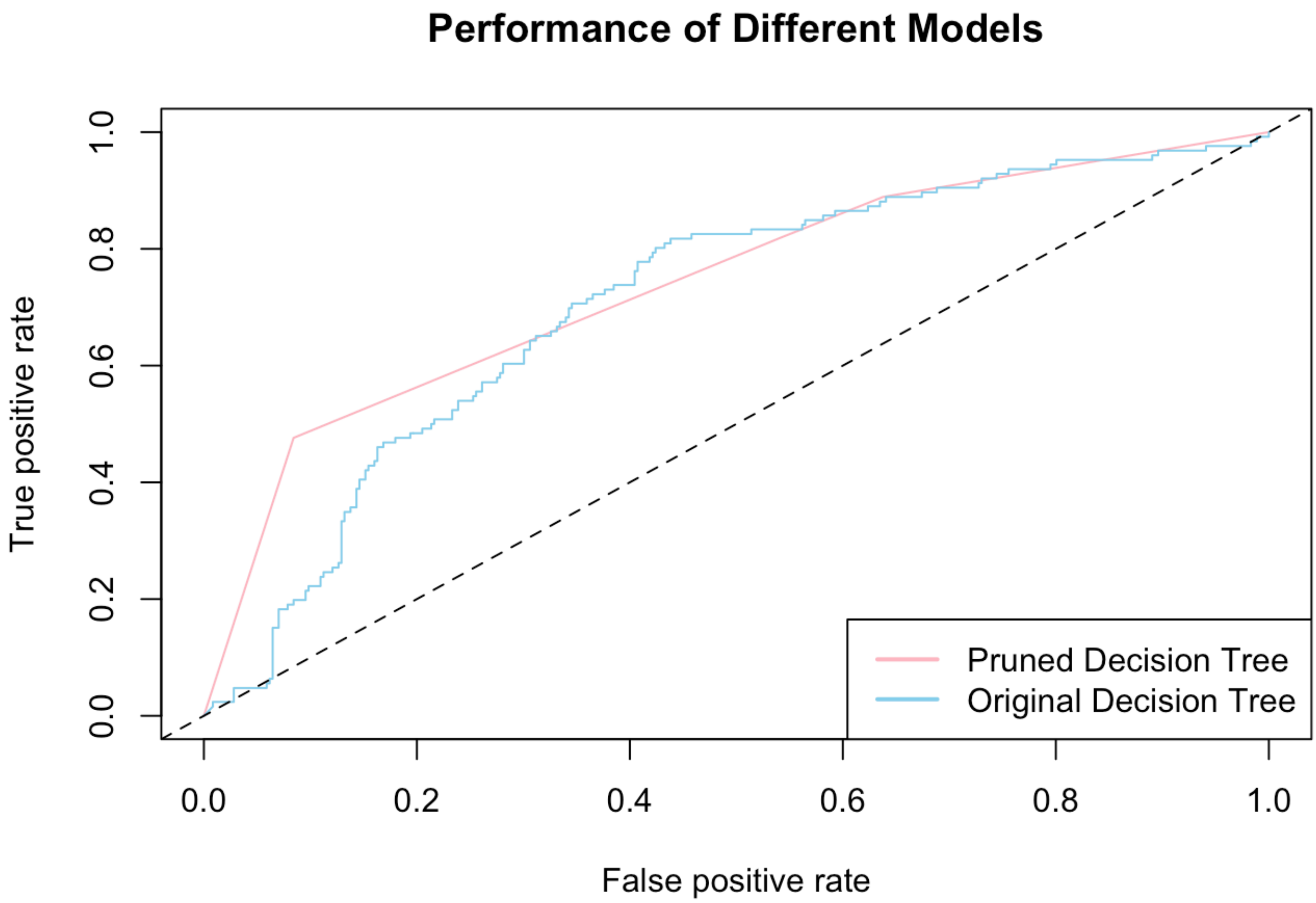

```
set.seed(319946955)

# Pruned Decision Tree prediction
PD.tree.prune.predict <- predict(PD.tree.prune, PD.test, type = "class")
PD.tree.prune.confusion_matrix <- table(actual = PD.test$Class, predicted = PD.tre
e.prune.predict)
PD.tree.prune.accuracy <- sum(diag(PD.tree.prune.confusion_matrix)) / sum(PD.tree.
prune.confusion_matrix)
print(PD.tree.prune.accuracy)
```

```
## [1] 0.8008299
```

```
# Pruned Decision Tree performance check and confusion matrix
PD.tree.prune.predict <- predict(PD.tree.prune, PD.test, type = "class")
PD.tree.prune.conf_matrix <- confusionMatrix(data = PD.tree.prune.predict, referen
ce = PD.test$Class)

# Pruned Decision Tree ROC and AUC
PD.pred.tree.prune <- predict(PD.tree.prune, PD.test, type = "vector")
PDT.prune.pred <- prediction(PD.pred.tree.prune[,2], PD.test$Class)
PDT.prune.perf <- performance(PDT.prune.pred, "tpr", "fpr")

# AUC - Pruned Decision Tree
PD.tree.prune.auc_value <- performance(PDT.prune.pred, "auc")
PD.tree.prune.auc <- PD.tree.prune.auc_value@y.values[[1]]
PD.tree.prune.auc
```

```
## [1] 0.7405141
```

```
plot(PDT.prune.perf, col = "lightpink", main = "Performance of Different Models")
plot(PDN.perf, add = TRUE, col = "skyblue")
abline(0, 1, lty = 2)
legend("bottomright", legend = c("Pruned Decision Tree", "Original Decision Tre
e"), col = c("lightpink", "skyblue"), lwd = 2)
```



Initially, I chose the decision tree, believing that if improved, it could potentially surpass the accuracy of bagging. After checking “prune” and “cross-validation,” the decision tree indeed showed some improvement, with an accuracy score of 0.8008, slightly higher than the original 0.7946. However, it was still lower than the accuracy achieved by bagging classification, which was the highest. Therefore, I opted for bagging as the best tree-based classifier and strived to further enhance it.

```
## rpart variable importance
##
##   only 20 most important variables shown (out of 25)
##
##     Overall
## A23 100.000
## A01  85.936
## A22  62.644
## A18  61.762
## A08  45.782
## A14  19.933
## A24  13.537
## A17   4.375
## A11   0.000
## A16   0.000
## A09   0.000
## A06   0.000
## A12   0.000
## A05   0.000
## A03   0.000
## A15   0.000
## A19   0.000
## A13   0.000
## A10   0.000
## A21   0.000
```

Based on the result of GridSearch, I could get the best Attributes which are A23, A01, A18, A22 and so on..

Improving Bagging

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 339  72
##          1  17  54
##
##                Accuracy : 0.8154
##                  95% CI : (0.7778, 0.849)
##     No Information Rate : 0.7386
##     P-Value [Acc > NIR] : 4.533e-05
##
##                   Kappa : 0.4433
##
##  Mcnemar's Test P-Value : 1.041e-08
##
##             Sensitivity : 0.9522
##             Specificity : 0.4286
##          Pos Pred Value : 0.8248
##          Neg Pred Value : 0.7606
##              Prevalence : 0.7386
##          Detection Rate : 0.7033
##    Detection Prevalence : 0.8527
##       Balanced Accuracy : 0.6904
##
##        'Positive' Class : 0
##
```

```
## [1] "Bagging Accuracy: 0.815352697095436"
```

When considering my decision, several factors played a crucial role. I prioritized AUC, accuracy, and sensitivity as the most important factors. These metrics were considered comprehensively when comparing decision tree, random forest, and bagging models.

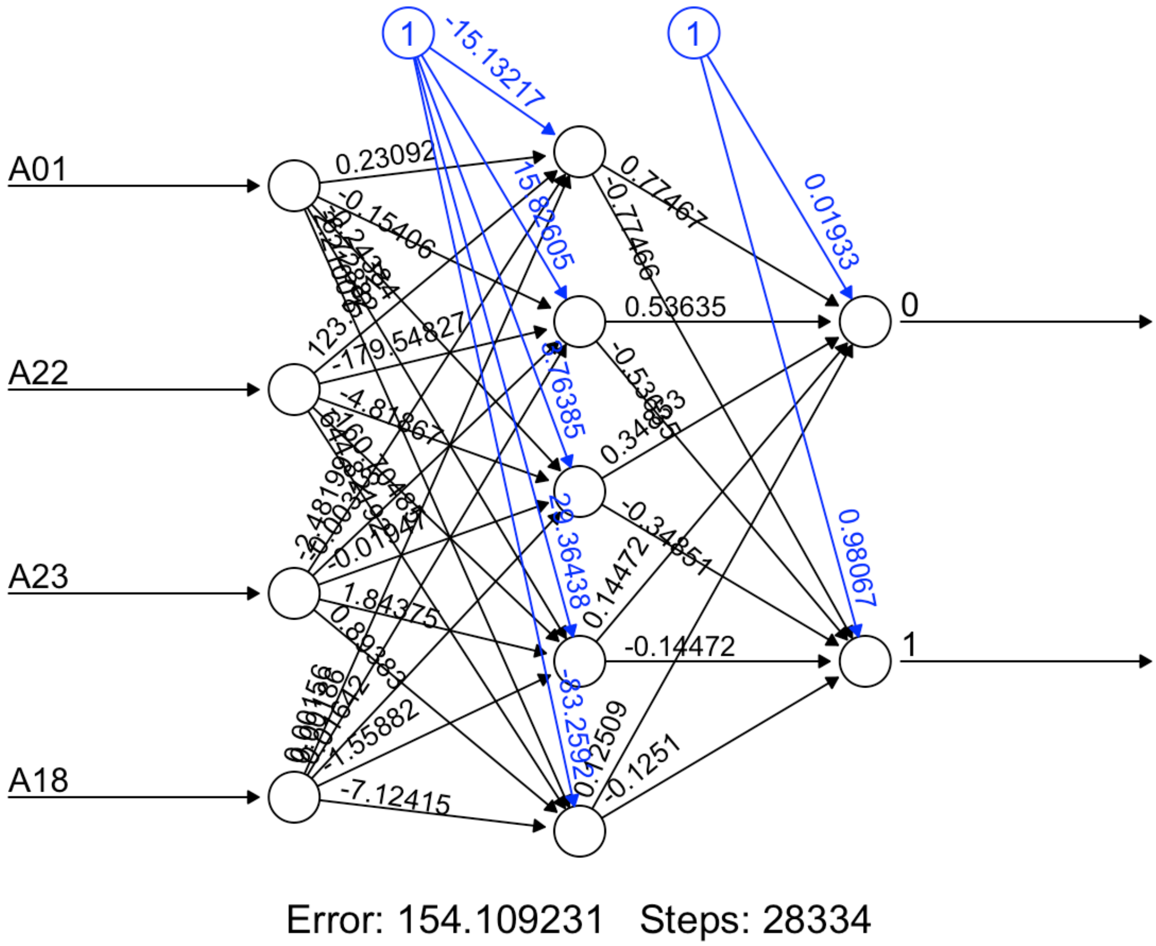
In my effort to improve the bagging model, I decided to exclude certain attributes from the dataset, namely A03, A05, A07, A09, A11, A13, A16, and A19, which were deemed unnecessary. Instead, I opted to use only columns 1, 18, 22, and 23 for bagging. This decision was influenced by examining the importance scores from the original bagging model. I noticed that many attributes had importance scores of 0, so I disregarded them. Conversely, columns 1, 18, 22, and 23 had notably high importance scores, leading me to select them for the improved bagging model.

Ultimately, my decision to focus on improving the bagging model was driven by the observation that, despite improvements to the decision tree and random forest models, they still fell short of the accuracy achieved by the original bagging model. Therefore, enhancing the original bagging model seemed to offer the closest path to achieving the best tree-based classification.

Question 11

```r
# Question 11

library(neuralnet)

PD.neural.train <- PD.train[, c(1, 22, 23, 18, 26)]
PD.neural.test <- PD.test[, c(1, 22, 23, 18, 26)]

pttrain = as.data.frame(PD.neural.train)

set.seed(319946955)
trial <- neuralnet(Class ~., pttrain, hidden = 5, threshold = 0.05)
```

```r
plot(trial, rep = 'best')
```




```
##          predicted
## observed   0   1
##         0 331  25
##         1  71  55
```

```
## ANN accuracy :  0.8008299
```

```
##       Classifier Accuracy
## 1 Decision Tree   0.7946
## 2   Naive Bayes   0.7344
## 3       Bagging   0.8133
## 4      Boosting   0.7925
## 5 Random Forest   0.8050
## 6           ANN   0.8008
```

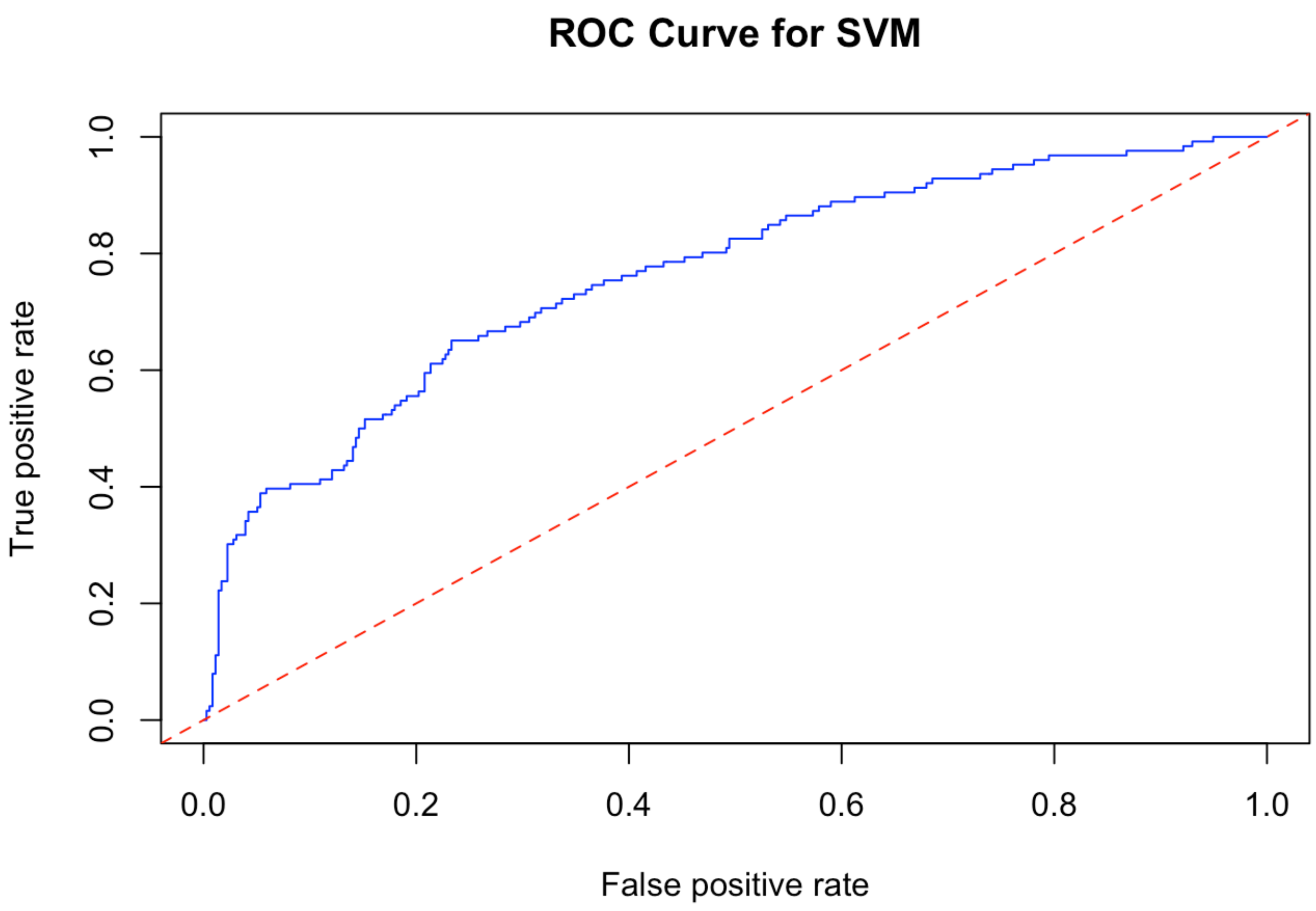
I chose the attributes that is highly related with output, which are A01, A22, A23 and A18. And I implemented ANN with hidden = 5. The accuracy of ANN was 0.8008 which is consider high. However, Our original baggin model's accuracy was still remained the highest which is 0.8133.

Question 12

```
library(e1071)
library(ROCR)

# SVM training
PD.svm <- svm(Class ~ ., data = PD.train, kernel = "linear", probability = TRUE)

# SVM predict
PD.svm.predict <- predict(PD.svm, PD.test, probability = TRUE)

# probability
PD.svm.prob <- attr(PD.svm.predict, "probabilities")

# progression for ROC in SVM
PDSVM.pred <- ROCR::prediction(PD.svm.prob[,2], PD.test$Class)
PDSVM.perf <- ROCR::performance(PDSVM.pred, "tpr", "fpr")

#AUC - SVM
PDSVM.auc_value <- performance(PDSVM.pred, "auc")
PDSVM.auc <- PDSVM.auc_value@y.values[[1]]
PDSVM.auc
```

```
## [1] 0.7622392
```

```
# ROC visualization
plot(PDSVM.perf, col = "blue", main = "ROC Curve for SVM")
abline(a = 0, b = 1, lty = 2, col = "red")
```




```
classifier_table2
```

```
##      Classifier Accuracy
## 1 Decision Tree   0.7946
## 2   Naive Bayes   0.7344
## 3       Bagging   0.8133
## 4      Boosting   0.7925
## 5 Random Forest   0.8050
## 6           ANN   0.8008
## 7           SVM   0.7822
```

I used SVM model for new classifier to the data and tested the performance in the same way as for previous models. (SVM was not covered in the course.) I used e1071 and ROCR library.

I was able to understand SVM in detail through (https://www.datacamp.com/tutorial/support-vector-machines-r (https://www.datacamp.com/tutorial/support-vector-machines-r)).

SVM stands for Support Vector Machine, which is a type of machine learning algorithm used for pattern recognition. It is a supervised learning model that, based on a dataset, creates a linear classification model to determine which category new data belongs to. The classification model uses an algorithm to find the boundary with the largest margin.

Additionally, we achieved an accuracy of 0.7822 and an AUC value of 0.7622392 as a result.

Appendix

```
library(tree)
library(e1071)
library(ROCR)
library(rpart)
library(rgl)
library(plyr)
library(adabag)
library(randomForest)
library(caret)
library(ggplot2)
#library(neuralnet)

rm(list = ls())
Phish <- read.csv("PhishingData.csv")
set.seed(319946955) # Your Student ID is the random seed
L <- as.data.frame(c(1:50))
L <- L[sample(nrow(L), 10, replace = FALSE),]
Phish <- Phish[(Phish$A01 %in% L),]
PD <- Phish[sample(nrow(Phish), 2000, replace = FALSE),] # sample of 2000 rows

# Question 1

count_0 <- sum(PD$Class == 0)
count_1 <- sum(PD$Class == 1)

# checking the ratio
ratio_0 <- count_0 / length(PD$Class)
ratio_1 <- count_1 / length(PD$Class)
cat("Ratio of Class 1 (proportion of phishing sites to legitimate sites) :", ratio
_1, "\n")

# making bar plot
barplot(c(ratio_0, ratio_1), names.arg = c("Class 0", "Class 1"), ylim = c(0,1),
        main = "Ratio of Class 0 and Class 1", ylab = "Ratio", col = c("skyblue",
"salmon"))

# Question 2, pre-proessing

PD_filtered <- PD[complete.cases(PD), ]
PD_filtered$Class <- as.factor(PD_filtered$Class)

dim(PD_filtered)



# Question 3
```

```
set.seed(319946955)
train.row = sample(1:nrow(PD_filtered), 0.7*nrow(PD_filtered))
PD.train = PD_filtered[train.row,]
PD.test = PD_filtered[-train.row,]


# Question 4

# Decision Tree
set.seed(319946955)
PD.tree = tree(Class ~., data = PD.train)

# Naive Bayes
set.seed(319946955)
PD.nav = naiveBayes(Class ~ . , data = PD.train)

set.seed(319946955)
sub <- sample(1:nrow(PD.train), 750, replace = FALSE)

# Bagging
set.seed(319946955)
PD.bag = bagging(Class ~ ., data = PD.train, mfinal = 10)

# Boosting
set.seed(319946955)
PD.boost <- boosting(Class ~ ., data = PD.train, mfinal = 10)

# Random Forest
set.seed(319946955)
PD.randomforest <- randomForest(Class ~., data=PD.train )


# Question 5

# Decision Tree
PD.tree.predict <- predict(PD.tree, PD.test, type = "class")
PD.tree.conf_matrix <- confusionMatrix(data = PD.tree.predict, reference = PD.test
$Class)

PD.tree.conf_matrix

PD.tree.accuracy <- PD.tree.conf_matrix$overall["Accuracy"]
# Naive Bayes
PD.nav.predict <- predict(PD.nav, PD.test)
PD.nav.conf_matrix_nb <- confusionMatrix(data = PD.nav.predict, reference = PD.tes
t$Class)

PD.nav.conf_matrix_nb

PD.nav.accuracy <- PD.nav.conf_matrix_nb$overall["Accuracy"]
```

```
# Bagging
PD.bag.predict <- predict.bagging(PD.bag, newdata = PD.test)
PD.bag.predict_factor <- factor(PD.bag.predict$class, levels = levels(PD.test$Clas
s))
PD.bag.conf_matrix <- confusionMatrix(data = PD.bag.predict_factor, reference = P
D.test$Class)

PD.bag.conf_matrix

PD.bag.accuracy <- PD.bag.conf_matrix$overall["Accuracy"]
# Boosting
PD.boost.predict <- predict.boosting(PD.boost, newdata = PD.test)
PD.boost.predict_factor <- factor(PD.boost.predict$class, levels = levels(PD.test$
Class))
PD.boost.conf_matrix <- confusionMatrix(data = PD.boost.predict_factor, reference
= PD.test$Class)
PD.boost.accuracy <- PD.boost.conf_matrix$overall["Accuracy"]

PD.boost.conf_matrix
# Random forest
PD.randomforest.predict <- predict(PD.randomforest,PD.test)
PD.randomforest.conf_matrix <- confusionMatrix(data = PD.randomforest.predict, ref
erence = PD.test$Class)

PD.randomforest.conf_matrix

PD.randomforest.accuracy <- PD.randomforest.conf_matrix$overall["Accuracy"]
classifiers_name <- c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Rand
om Forest")
accuracy_of_classifiers <- c(PD.tree.accuracy,
                             PD.nav.accuracy,
                             PD.bag.accuracy,
                             PD.boost.accuracy,
                             PD.randomforest.accuracy)
accuracy_of_classifiers_rounded <- round(accuracy_of_classifiers, 4)
classifier_table <- data.frame(Classifier = classifiers_name, Accuracy = accuracy_
of_classifiers_rounded)

classifier_table
classifier_table_copy = classifier_table

# Question 7


classifiers_name <- c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Rand
om Forest")
accuracy_of_classifiers <- c(PD.tree.accuracy,
                             PD.nav.accuracy,
                             PD.bag.accuracy,
```

```
                             PD.boost.accuracy,
                             PD.randomforest.accuracy)
confidence_of_classifiers <- c(PD.tree.sensitivity,
                               PD.nav.sensitivity,
                               PD.bag.sensitivity,
                               PD.boost.sensitivity,
                               PD.randomforest.sensitivity)
AUC_table <- c(PD.tree.auc,
                               PD.nav.auc,
                               PDBA.auc,
                               PDBO.auc,
                               PDRF.auc)
accuracy_of_classifiers_rounded <- round(accuracy_of_classifiers, 4)
confidence_of_classifiers_rounded <- round(confidence_of_classifiers, 4)
classifier_table <- data.frame(Classifier = classifiers_name,
                               Accuracy = accuracy_of_classifiers_rounded,
                               Confidence = confidence_of_classifiers_rounded,
                               AUC = AUC_table)


classifier_table

# Question 9

PD.new.train <- PD.train[, c(1, 22, 23, 18, 26)]
PD.new.test <- PD.test[, c(1, 22, 23, 18, 26)]


set.seed(319946955)
PD.new.tree <-tree(Class ~., data = PD.new.train)

plot(PD.new.tree)
text(PD.new.tree)
title(main = "New Decision Tree")

# Decision Tree
PD.new.tree.predict <- predict(PD.new.tree, PD.new.test, type = "class")
PD.new.tree.conf_matrix <- confusionMatrix(data = PD.new.tree.predict, reference =
PD.new.test$Class)

PD.new.tree.conf_matrix

PD.new.tree.accuracy <- PD.new.tree.conf_matrix$overall["Accuracy"]
print(paste("NEW Decision Tree Accuracy:", PD.new.tree.accuracy))

# Question 10
# Question 10 - Gridsearch

# initialize the hyperparameters value
hyperparameters <- data.frame(cp = c(0.01, 0.05, 0.1, 0.2, 0.5))
```

```
# how are we going to do the train
ctrl <- trainControl(method = "cv", number = 5)

# gridsearch
model <- train(Class ~ ., data = PD.train, method = "rpart",
               trControl = ctrl, tuneGrid = hyperparameters)

importance <- varImp(model)

print(importance)

# Improving Bagging

PD.new.train <- PD.train[, c(1, 22, 23, 18, 26)]
PD.new.test <- PD.test[, c(1, 22, 23, 18, 26)]
set.seed(319946955)
PD.new.bag = bagging(Class ~ ., data = PD.new.train, mfinal = 10)

# Bagging
set.seed(319946955)
PD.new.bag.predict <- predict.bagging(PD.new.bag, newdata = PD.new.test)
PD.new.bag.predict_factor <- factor(PD.new.bag.predict$class, levels = levels(PD.n
ew.test$Class))
PD.new.bag.conf_matrix <- confusionMatrix(data = PD.new.bag.predict_factor, refere
nce = PD.new.test$Class)

PD.new.bag.conf_matrix

PD.new.bag.accuracy <- PD.new.bag.conf_matrix$overall["Accuracy"]
print(paste("Bagging Accuracy:", PD.new.bag.accuracy))

# Question 11
# Question 11

library(neuralnet)

PD.neural.train <- PD.train[, c(1, 22, 23, 18, 26)]
PD.neural.test <- PD.test[, c(1, 22, 23, 18, 26)]

pttrain = as.data.frame(PD.neural.train)

set.seed(319946955)
trial <- neuralnet(Class ~., pttrain, hidden = 5, threshold = 0.05)


plot(trial)

library(dplyr)
library(pROC)
```

```
PD.neural.predict = predict(trial, PD.neural.test)

labels <- c('0','1')

prediction_checker <- labels[max.col(PD.neural.predict)]

PD.neural.conf_matrix <- table(observed = PD.neural.test$Class, predicted = predic
tion_checker)

PD.neural.conf_matrix

# accuracy calculate
PD.neural.accuracy <- sum(diag(PD.neural.conf_matrix)) / sum(PD.neural.conf_matri
x)

cat("ANN accuracy : ", PD.neural.accuracy)

classifiers_name <- c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Rand
om Forest", "ANN")
accuracy_of_classifiers <- c(PD.tree.accuracy,
                             PD.nav.accuracy,
                             PD.bag.accuracy,
                             PD.boost.accuracy,
                             PD.randomforest.accuracy,
                             PD.neural.accuracy)
accuracy_of_classifiers_rounded <- round(accuracy_of_classifiers, 4)
classifier_table2 <- data.frame(Classifier = classifiers_name, Accuracy = accuracy
_of_classifiers_rounded)

classifier_table2

# Question 12

PD.svm <- svm(Class ~ ., PD.train, kernel = "linear")

PD.svm.predict = predict(PD.svm, PD.test)

PD.svm.conf_matrix <- table(actual = PD.test$Class, predicted = PD.svm.predict)

PD.svm.accuracy <- sum(diag(PD.svm.conf_matrix)) / sum(PD.svm.conf_matrix)

classifiers_name <- c("Decision Tree", "Naive Bayes", "Bagging", "Boosting", "Rand
om Forest", "ANN", "SVM")
accuracy_of_classifiers <- c(PD.tree.accuracy,
                             PD.nav.accuracy,
                             PD.bag.accuracy,
                             PD.boost.accuracy,
                             PD.randomforest.accuracy,
                             PD.neural.accuracy,
```

```
                                 PD.svm.accuracy)
accuracy_of_classifiers_rounded <- round(accuracy_of_classifiers, 4)
classifier_table2 <- data.frame(Classifier = classifiers_name, Accuracy = accuracy
_of_classifiers_rounded)
```