Machine Learning | Homework 5

Liana Harutyunyan

April 30, 2019

```r
library(ggplot2)
library(caret)
```

```
## Loading required package: lattice
```

```r
library(ISLR)
library(MASS)
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(rpart.plot)
```

```
## Loading required package: rpart
```

Getting the data.

```r
german_credit <-
  read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/german.data")

colnames(german_credit) <- c("chk_acct", "duration", "credit_his", "purpose",
                             "amount", "saving_acct", "present_emp",
                             "installment_rate", "sex", "other_debtor",
                             "present_resid", "property", "age",
                             "other_install", "housing", "n_credits",
                             "job", "n_people", "telephone",
                             "foreign", "response")


ind <- (german_credit$response == 2)
german_credit$response <- rep("Negative_class", length(german_credit$response))
german_credit$response[ind] <- "Positive_class"
german_credit$response <- as.factor(german_credit$response)
```

Divide dataset into training (80%) and test sets (20%).

```r
set.seed(1)
train_index <- createDataPartition(german_credit$response, p = 0.8, list = F)
train_data <- german_credit[train_index,]
test_data <- german_credit[-train_index,]
```

1. Apply classification trees (method = "rpart" in the Caret package) on the training set with 10-fold cross-validation. Show the confusion matrices and the ROC curves for the training and test sets. Draw the corresponding decision tree (score = 30).

```r
grid = expand.grid(cp = seq(0, 0.4, by = 0.01))

train_control_1 = trainControl(method = "cv",
                               number = 10,
                               classProbs = T)

model_tree_1 = train(response ~ .,
                     data = train_data,
                     method = "rpart",
                     tuneLength = 30,
                     trControl = train_control_1,
                     tuneGrid = grid)

model_tree_1
```

```
## CART
##
## 800 samples
##  20 predictor
##   2 classes: 'Negative_class', 'Positive_class'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 720, 720, 720, 720, 720, 720, ...
## Resampling results across tuning parameters:
##
##   cp    Accuracy  Kappa
##   0.00  0.70000   0.25005688
##   0.01  0.69625   0.20343988
##   0.02  0.70250   0.19655228
##   0.03  0.70500   0.09199016
##   0.04  0.69250   0.01739927
##   0.05  0.70000   0.00000000
##   0.06  0.70000   0.00000000
##   0.07  0.70000   0.00000000
##   0.08  0.70000   0.00000000
##   0.09  0.70000   0.00000000
##   0.10  0.70000   0.00000000
##   0.11  0.70000   0.00000000
##   0.12  0.70000   0.00000000
##   0.13  0.70000   0.00000000
##   0.14  0.70000   0.00000000
```

```
##   0.15  0.70000   0.00000000
##   0.16  0.70000   0.00000000
##   0.17  0.70000   0.00000000
##   0.18  0.70000   0.00000000
##   0.19  0.70000   0.00000000
##   0.20  0.70000   0.00000000
##   0.21  0.70000   0.00000000
##   0.22  0.70000   0.00000000
##   0.23  0.70000   0.00000000
##   0.24  0.70000   0.00000000
##   0.25  0.70000   0.00000000
##   0.26  0.70000   0.00000000
##   0.27  0.70000   0.00000000
##   0.28  0.70000   0.00000000
##   0.29  0.70000   0.00000000
##   0.30  0.70000   0.00000000
##   0.31  0.70000   0.00000000
##   0.32  0.70000   0.00000000
##   0.33  0.70000   0.00000000
##   0.34  0.70000   0.00000000
##   0.35  0.70000   0.00000000
##   0.36  0.70000   0.00000000
##   0.37  0.70000   0.00000000
##   0.38  0.70000   0.00000000
##   0.39  0.70000   0.00000000
##   0.40  0.70000   0.00000000
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03.
```

Now to see the confusion matrices and ROC curves for both train and test datasets, we need to make predictions with the help of the model constructed above:

```r
pred_tree_train_1 = predict(model_tree_1, newdata = train_data, type = "raw")
pred_tree_test_1 = predict(model_tree_1, newdata = test_data, type = "raw")
```

So the confusion matrix for the Training dataset is as follows:

```r
cm_train_1 = confusionMatrix(pred_tree_train_1, data = train_data$response, positive = "Positive_class")
cm_train_1
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       Negative_class Positive_class
##   Negative_class            497             63
##   Positive_class            135            105
##
##                Accuracy : 0.7525
##                  95% CI : (0.7211, 0.7821)
##     No Information Rate : 0.79
##     P-Value [Acc > NIR] : 0.9954
##
```

```
##                   Kappa : 0.3555
##  Mcnemar's Test P-Value : 4.517e-07
##
##             Sensitivity : 0.6250
##             Specificity : 0.7864
##          Pos Pred Value : 0.4375
##          Neg Pred Value : 0.8875
##              Prevalence : 0.2100
##          Detection Rate : 0.1313
##    Detection Prevalence : 0.3000
##       Balanced Accuracy : 0.7057
##
##        'Positive' Class : Positive_class
##
```

And the one for Test dataset is:

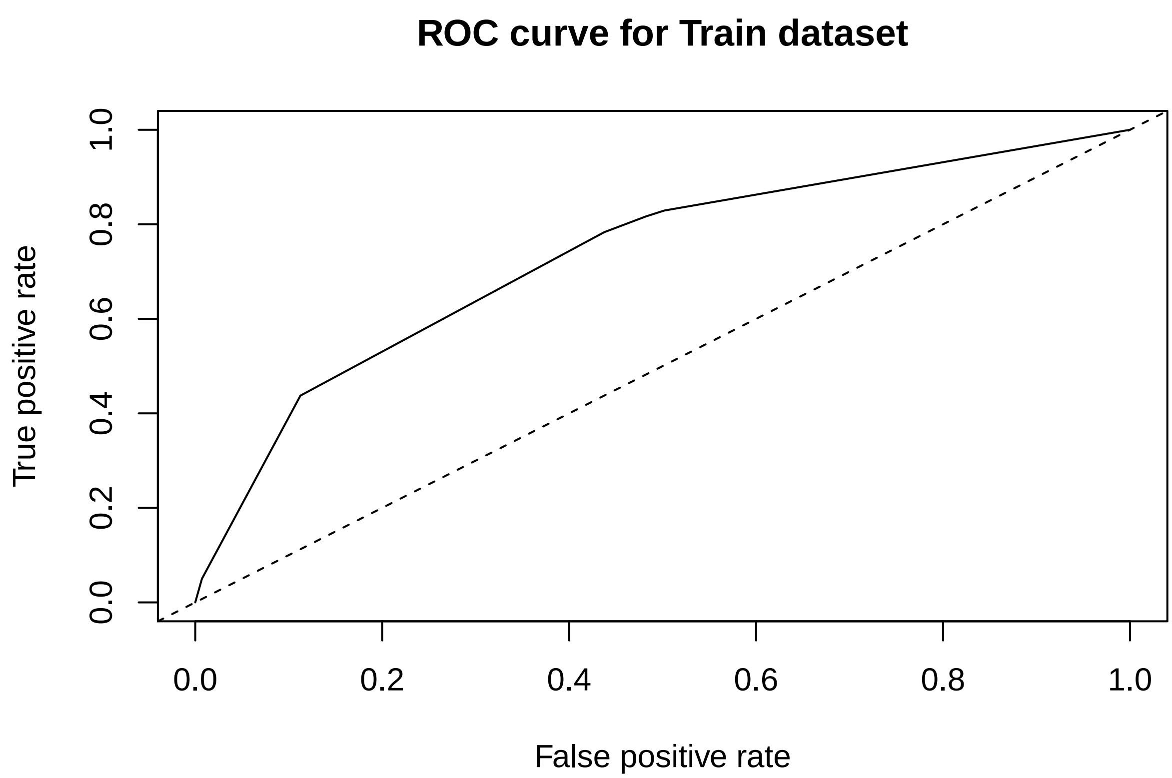
```r
cm_test_1 = confusionMatrix(pred_tree_test_1, data = test_data$response, positive = "Positive_class")
cm_test_1
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction       Negative_class Positive_class
##   Negative_class            120             20
##   Positive_class             28             32
##
##                Accuracy : 0.76
##                  95% CI : (0.6947, 0.8174)
##     No Information Rate : 0.74
##     P-Value [Acc > NIR] : 0.2894
##
##                   Kappa : 0.4059
##  Mcnemar's Test P-Value : 0.3123
##
##             Sensitivity : 0.6154
##             Specificity : 0.8108
##          Pos Pred Value : 0.5333
##          Neg Pred Value : 0.8571
##              Prevalence : 0.2600
##          Detection Rate : 0.1600
##    Detection Prevalence : 0.3000
##       Balanced Accuracy : 0.7131
##
##        'Positive' Class : Positive_class
##
```

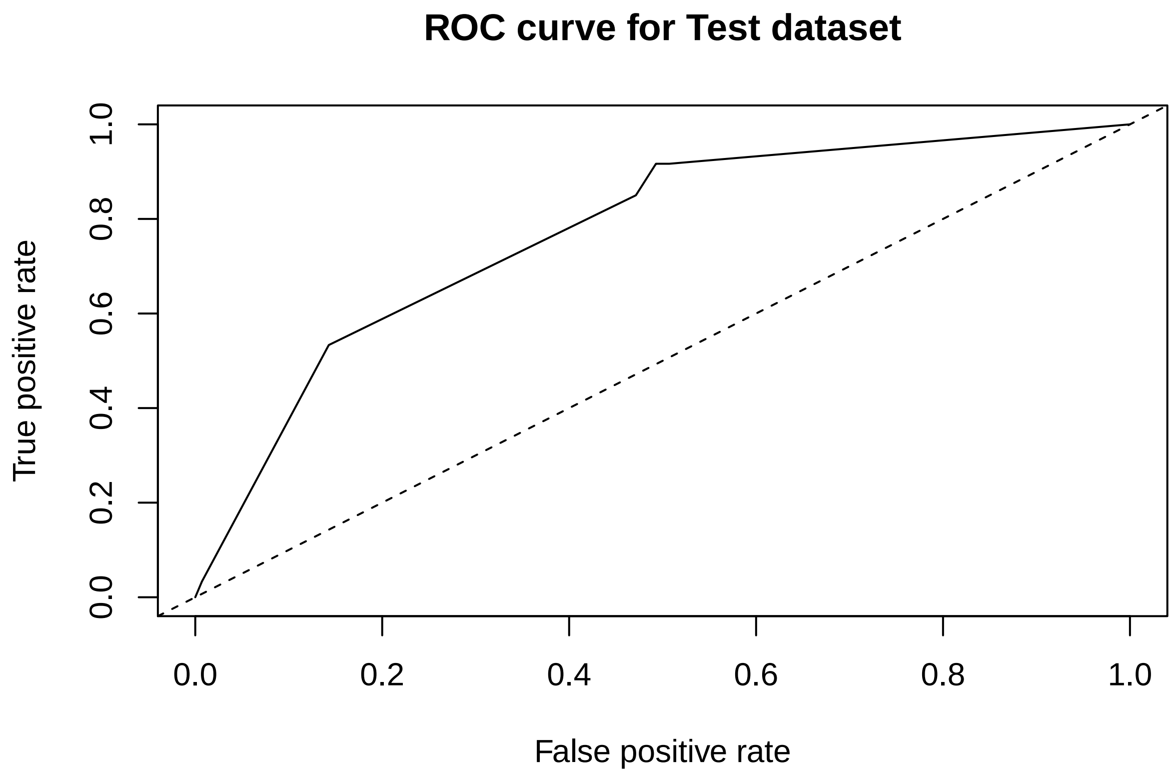
In addition, here are the ROC curves for the training and testing data respectively:

```r
pred_tree_train_prob_1 = predict(model_tree_1, newdata = train_data, type = "prob")

prediction_obj_train_1 = prediction(pred_tree_train_prob_1[,2], train_data$response)
```

```
ROC_train_1 = performance(prediction_obj_train_1, "tpr", "fpr")
plot(ROC_train_1, main = "ROC curve for Train dataset")
abline(0, 1, lty = 2)
```

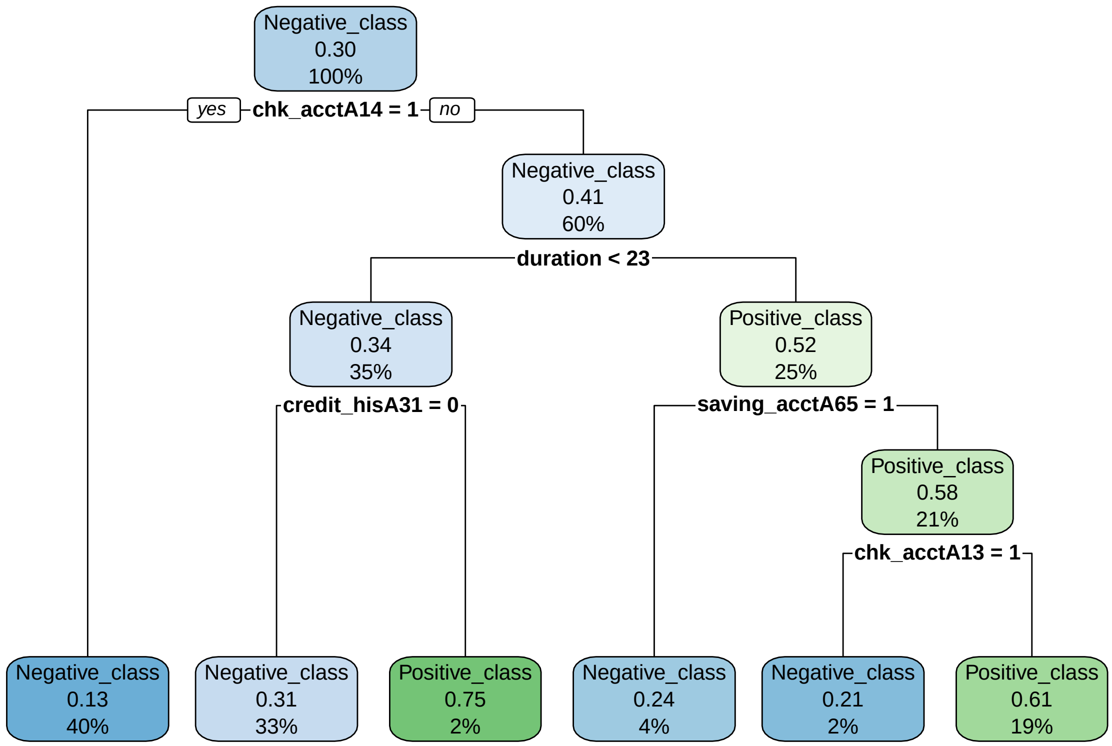



```
pred_tree_test_prob_1 = predict(model_tree_1, newdata = test_data, type = "prob")

prediction_obj_test_1 = prediction(pred_tree_test_prob_1[,2], test_data$response)

ROC_test_1 = performance(prediction_obj_test_1, "tpr", "fpr")
plot(ROC_test_1, main = "ROC curve for Test dataset")
abline(0, 1, lty = 2)
```



And here is the decision tree:

```
rpart.plot(model_tree_1$finalModel)
```



2. Apply random forests (method = "rf") on the training set. Show the confusion matrices and the ROC curves for the training and test sets. (score = 30).

```
grid2 = expand.grid(mtry = seq(2, ncol(train_data)-1, by = 1))

train_control_2 = trainControl(method = "repeatedcv",
                               number = 10,
                               repeats = 3,
                               allowParallel = TRUE)

model_tree_2 = train(response ~ .,
                     data = train_data,
                     method = "rf",
                     trControl = train_control_2,
                     tuneGrid = grid2,
                     num.threads = 3,
                     ntree = 500)
```

The confusion matrices are for training and testing sets respectively:

```
pred_tree_train_2 = predict(model_tree_2, newdata = train_data, type = "raw")
pred_tree_test_2 = predict(model_tree_2, newdata = test_data, type = "raw")
```

```r
cm_train_2 = confusionMatrix(pred_tree_train_2, data = train_data$response, positive = "Positive_class")
cm_test_2 = confusionMatrix(pred_tree_test_2, data = test_data$response, positive = "Positive_class")
```

```r
cm_train_2
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       Negative_class Positive_class
##   Negative_class            560              0
##   Positive_class              0            240
##
##                Accuracy : 1
##                  95% CI : (0.9954, 1)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
##             Sensitivity : 1.0
##             Specificity : 1.0
##          Pos Pred Value : 1.0
##          Neg Pred Value : 1.0
##              Prevalence : 0.3
##          Detection Rate : 0.3
##    Detection Prevalence : 0.3
##       Balanced Accuracy : 1.0
##
##        'Positive' Class : Positive_class
##
```

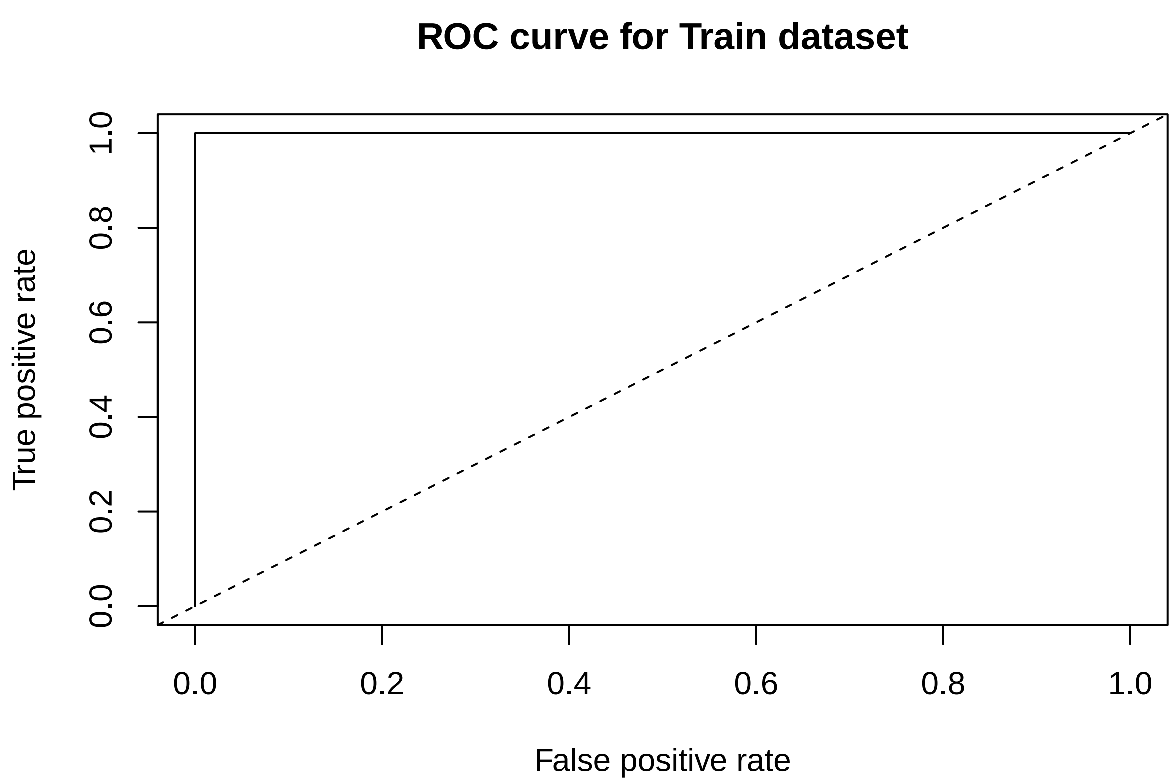
```r
cm_test_2
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction       Negative_class Positive_class
##   Negative_class            129             11
##   Positive_class             35             25
##
##                Accuracy : 0.77
##                  95% CI : (0.7054, 0.8264)
##     No Information Rate : 0.82
##     P-Value [Acc > NIR] : 0.970362
##
##                   Kappa : 0.3817
##  Mcnemar's Test P-Value : 0.000696
##
##             Sensitivity : 0.6944
##             Specificity : 0.7866
##          Pos Pred Value : 0.4167
```

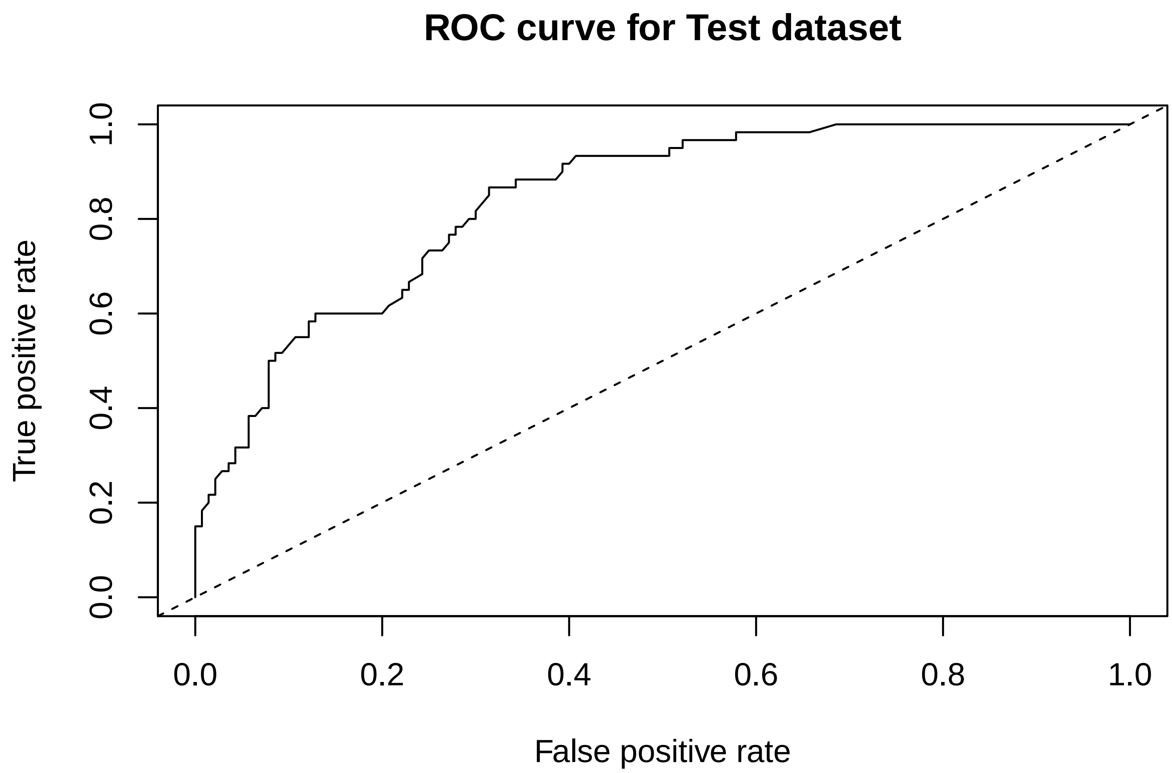
```
##          Neg Pred Value : 0.9214
##              Prevalence : 0.1800
##          Detection Rate : 0.1250
##    Detection Prevalence : 0.3000
##       Balanced Accuracy : 0.7405
##
##        'Positive' Class : Positive_class
##
```

And the ROC curves for training and testing sets respectively:

```
pred_tree_train_prob_2 = predict(model_tree_2, newdata = train_data, type = "prob")

prediction_obj_train_2 = prediction(pred_tree_train_prob_2[,2], train_data$response)

ROC_train_2 = performance(prediction_obj_train_2, "tpr", "fpr")
plot(ROC_train_2, main = "ROC curve for Train dataset")
abline(0, 1, lty = 2)
```




```
pred_tree_test_prob_2 = predict(model_tree_2, newdata = test_data, type = "prob")

prediction_obj_test_2 = prediction(pred_tree_test_prob_2[,2], test_data$response)

ROC_test_2 = performance(prediction_obj_test_2, "tpr", "fpr")
plot(ROC_test_2, main = "ROC curve for Test dataset")
abline(0, 1, lty = 2)
```



3. Apply boosting (method = "adaboost") of decision trees. Show the confusion matrices and the ROC curves for training and test sets (score = 30).

```r
set.seed(1)
train_control_3 = trainControl(method = "cv",
                               number = 5,
                               verbose = T)

model_tree_3 = train(response ~ .,
                     data = train_data,
                     method = "adaboost",
                     trControl = train_control_3,
                     tuneLength = 10)
```

```
## + Fold1: nIter= 50, method=Adaboost.M1
## - Fold1: nIter= 50, method=Adaboost.M1
## + Fold1: nIter=100, method=Adaboost.M1
## - Fold1: nIter=100, method=Adaboost.M1
## + Fold1: nIter=150, method=Adaboost.M1
## - Fold1: nIter=150, method=Adaboost.M1
## + Fold1: nIter=200, method=Adaboost.M1
## - Fold1: nIter=200, method=Adaboost.M1
## + Fold1: nIter=250, method=Adaboost.M1
## - Fold1: nIter=250, method=Adaboost.M1
```

```
## + Fold1: nIter=300, method=Adaboost.M1
## - Fold1: nIter=300, method=Adaboost.M1
## + Fold1: nIter=350, method=Adaboost.M1
## - Fold1: nIter=350, method=Adaboost.M1
## + Fold1: nIter=400, method=Adaboost.M1
## - Fold1: nIter=400, method=Adaboost.M1
## + Fold1: nIter=450, method=Adaboost.M1
## - Fold1: nIter=450, method=Adaboost.M1
## + Fold1: nIter=500, method=Adaboost.M1
## - Fold1: nIter=500, method=Adaboost.M1
## + Fold1: nIter= 50, method=Real adaboost
## - Fold1: nIter= 50, method=Real adaboost
## + Fold1: nIter=100, method=Real adaboost
## - Fold1: nIter=100, method=Real adaboost
## + Fold1: nIter=150, method=Real adaboost
## - Fold1: nIter=150, method=Real adaboost
## + Fold1: nIter=200, method=Real adaboost
## - Fold1: nIter=200, method=Real adaboost
## + Fold1: nIter=250, method=Real adaboost
## - Fold1: nIter=250, method=Real adaboost
## + Fold1: nIter=300, method=Real adaboost
## - Fold1: nIter=300, method=Real adaboost
## + Fold1: nIter=350, method=Real adaboost
## - Fold1: nIter=350, method=Real adaboost
## + Fold1: nIter=400, method=Real adaboost
## - Fold1: nIter=400, method=Real adaboost
## + Fold1: nIter=450, method=Real adaboost
## - Fold1: nIter=450, method=Real adaboost
## + Fold1: nIter=500, method=Real adaboost
## - Fold1: nIter=500, method=Real adaboost
## + Fold2: nIter= 50, method=Adaboost.M1
## - Fold2: nIter= 50, method=Adaboost.M1
## + Fold2: nIter=100, method=Adaboost.M1
## - Fold2: nIter=100, method=Adaboost.M1
## + Fold2: nIter=150, method=Adaboost.M1
## - Fold2: nIter=150, method=Adaboost.M1
## + Fold2: nIter=200, method=Adaboost.M1
## - Fold2: nIter=200, method=Adaboost.M1
## + Fold2: nIter=250, method=Adaboost.M1
## - Fold2: nIter=250, method=Adaboost.M1
## + Fold2: nIter=300, method=Adaboost.M1
## - Fold2: nIter=300, method=Adaboost.M1
## + Fold2: nIter=350, method=Adaboost.M1
## - Fold2: nIter=350, method=Adaboost.M1
## + Fold2: nIter=400, method=Adaboost.M1
## - Fold2: nIter=400, method=Adaboost.M1
## + Fold2: nIter=450, method=Adaboost.M1
## - Fold2: nIter=450, method=Adaboost.M1
## + Fold2: nIter=500, method=Adaboost.M1
## - Fold2: nIter=500, method=Adaboost.M1
## + Fold2: nIter= 50, method=Real adaboost
## - Fold2: nIter= 50, method=Real adaboost
## + Fold2: nIter=100, method=Real adaboost
## - Fold2: nIter=100, method=Real adaboost
```

```
## + Fold2: nIter=150, method=Real adaboost
## - Fold2: nIter=150, method=Real adaboost
## + Fold2: nIter=200, method=Real adaboost
## - Fold2: nIter=200, method=Real adaboost
## + Fold2: nIter=250, method=Real adaboost
## - Fold2: nIter=250, method=Real adaboost
## + Fold2: nIter=300, method=Real adaboost
## - Fold2: nIter=300, method=Real adaboost
## + Fold2: nIter=350, method=Real adaboost
## - Fold2: nIter=350, method=Real adaboost
## + Fold2: nIter=400, method=Real adaboost
## - Fold2: nIter=400, method=Real adaboost
## + Fold2: nIter=450, method=Real adaboost
## - Fold2: nIter=450, method=Real adaboost
## + Fold2: nIter=500, method=Real adaboost
## - Fold2: nIter=500, method=Real adaboost
## + Fold3: nIter= 50, method=Adaboost.M1
## - Fold3: nIter= 50, method=Adaboost.M1
## + Fold3: nIter=100, method=Adaboost.M1
## - Fold3: nIter=100, method=Adaboost.M1
## + Fold3: nIter=150, method=Adaboost.M1
## - Fold3: nIter=150, method=Adaboost.M1
## + Fold3: nIter=200, method=Adaboost.M1
## - Fold3: nIter=200, method=Adaboost.M1
## + Fold3: nIter=250, method=Adaboost.M1
## - Fold3: nIter=250, method=Adaboost.M1
## + Fold3: nIter=300, method=Adaboost.M1
## - Fold3: nIter=300, method=Adaboost.M1
## + Fold3: nIter=350, method=Adaboost.M1
## - Fold3: nIter=350, method=Adaboost.M1
## + Fold3: nIter=400, method=Adaboost.M1
## - Fold3: nIter=400, method=Adaboost.M1
## + Fold3: nIter=450, method=Adaboost.M1
## - Fold3: nIter=450, method=Adaboost.M1
## + Fold3: nIter=500, method=Adaboost.M1
## - Fold3: nIter=500, method=Adaboost.M1
## + Fold3: nIter= 50, method=Real adaboost
## - Fold3: nIter= 50, method=Real adaboost
## + Fold3: nIter=100, method=Real adaboost
## - Fold3: nIter=100, method=Real adaboost
## + Fold3: nIter=150, method=Real adaboost
## - Fold3: nIter=150, method=Real adaboost
## + Fold3: nIter=200, method=Real adaboost
## - Fold3: nIter=200, method=Real adaboost
## + Fold3: nIter=250, method=Real adaboost
## - Fold3: nIter=250, method=Real adaboost
## + Fold3: nIter=300, method=Real adaboost
## - Fold3: nIter=300, method=Real adaboost
## + Fold3: nIter=350, method=Real adaboost
## - Fold3: nIter=350, method=Real adaboost
## + Fold3: nIter=400, method=Real adaboost
## - Fold3: nIter=400, method=Real adaboost
## + Fold3: nIter=450, method=Real adaboost
## - Fold3: nIter=450, method=Real adaboost
```

```
## + Fold3: nIter=500, method=Real adaboost
## - Fold3: nIter=500, method=Real adaboost
## + Fold4: nIter= 50, method=Adaboost.M1
## - Fold4: nIter= 50, method=Adaboost.M1
## + Fold4: nIter=100, method=Adaboost.M1
## - Fold4: nIter=100, method=Adaboost.M1
## + Fold4: nIter=150, method=Adaboost.M1
## - Fold4: nIter=150, method=Adaboost.M1
## + Fold4: nIter=200, method=Adaboost.M1
## - Fold4: nIter=200, method=Adaboost.M1
## + Fold4: nIter=250, method=Adaboost.M1
## - Fold4: nIter=250, method=Adaboost.M1
## + Fold4: nIter=300, method=Adaboost.M1
## - Fold4: nIter=300, method=Adaboost.M1
## + Fold4: nIter=350, method=Adaboost.M1
## - Fold4: nIter=350, method=Adaboost.M1
## + Fold4: nIter=400, method=Adaboost.M1
## - Fold4: nIter=400, method=Adaboost.M1
## + Fold4: nIter=450, method=Adaboost.M1
## - Fold4: nIter=450, method=Adaboost.M1
## + Fold4: nIter=500, method=Adaboost.M1
## - Fold4: nIter=500, method=Adaboost.M1
## + Fold4: nIter= 50, method=Real adaboost
## - Fold4: nIter= 50, method=Real adaboost
## + Fold4: nIter=100, method=Real adaboost
## - Fold4: nIter=100, method=Real adaboost
## + Fold4: nIter=150, method=Real adaboost
## - Fold4: nIter=150, method=Real adaboost
## + Fold4: nIter=200, method=Real adaboost
## - Fold4: nIter=200, method=Real adaboost
## + Fold4: nIter=250, method=Real adaboost
## - Fold4: nIter=250, method=Real adaboost
## + Fold4: nIter=300, method=Real adaboost
## - Fold4: nIter=300, method=Real adaboost
## + Fold4: nIter=350, method=Real adaboost
## - Fold4: nIter=350, method=Real adaboost
## + Fold4: nIter=400, method=Real adaboost
## - Fold4: nIter=400, method=Real adaboost
## + Fold4: nIter=450, method=Real adaboost
## - Fold4: nIter=450, method=Real adaboost
## + Fold4: nIter=500, method=Real adaboost
## - Fold4: nIter=500, method=Real adaboost
## + Fold5: nIter= 50, method=Adaboost.M1
## - Fold5: nIter= 50, method=Adaboost.M1
## + Fold5: nIter=100, method=Adaboost.M1
## - Fold5: nIter=100, method=Adaboost.M1
## + Fold5: nIter=150, method=Adaboost.M1
## - Fold5: nIter=150, method=Adaboost.M1
## + Fold5: nIter=200, method=Adaboost.M1
## - Fold5: nIter=200, method=Adaboost.M1
## + Fold5: nIter=250, method=Adaboost.M1
## - Fold5: nIter=250, method=Adaboost.M1
## + Fold5: nIter=300, method=Adaboost.M1
## - Fold5: nIter=300, method=Adaboost.M1
```

```
## + Fold5: nIter=350, method=Adaboost.M1
## - Fold5: nIter=350, method=Adaboost.M1
## + Fold5: nIter=400, method=Adaboost.M1
## - Fold5: nIter=400, method=Adaboost.M1
## + Fold5: nIter=450, method=Adaboost.M1
## - Fold5: nIter=450, method=Adaboost.M1
## + Fold5: nIter=500, method=Adaboost.M1
## - Fold5: nIter=500, method=Adaboost.M1
## + Fold5: nIter= 50, method=Real adaboost
## - Fold5: nIter= 50, method=Real adaboost
## + Fold5: nIter=100, method=Real adaboost
## - Fold5: nIter=100, method=Real adaboost
## + Fold5: nIter=150, method=Real adaboost
## - Fold5: nIter=150, method=Real adaboost
## + Fold5: nIter=200, method=Real adaboost
## - Fold5: nIter=200, method=Real adaboost
## + Fold5: nIter=250, method=Real adaboost
## - Fold5: nIter=250, method=Real adaboost
## + Fold5: nIter=300, method=Real adaboost
## - Fold5: nIter=300, method=Real adaboost
## + Fold5: nIter=350, method=Real adaboost
## - Fold5: nIter=350, method=Real adaboost
## + Fold5: nIter=400, method=Real adaboost
## - Fold5: nIter=400, method=Real adaboost
## + Fold5: nIter=450, method=Real adaboost
## - Fold5: nIter=450, method=Real adaboost
## + Fold5: nIter=500, method=Real adaboost
## - Fold5: nIter=500, method=Real adaboost
## Aggregating results
## Selecting tuning parameters
## Fitting nIter = 500, method = Real adaboost on full training set
```

The Confusion matrices for Training and Test sets respectively:

```r
pred_tree_train_3 = predict(model_tree_3, newdata = train_data, type = "raw")
cm_train_3 = confusionMatrix(pred_tree_train_3, data = train_data$response, positive = "Positive_class")

cm_train_3
```

```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction       Negative_class Positive_class
##   Negative_class            560              0
##   Positive_class              0            240
##
##                Accuracy : 1
##                  95% CI : (0.9954, 1)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
```

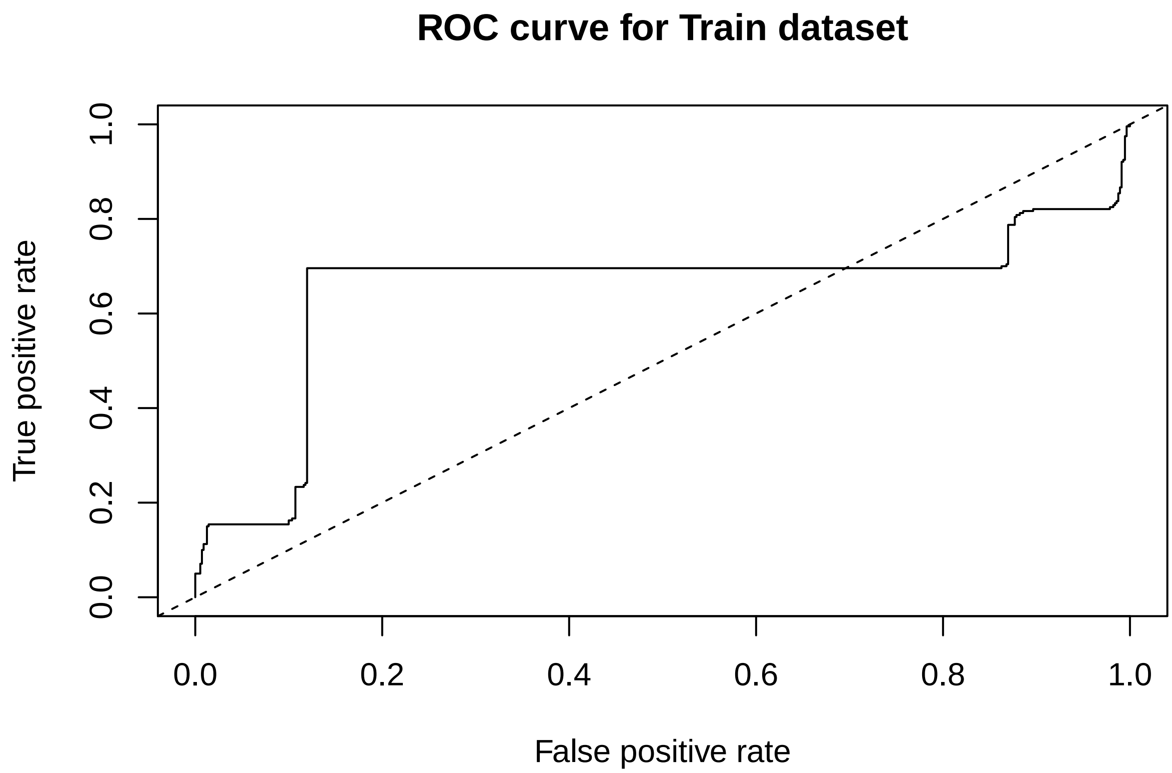
```
##
##             Sensitivity : 1.0
##             Specificity : 1.0
##          Pos Pred Value : 1.0
##          Neg Pred Value : 1.0
##              Prevalence : 0.3
##          Detection Rate : 0.3
##    Detection Prevalence : 0.3
##       Balanced Accuracy : 1.0
##
##        'Positive' Class : Positive_class
##
```

```
pred_tree_test_3 = predict(model_tree_3, newdata = test_data, type = "raw")
cm_test_3 = confusionMatrix(pred_tree_test_3, data = test_data$response, positive = "Positive_class")

cm_test_3
```

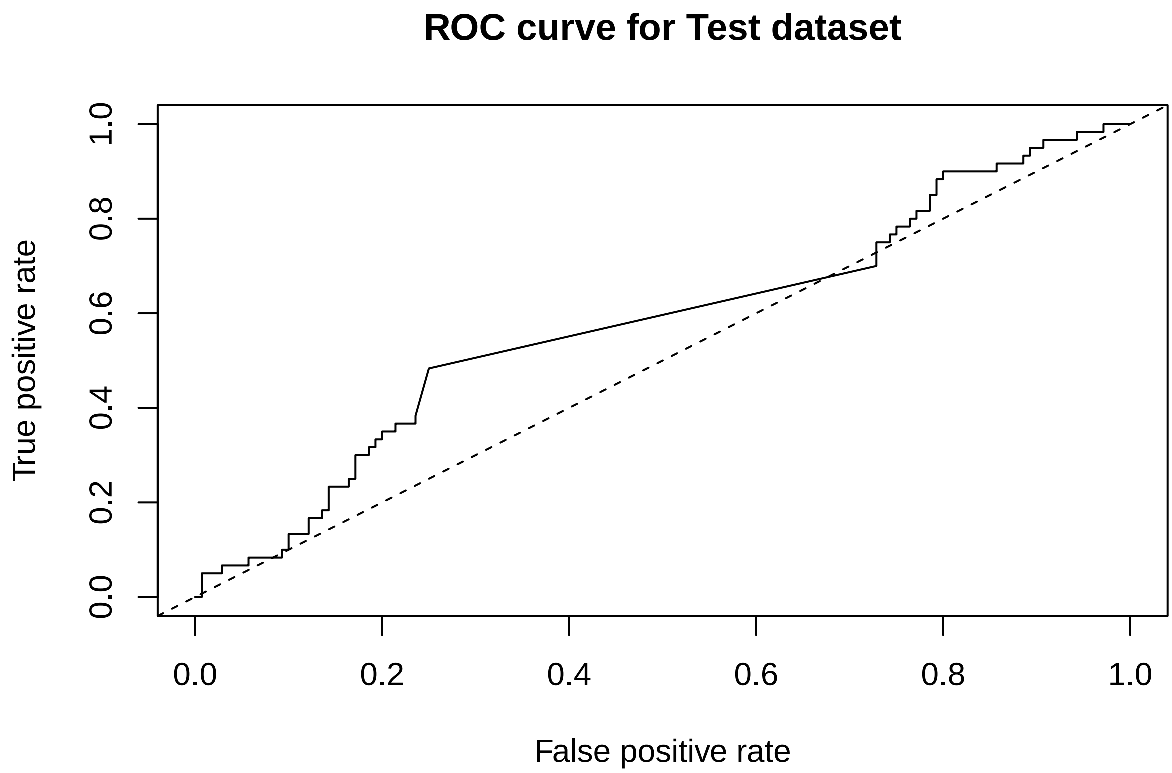
```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         Negative_class Positive_class
##   Negative_class            128             12
##   Positive_class             38             22
##
##                Accuracy : 0.75
##                  95% CI : (0.684, 0.8084)
##     No Information Rate : 0.83
##     P-Value [Acc > NIR] : 0.998463
##
##                   Kappa : 0.3207
##  Mcnemar's Test P-Value : 0.000407
##
##             Sensitivity : 0.6471
##             Specificity : 0.7711
##          Pos Pred Value : 0.3667
##          Neg Pred Value : 0.9143
##              Prevalence : 0.1700
##          Detection Rate : 0.1100
##    Detection Prevalence : 0.3000
##       Balanced Accuracy : 0.7091
##
##        'Positive' Class : Positive_class
##
```

And finally, here are the ROC curves for Training and Test set for Boosting:

```
pred_tree_train_prob_3 = predict(model_tree_3, newdata = train_data, type = "prob")

prediction_obj_train_3 = prediction(pred_tree_train_prob_3[,2], train_data$response)

ROC_train_3 = performance(prediction_obj_train_3, "tpr", "fpr")
plot(ROC_train_3, main = "ROC curve for Train dataset")
abline(0, 1, lty = 2)
```




```r
pred_tree_test_prob_3 = predict(model_tree_3, newdata = test_data, type = "prob")

prediction_obj_test_3 = prediction(pred_tree_test_prob_3[,2], test_data$response)

ROC_test_3 = performance(prediction_obj_test_3, "tpr", "fpr")
plot(ROC_test_3, main = "ROC curve for Test dataset")
abline(0, 1, lty = 2)
```



4. Compare the precisions on the test set and pick the best model for classification (score = 10).

```
prec_1 <- cm_test_1$byClass["Pos Pred Value"]
prec_2 <- cm_test_2$byClass["Pos Pred Value"]
prec_3 <- cm_test_3$byClass["Pos Pred Value"]
precisions <- c(prec_1, prec_2, prec_3)
names(precisions) <- c("Classification Trees", "Random Forests", "Boosting of Dec. Trees")

precisions
```

```
##   Classification Trees         Random Forests Boosting of Dec. Trees
##              0.5333333              0.4166667              0.3666667
```

So by comparing the precisions, the best model for classification is Classification Trees.