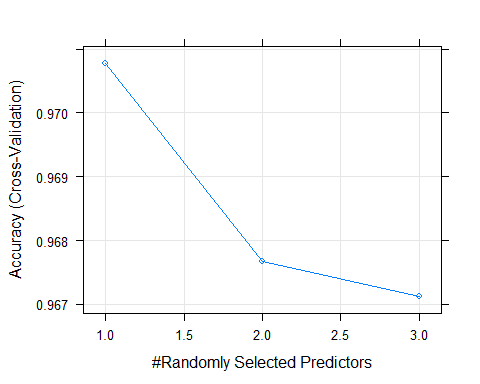
BUDA535Module5

## Problem 1 :

Using the Default data in the ISLR package use random forests, bagging, and classification trees to create a model to predict customer defaults. Use caret to do model selection (select tuning parameters) using 5 or 10 fold cross validation. Before model create a hold out sample of 10% of the total data (10% of each class) and compare the final model selected of each type. Which preforms better? What insights can you give on your models if any?

set.seed(3476)  
train\_in <- createDataPartition(Default$default, p=.90, list = FALSE,times = 1)  
train\_data <- Default[train\_in,]  
test\_data <- Default[-train\_in,]  
  
fitControl = trainControl(method="cv",number = 10)  
  
#Random Forests  
set.seed(3476)  
randomforest\_train\_model <- train(default~.,data=train\_data,trControl=fitControl, method ='rf',family="binomial",tuneGrid = expand.grid(mtry=1:3))  
plot(randomforest\_train\_model)



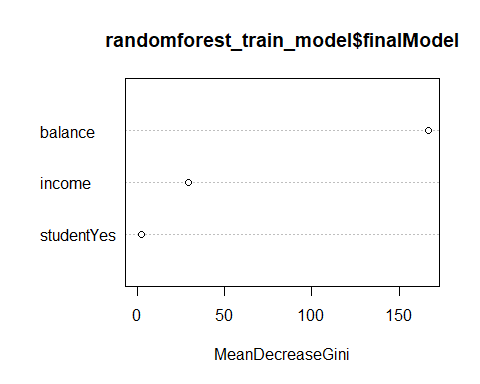
randomforest\_train\_model$finalModel

##   
## Call:  
## randomForest(x = x, y = y, mtry = param$mtry, family = "binomial")   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 1  
##   
## OOB estimate of error rate: 2.81%  
## Confusion matrix:  
## No Yes class.error  
## No 8661 40 0.004597173  
## Yes 213 87 0.710000000

varImp(randomforest\_train\_model)

## rf variable importance  
##   
## Overall  
## balance 100.0  
## income 16.4  
## studentYes 0.0

varImpPlot(randomforest\_train\_model$finalModel,type = 2)



randomforest.besttuned = displaybesttuneresults(randomforest\_train\_model)  
  
pred.randomForest <- predict(randomforest\_train\_model,test\_data)  
cm.randomForest = confusionMatrix(pred.randomForest,test\_data$default, positive = "Yes")  
cm.randomForest

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 963 27  
## Yes 3 6  
##   
## Accuracy : 0.97   
## 95% CI : (0.9574, 0.9796)  
## No Information Rate : 0.967   
## P-Value [Acc > NIR] : 0.3372   
##   
## Kappa : 0.2755   
##   
## Mcnemar's Test P-Value : 2.679e-05   
##   
## Sensitivity : 0.181818   
## Specificity : 0.996894   
## Pos Pred Value : 0.666667   
## Neg Pred Value : 0.972727   
## Prevalence : 0.033033   
## Detection Rate : 0.006006   
## Detection Prevalence : 0.009009   
## Balanced Accuracy : 0.589356   
##   
## 'Positive' Class : Yes   
##

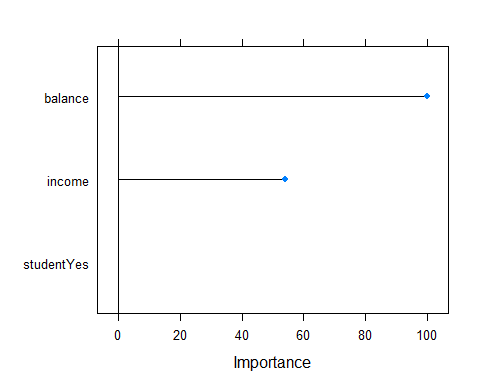
# Bagging  
set.seed(3476)  
bagging\_train\_model <- train(default~.,data=train\_data,trControl=fitControl, method="treebag")  
#bagging\_train\_model <- train(default~.,data=train\_data,trControl=fitControl, method="cforest", controls = party::cforest\_unbiased(ntree = 20), tuneGrid = data.frame(.mtry=3))  
bagging\_train\_model$finalModel

##   
## Bagging classification trees with 25 bootstrap replications

varImp(bagging\_train\_model$finalModel)

## Overall  
## balance 468.15021  
## income 272.61257  
## studentYes 42.86506

plot(varImp(bagging\_train\_model))



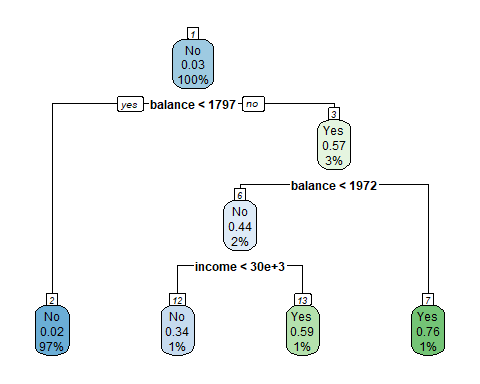
bagging.besttuned = displaybesttuneresults(bagging\_train\_model)  
  
  
pred.bagging <- predict(bagging\_train\_model,test\_data)  
cm.bagging = confusionMatrix(pred.bagging,test\_data$default,positive = "Yes")  
cm.bagging

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 960 25  
## Yes 6 8  
##   
## Accuracy : 0.969   
## 95% CI : (0.9562, 0.9788)  
## No Information Rate : 0.967   
## P-Value [Acc > NIR] : 0.405309   
##   
## Kappa : 0.3272   
##   
## Mcnemar's Test P-Value : 0.001225   
##   
## Sensitivity : 0.242424   
## Specificity : 0.993789   
## Pos Pred Value : 0.571429   
## Neg Pred Value : 0.974619   
## Prevalence : 0.033033   
## Detection Rate : 0.008008   
## Detection Prevalence : 0.014014   
## Balanced Accuracy : 0.618107   
##   
## 'Positive' Class : Yes   
##

# classification Trees  
set.seed(3476)  
#classTree\_train\_model <- train(default~.,data=train\_data,method="rpart",trControl=fitControl, tuneGrid=expand.grid(cp=0.01))  
classTree\_train\_model <- train(default~.,data=train\_data,method="rpart",trControl=fitControl, tuneLength=10)  
#classTree\_train\_model <- train(default~.,data=train\_data,method="rpart2",trControl=fitControl, tuneGrid=expand.grid(maxdepth=1:10))  
classTree\_train\_model$finalModel

## n= 9001   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 9001 300 No (0.96667037 0.03332963)   
## 2) balance< 1797.017 8738 150 No (0.98283360 0.01716640) \*  
## 3) balance>=1797.017 263 113 Yes (0.42965779 0.57034221)   
## 6) balance< 1971.915 154 67 No (0.56493506 0.43506494)   
## 12) income< 30111.74 95 32 No (0.66315789 0.33684211) \*  
## 13) income>=30111.74 59 24 Yes (0.40677966 0.59322034) \*  
## 7) balance>=1971.915 109 26 Yes (0.23853211 0.76146789) \*

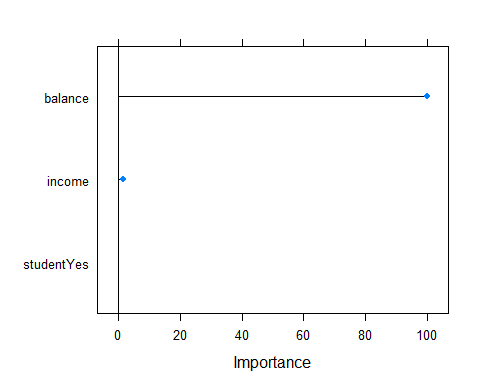
rpart.plot(classTree\_train\_model$finalModel, nn=TRUE)



varImp(classTree\_train\_model$finalModel)

## Overall  
## balance 171.583960  
## income 11.595524  
## studentYes 8.909344

plot(varImp(classTree\_train\_model))



classTree.besttuned = displaybesttuneresults(classTree\_train\_model)  
  
  
pred.classTree <- predict(classTree\_train\_model,test\_data)  
cm.classTree = confusionMatrix(pred.classTree,test\_data$default, positive = "Yes")  
cm.classTree

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 962 23  
## Yes 4 10  
##   
## Accuracy : 0.973   
## 95% CI : (0.9609, 0.9821)  
## No Information Rate : 0.967   
## P-Value [Acc > NIR] : 0.165223   
##   
## Kappa : 0.414   
##   
## Mcnemar's Test P-Value : 0.000532   
##   
## Sensitivity : 0.30303   
## Specificity : 0.99586   
## Pos Pred Value : 0.71429   
## Neg Pred Value : 0.97665   
## Prevalence : 0.03303   
## Detection Rate : 0.01001   
## Detection Prevalence : 0.01401   
## Balanced Accuracy : 0.64944   
##   
## 'Positive' Class : Yes   
##

# Accuracy/best Tuned of model on trained data  
list(title = "Best Tuned results",  
 randomforest\_pred =randomforest.besttuned,  
 bagging\_pred = bagging.besttuned,  
 classificationtree\_pred = classTree.besttuned)

## $title  
## [1] "Best Tuned results"  
##   
## $randomforest\_pred  
## mtry Accuracy Kappa AccuracySD KappaSD  
## 1 1 0.9707817 0.363796 0.003623128 0.1024161  
##   
## $bagging\_pred  
## parameter Accuracy Kappa AccuracySD KappaSD  
## 1 none 0.9670046 0.3877031 0.004405726 0.09559085  
##   
## $classificationtree\_pred  
## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.0137037 0.9720038 0.425907 0.003762532 0.1041002

# Accruacy of model on test data  
list(title = "Accuracy of prediction on test data",  
 randomforest\_pred =cm.randomForest$overall,  
 bagging\_pred = cm.bagging$overall,  
 classificationtree\_pred = cm.classTree$overall)

## $title  
## [1] "Accuracy of prediction on test data"  
##   
## $randomforest\_pred  
## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 9.699700e-01 2.754569e-01 9.574063e-01 9.796491e-01 9.669670e-01   
## AccuracyPValue McnemarPValue   
## 3.372103e-01 2.678522e-05   
##   
## $bagging\_pred  
## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 0.968968969 0.327185036 0.956241445 0.978820552 0.966966967   
## AccuracyPValue McnemarPValue   
## 0.405308805 0.001225424   
##   
## $classificationtree\_pred  
## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 0.9729729730 0.4139998696 0.9609194153 0.9821151347 0.9669669670   
## AccuracyPValue McnemarPValue   
## 0.1652232894 0.0005320055

# Other stats of Confusion Matrix  
list(title = "Other Stats of Confusion Matrix",  
 randomforest\_pred =cm.randomForest$byClass,  
 bagging\_pred = cm.bagging$byClass,  
 classificationtree\_pred = cm.classTree$byClass)

## $title  
## [1] "Other Stats of Confusion Matrix"  
##   
## $randomforest\_pred  
## Sensitivity Specificity Pos Pred Value   
## 0.181818182 0.996894410 0.666666667   
## Neg Pred Value Precision Recall   
## 0.972727273 0.666666667 0.181818182   
## F1 Prevalence Detection Rate   
## 0.285714286 0.033033033 0.006006006   
## Detection Prevalence Balanced Accuracy   
## 0.009009009 0.589356296   
##   
## $bagging\_pred  
## Sensitivity Specificity Pos Pred Value   
## 0.242424242 0.993788820 0.571428571   
## Neg Pred Value Precision Recall   
## 0.974619289 0.571428571 0.242424242   
## F1 Prevalence Detection Rate   
## 0.340425532 0.033033033 0.008008008   
## Detection Prevalence Balanced Accuracy   
## 0.014014014 0.618106531   
##   
## $classificationtree\_pred  
## Sensitivity Specificity Pos Pred Value   
## 0.30303030 0.99585921 0.71428571   
## Neg Pred Value Precision Recall   
## 0.97664975 0.71428571 0.30303030   
## F1 Prevalence Detection Rate   
## 0.42553191 0.03303303 0.01001001   
## Detection Prevalence Balanced Accuracy   
## 0.01401401 0.64944476

# Prediction Eval  
list(title = "Prediction Eval",  
 rf\_model = cm.randomForest$table %>% prop.table() %>% round(3),  
 bagging\_model = cm.bagging$table %>% prop.table() %>% round(3),  
 classtree\_model = cm.classTree$table %>% prop.table() %>% round(3))

## $title  
## [1] "Prediction Eval"  
##   
## $rf\_model  
## Reference  
## Prediction No Yes  
## No 0.964 0.027  
## Yes 0.003 0.006  
##   
## $bagging\_model  
## Reference  
## Prediction No Yes  
## No 0.961 0.025  
## Yes 0.006 0.008  
##   
## $classtree\_model  
## Reference  
## Prediction No Yes  
## No 0.963 0.023  
## Yes 0.004 0.010

### Interpretation :

As per the output of the final model selected for random forest method, number of trees trained are 500 (default) and mtry used is 1 as accuracy (97.07%) is high among all three mtry. The number of variables sampled at each split is 1 (mtry : tuning parameter) which is considered optimal in this case. The error rate of model is 2.81% which is good. As per the class error, true negatives prediction accuracy is better than true positives prediction accuracy. Looking at the very important variables, balance is considered to have greater impact on response. Variables imp plot indicates same : the average decrease in node impurity which is from splits over that variable is high. Predicting the final model on test data, accuracy of model is 97% with very less sensitivity of 18.18% which means prediction of true positives accuracy is less. In test data, actually, there are 3% of defaulters while model predicted less than 1% on test data. In random forest method, only a subset of features are selected at random out of the total and the best split feature from the subset is used to split each node in a tree.

As per the output of the final model selected for bagging method, accuracy is 96.8% on train data. No tuning parameters for treebag method. Looking at imp variables, again, balance variable have more impact on response default. Predicting the fit model on test data, 96.9 % of model accuracy with almost similar to accuracy on trained data. Again, sensitivity is just 24.24% which is prediction of true positivies accuracy is less. All features are considered for splitting a node in this method.

As per the output of the final model selected for classification tree method, cp=0.013 is used and model accuracy is 97.24%. Each node in this visual display gives the estimate at each node as well as the percentage of train data observations that are placed in that node. Based on the tree and the varImp method results, again, balance is one of the most effective varaible for partitioning the nodes of the tree. Predicting the fit model on test data, classification accuracy is 97.3% with 30.30% sensitivity. Looking at the confusion matrix data, 1% of the observations are predicted as defaulters (true positives).

**Comparisons / Which performs better? :**

Comparing the accuracy of test data and train data, they are almost same for each of these three models which is good.

Looking at the classification accuracy (on test data) of all three models, classification trees method have the highest accuracy of 97.29%. Agaim, Classification trees method have the highest sensitivity(true positives) of 30.30% which means 30.30% of observations are predicted to be true positives. Other two methods have very low sensitivity. All models have almost same specificity which predicts observations to be true negatives. p-value of classificcation trees is 0.16 is lowest than other 2 models. At level of .1, model shows dependency between dependent and predictor variables.

Looking at the prediction eval metric, 96% of the observations are predicted as true negatives by all three methods. Furthermore, classification trees method predicted 1.5% of the obersations as true postivies while other 2 method predicted less than 1% of the observations as true positives.

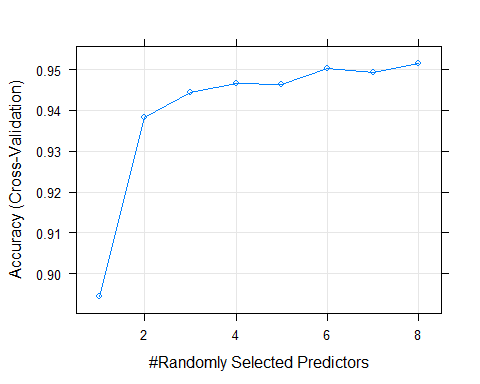
It is seen in the classification tree plot, defaulters are splitted by balance and income in different range. At the top parent node, it is overall probability of defaulters. 3% of customers are defaulters. Node splits with balance < 1797. If yes, then you go down to left child node (#2). 97% are with balance < 1797 with defaulter probability of 2%. If no, then you go down to right child node(#3). 3% are with balance > 1797 with no-defaulter probability of 57%.You keep on going like that to understand what features impact the likelihood of customer defaulted.

Looking at the stats, classification trees method performs better for this data set.

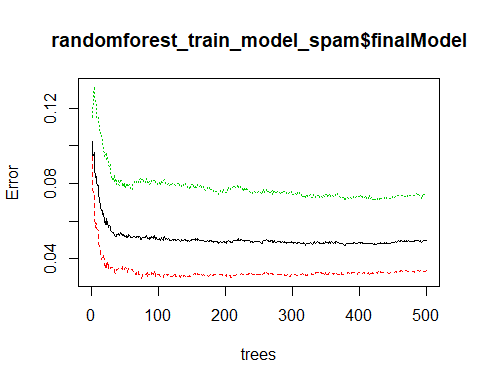
## Problem 2 :

The attached .csv file (spam.csv has 4600 emails of which about 1800 are considered spam (spam variable). The rest of the variables are indicators of the presence of 54 keywords or character. For instance word\_address is an indicator if the word address is present or not, char\_semicolon is an indicator asking if the character semicolon is present or not. It also contains variables for counts of capital letters, such as the total count, longest continuous block of capital letters. Use this information on the words in the email to create a classifier for spam prediction using random forest, bagging and classification trees. Use caret to do model selection. Before fitting hold out 200 observations to use as a validation set, where you can compare the best three models. Discuss any insights you can derive from your models or classification rates.

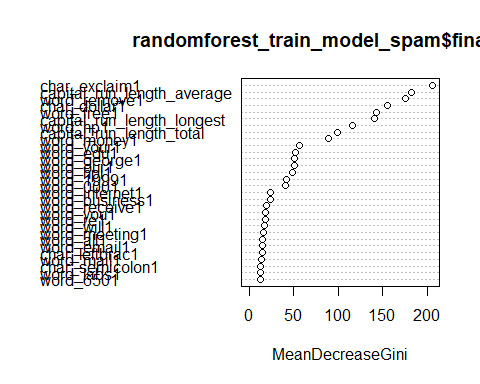
#Convert columns with length 2 i.e. indicator columns with 0 and 1 to factors  
column\_names <- sapply(spam, function(column) length(unique(column)) < 3)  
spam[,column\_names] <- lapply(spam[,column\_names], factor)  
  
#summary(spam)  
  
set.seed(4252020)  
train\_in1 <- sample(seq\_len(nrow(spam)), size =4401)  
  
train\_data1 <- spam[train\_in1,]  
test\_data1 <- spam[-train\_in1,]  
  
fitControl <- trainControl(method="cv", number = 10)  
  
# Random forest  
set.seed(4252020)  
randomforest\_train\_model\_spam <- train(spam ~.,data=train\_data1,method="rf",trControl=fitControl, tuneGrid=expand.grid(mtry=1:8))  
#randomforest\_train\_model\_spam  
plot(randomforest\_train\_model\_spam)



plot(randomforest\_train\_model\_spam$finalModel)



varImpPlot(randomforest\_train\_model\_spam$finalModel, type=2)



randomforest.besttuned.spam = displaybesttuneresults(randomforest\_train\_model\_spam)  
  
pred.randomForest.spam <- predict(randomforest\_train\_model\_spam, test\_data1, type="raw")  
cm.randomForest.spam = confusionMatrix(pred.randomForest.spam,test\_data1$spam, positive = "1")  
cm.randomForest.spam

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 121 2  
## 1 4 73  
##   
## Accuracy : 0.97   
## 95% CI : (0.9358, 0.9889)  
## No Information Rate : 0.625   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9363   
##   
## Mcnemar's Test P-Value : 0.6831   
##   
## Sensitivity : 0.9733   
## Specificity : 0.9680   
## Pos Pred Value : 0.9481   
## Neg Pred Value : 0.9837   
## Prevalence : 0.3750   
## Detection Rate : 0.3650   
## Detection Prevalence : 0.3850   
## Balanced Accuracy : 0.9707   
##   
## 'Positive' Class : 1   
##

# ROC curve  
pred.randomForest.spam.roc <- predict(randomforest\_train\_model\_spam, test\_data1, type="prob")  
rocCurve.randomforest.spam <- roc(test\_data1$spam, pred.randomForest.spam.roc[,"1"])

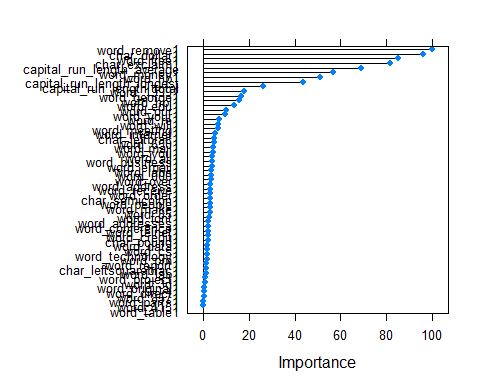
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases

# Bagging  
set.seed(4252020)  
bagging\_train\_model\_spam <- train(spam~., data=train\_data1, trControl = fitControl, method="treebag")  
#bagging\_train\_model\_spam <- train(spam~.,data=train\_data1,trControl=fitControl, method="cforest", controls = party::cforest\_unbiased(ntree = 20), tuneGrid = data.frame(.mtry=8))  
bagging\_train\_model\_spam$finalModel

##   
## Bagging classification trees with 25 bootstrap replications

plot(varImp(bagging\_train\_model\_spam))



bagging.besttuned.spam = displaybesttuneresults(bagging\_train\_model\_spam)  
#varImp(bagging\_train\_model\_spam$finalModel)  
  
pred.bagging.spam <- predict(bagging\_train\_model\_spam, test\_data1)  
cm.bagging.spam = confusionMatrix(pred.bagging.spam ,test\_data1$spam, positive = "1")  
cm.bagging.spam

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 119 2  
## 1 6 73  
##   
## Accuracy : 0.96   
## 95% CI : (0.9227, 0.9826)  
## No Information Rate : 0.625   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.9156   
##   
## Mcnemar's Test P-Value : 0.2888   
##   
## Sensitivity : 0.9733   
## Specificity : 0.9520   
## Pos Pred Value : 0.9241   
## Neg Pred Value : 0.9835   
## Prevalence : 0.3750   
## Detection Rate : 0.3650   
## Detection Prevalence : 0.3950   
## Balanced Accuracy : 0.9627   
##   
## 'Positive' Class : 1   
##

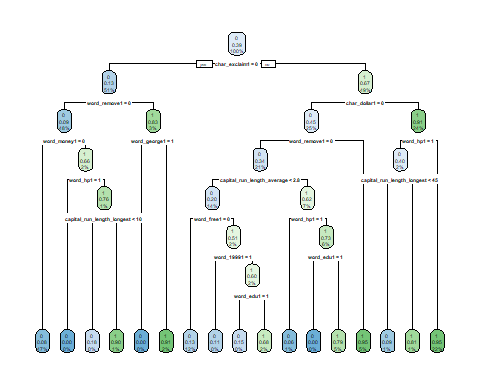
# ROC curve  
pred.bagging.spam.roc <- predict(bagging\_train\_model\_spam, test\_data1, type="prob")  
rocCurve.bagging.spam <- roc(test\_data1$spam, pred.bagging.spam.roc[,"1"])

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

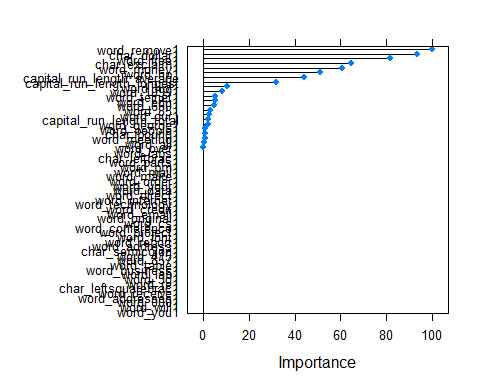
# Classification trees  
set.seed(4252020)  
#classTree\_train\_model\_spam <- train(spam~., data=train\_data1, trControl = fitControl, method="rpart",tuneGrid=expand.grid(cp=0.01))  
classTree\_train\_model\_spam <- train(spam~., data=train\_data1, trControl = fitControl, method="rpart",tuneLength=10)  
classTree\_train\_model\_spam$finalModel

## n= 4401   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 4401 1738 0 (0.60508975 0.39491025)   
## 2) char\_exclaim1< 0.5 2240 292 0 (0.86964286 0.13035714)   
## 4) word\_remove1< 0.5 2129 200 0 (0.90605918 0.09394082)   
## 8) word\_money1< 0.5 2061 155 0 (0.92479379 0.07520621) \*  
## 9) word\_money1>=0.5 68 23 1 (0.33823529 0.66176471)   
## 18) word\_hp1>=0.5 9 0 0 (1.00000000 0.00000000) \*  
## 19) word\_hp1< 0.5 59 14 1 (0.23728814 0.76271186)   
## 38) capital\_run\_length\_longest< 9.5 11 2 0 (0.81818182 0.18181818) \*  
## 39) capital\_run\_length\_longest>=9.5 48 5 1 (0.10416667 0.89583333) \*  
## 5) word\_remove1>=0.5 111 19 1 (0.17117117 0.82882883)   
## 10) word\_george1>=0.5 10 0 0 (1.00000000 0.00000000) \*  
## 11) word\_george1< 0.5 101 9 1 (0.08910891 0.91089109) \*  
## 3) char\_exclaim1>=0.5 2161 715 1 (0.33086534 0.66913466)   
## 6) char\_dollar1< 0.5 1122 503 0 (0.55169340 0.44830660)   
## 12) word\_remove1< 0.5 917 308 0 (0.66412214 0.33587786)   
## 24) capital\_run\_length\_average< 2.7545 616 121 0 (0.80357143 0.19642857)   
## 48) word\_free1< 0.5 507 65 0 (0.87179487 0.12820513) \*  
## 49) word\_free1>=0.5 109 53 1 (0.48623853 0.51376147)   
## 98) word\_19991>=0.5 19 2 0 (0.89473684 0.10526316) \*  
## 99) word\_19991< 0.5 90 36 1 (0.40000000 0.60000000)   
## 198) word\_edu1>=0.5 13 2 0 (0.84615385 0.15384615) \*  
## 199) word\_edu1< 0.5 77 25 1 (0.32467532 0.67532468) \*  
## 25) capital\_run\_length\_average>=2.7545 301 114 1 (0.37873754 0.62126246)   
## 50) word\_hp1>=0.5 48 3 0 (0.93750000 0.06250000) \*  
## 51) word\_hp1< 0.5 253 69 1 (0.27272727 0.72727273)   
## 102) word\_edu1>=0.5 19 0 0 (1.00000000 0.00000000) \*  
## 103) word\_edu1< 0.5 234 50 1 (0.21367521 0.78632479) \*  
## 13) word\_remove1>=0.5 205 10 1 (0.04878049 0.95121951) \*  
## 7) char\_dollar1>=0.5 1039 96 1 (0.09239654 0.90760346)   
## 14) word\_hp1>=0.5 82 33 0 (0.59756098 0.40243902)   
## 28) capital\_run\_length\_longest< 44.5 46 4 0 (0.91304348 0.08695652) \*  
## 29) capital\_run\_length\_longest>=44.5 36 7 1 (0.19444444 0.80555556) \*  
## 15) word\_hp1< 0.5 957 47 1 (0.04911181 0.95088819) \*

rpart.plot(classTree\_train\_model\_spam$finalModel)



#varImp(classTree\_train\_model\_spam$finalModel)  
plot(varImp(classTree\_train\_model\_spam))



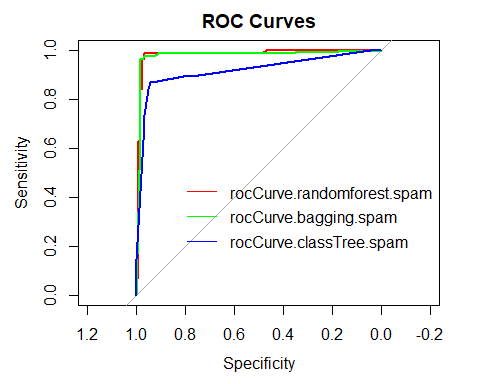
classTree.besttuned.spam = displaybesttuneresults(classTree\_train\_model\_spam)  
  
  
pred.classTree.spam <- predict(classTree\_train\_model\_spam, test\_data1)  
cm.classTree.spam = confusionMatrix(pred.classTree.spam ,test\_data1$spam, positive = "1")  
cm.classTree.spam

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 118 10  
## 1 7 65  
##   
## Accuracy : 0.915   
## 95% CI : (0.8674, 0.9497)  
## No Information Rate : 0.625   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8172   
##   
## Mcnemar's Test P-Value : 0.6276   
##   
## Sensitivity : 0.8667   
## Specificity : 0.9440   
## Pos Pred Value : 0.9028   
## Neg Pred Value : 0.9219   
## Prevalence : 0.3750   
## Detection Rate : 0.3250   
## Detection Prevalence : 0.3600   
## Balanced Accuracy : 0.9053   
##   
## 'Positive' Class : 1   
##

# ROC curve  
pred.classTree.spam.roc <- predict(classTree\_train\_model\_spam, test\_data1, type="prob")  
rocCurve.classTree.spam <- roc(test\_data1$spam, pred.classTree.spam.roc[,"1"])

## Setting levels: control = 0, case = 1  
## Setting direction: controls < cases

# ROc Curve  
plot(rocCurve.randomforest.spam, col="red", main="ROC Curves")  
plot(rocCurve.bagging.spam, add=TRUE,col="green")  
plot(rocCurve.classTree.spam, add=TRUE,col="blue")  
legend("bottomright", c("rocCurve.randomforest.spam", "rocCurve.bagging.spam","rocCurve.classTree.spam") ,col = c("red","green","blue"), inset=c(0,0.15), lty=1, bty="n")



# Accuracy and best tuned of trained data  
list(title = "Accuracy and best tuned of trained data",  
 randomforest\_pred =randomforest.besttuned.spam,  
 bagging\_pred = bagging.besttuned.spam,  
 classificationtree\_pred = classTree.besttuned.spam)

## $title  
## [1] "Accuracy and best tuned of trained data"  
##   
## $randomforest\_pred  
## mtry Accuracy Kappa AccuracySD KappaSD  
## 1 8 0.9516052 0.8982334 0.01185781 0.02506979  
##   
## $bagging\_pred  
## parameter Accuracy Kappa AccuracySD KappaSD  
## 1 none 0.9404714 0.8750259 0.01227258 0.02599067  
##   
## $classificationtree\_pred  
## cp Accuracy Kappa AccuracySD KappaSD  
## 1 0.003452244 0.907977 0.805867 0.01245588 0.02663136

# Accruacy of prediction on test data  
list(title = "Accuracy of prediction on test data",  
 randomforest\_pred =cm.randomForest.spam$overall,  
 bagging\_pred = cm.bagging.spam$overall,  
 classificationtree\_pred = cm.classTree.spam$overall)

## $title  
## [1] "Accuracy of prediction on test data"  
##   
## $randomforest\_pred  
## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 9.700000e-01 9.363395e-01 9.358494e-01 9.889125e-01 6.250000e-01   
## AccuracyPValue McnemarPValue   
## 6.074827e-32 6.830914e-01   
##   
## $bagging\_pred  
## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 9.600000e-01 9.155673e-01 9.227080e-01 9.825752e-01 6.250000e-01   
## AccuracyPValue McnemarPValue   
## 1.489854e-29 2.888444e-01   
##   
## $classificationtree\_pred  
## Accuracy Kappa AccuracyLower AccuracyUpper AccuracyNull   
## 9.150000e-01 8.172043e-01 8.673948e-01 9.497041e-01 6.250000e-01   
## AccuracyPValue McnemarPValue   
## 5.482159e-21 6.276258e-01

# Other stats of Confusion Matrix  
list(title = "Other Stats of Confusion Matrix",  
 randomforest\_pred =cm.randomForest.spam$byClass,  
 bagging\_pred = cm.bagging.spam$byClass,  
 classificationtree\_pred = cm.classTree.spam$byClass)

## $title  
## [1] "Other Stats of Confusion Matrix"  
##   
## $randomforest\_pred  
## Sensitivity Specificity Pos Pred Value   
## 0.9733333 0.9680000 0.9480519   
## Neg Pred Value Precision Recall   
## 0.9837398 0.9480519 0.9733333   
## F1 Prevalence Detection Rate   
## 0.9605263 0.3750000 0.3650000   
## Detection Prevalence Balanced Accuracy   
## 0.3850000 0.9706667   
##   
## $bagging\_pred  
## Sensitivity Specificity Pos Pred Value   
## 0.9733333 0.9520000 0.9240506   
## Neg Pred Value Precision Recall   
## 0.9834711 0.9240506 0.9733333   
## F1 Prevalence Detection Rate   
## 0.9480519 0.3750000 0.3650000   
## Detection Prevalence Balanced Accuracy   
## 0.3950000 0.9626667   
##   
## $classificationtree\_pred  
## Sensitivity Specificity Pos Pred Value   
## 0.8666667 0.9440000 0.9027778   
## Neg Pred Value Precision Recall   
## 0.9218750 0.9027778 0.8666667   
## F1 Prevalence Detection Rate   
## 0.8843537 0.3750000 0.3250000   
## Detection Prevalence Balanced Accuracy   
## 0.3600000 0.9053333

# Prediction Eval  
list(title = "Prediction Eval",  
 rf\_model = cm.randomForest.spam$table %>% prop.table() %>% round(3),  
 bagging\_model = cm.bagging.spam$table %>% prop.table() %>% round(3),  
 classtree\_model = cm.classTree.spam$table %>% prop.table() %>% round(3))

## $title  
## [1] "Prediction Eval"  
##   
## $rf\_model  
## Reference  
## Prediction 0 1  
## 0 0.605 0.010  
## 1 0.020 0.365  
##   
## $bagging\_model  
## Reference  
## Prediction 0 1  
## 0 0.595 0.010  
## 1 0.030 0.365  
##   
## $classtree\_model  
## Reference  
## Prediction 0 1  
## 0 0.590 0.050  
## 1 0.035 0.325

### Interpretation :

As per the output of final model selected for random forest method, number of trees trained are 500(by default) and mtry (tuning parameter) is 8 which has highest accuracy of 95.16%. Error rate is 4.95% which is good. As per the class error, true negatives accuracy is better than true positives accuracy similarly visible in graph of final model. Looking at the model plot, highest accuracy is at mtry=8 which is considered optimal. Very imp plot show the variables which are significant in spliting the node of the trees. A higher descrease in Gini (inquity) means that a particular predictor plays a greater role in partitioning the data into the defined classes. Predicting on test data gives 97% of accuracy with good % of sensitivty and specificity. p-value is < 0.05 which shows great dependency between dependent and predictor variables.

As per the output of final model selected for bagging method, 25 bootstrap replications are created and accuracy is 94.05% on train data. Predicting the fit model on test data, 91.5% of model accuracy which is little less than the accuracy on trained data.

As per the output of the final model selected for classification tree method, model accuracy is 90.79% and cp = 0.0034 is selected as optimal. Each node in this visual display gives the estimate at each node as well as the percentage of train data observations that are placed in that node. Based on the tree and the varImp method results,it is visible that which variables plays imp role in dividing the nodes of the tree. Predicting the fit model on test data, classification accuracy is 91.5% which is almost same accuracy as on train data.

Selected mtry = 1:8 for RF method because tried large number of parameters but error rate is not changing much but computation time is more. As per my understanding and testing that i did, variance does not change much so better to take small number but this again on data sets. mtry =1:8 carries required informations to create the nodes of the tree.

**Camparisons :**

Looking at the classification accuracy (on test data) of all three models, random forest method have the highest accuracy of 97%. Agaim, random forest method have the highest sensitivity(true positives) of 97.33 which means 97.33% of the observations are predicted to be true positives. Other two methods have less sensitivity as compared to RF. Specificty of random forest method is higher than other two methods. RF predicts 96.8% of obervartions as true negatives. p-value of all the models is < 0.05 which rejects the null hypothesis at 95% CI and shows great dependency between dependent and predictor variables and the lowest p-value is of rf method.

Looking at the prediction eval metric, Prediction accuracy of random forest is better than other two methods. 60% of the observations are predicted as emails and 36.5% of observations are predicted as spam. About 1% of observations are predicted as spam but they are actually not spam and 2% are predicted as not spam but actually they are spam emails. If we compare the results with actual dataset, 39.12% of observations are spam and 60.88% of observations are emails. Hence, the random forest classifier is better to predict if the response is spam or not for this dataset.

Looking at the ROC curves, RF (red) and bagging(green) have more area under the curve which is good.

Based on the conparison of above stats of all three models, random forest method performs better to predict if the email is spam or not. It is seen from the importance plot that variable char\_exclaim is the most important feature followed by captial\_run\_length\_average, work\_remove, Char\_dollar and many more till point 50 which played greater role in predicting if email is spam or not.