classification_Xinming

Xinming Liu

11/27/2020

Set A - Debate 1

```
## SetA Preprocessing
users <- read.csv("user_setA/users.csv", stringsAsFactors = FALSE)
#head(a.users)
tweets1 <- read.csv("user_setA/tweets_debate1.csv", stringsAsFactors = FALSE)
tweets2 <- read.csv("user_setA/tweets_debate2.csv", stringsAsFactors = FALSE)
tweets3 <- read.csv("user_setA/tweets_debate3.csv", stringsAsFactors = FALSE)
tweets4 <- read.csv("user_setA/tweets_debateVP.csv", stringsAsFactors = FALSE)
tweets <- rbind(tweets1, tweets2, tweets3, tweets4)
#head(tweets)
tweets<-tweets%>%
  mutate(userID=as.numeric((userID)))
data <- data.table(users, key="userID")[
  data.table(tweets, key="userID"),
  allow.cartesian=TRUE
]
data <- subset(data, party=='D' | party=='R')
data <- subset(data, state_code=="FL")
#head(data)
use <- data.frame(data$text, factor(data$party), stringsAsFactors = FALSE)
colnames(use) <- c("text", "party")
#head(use)
```

```
## Create DocumentTermMatrix
corpus <- Corpus(VectorSource(use$text))
corpus = clean_corpus(corpus)
#td.mat = TermDocumentMatrix(corpus)
dt.mat = DocumentTermMatrix(corpus)
## dt.mat is not a matrix here
```

```
## Feature words extraction (this may create NA values) due to limited memory
dt.mat.use = removeSparseTerms(dt.mat, 0.95)
## Sparsity = 0.95 (7 terms remaining) ~ 0.97 (17 terms remaining) seems acceptable
```

```
## Attach class label
alldata <- as.matrix(dt.mat.use)
alldata <- cbind(alldata, use$party)
colnames(alldata)[ncol(alldata)] <- "Class"
## Class=1 for Democrats, Class=2 for Republican
alldata <- as.data.frame(alldata)
```

```r
alldata$Class <- as.factor(alldata$Class)
levels(alldata$Class) <- c("Democrats",  "Republican")
```

Use 10-fold CV with 70% data for training, 30% data for testing

```r
## Train-Test Split
set.seed(9000)
## 70% for training, 30% for testing
TrainingDataIndex <- createDataPartition(alldata$Class, p=0.7, list = FALSE)
trainingData <- alldata[TrainingDataIndex,]
testData <- alldata[-TrainingDataIndex,]
## 10-fold CV (cannot do repeatedcv due to CPU performance)
TrainingParameters <- trainControl(method = "cv", number = 10, classProbs = TRUE, summaryFunction = two
```

Classification using kNN (too many ties)

```r
# ## Training
# fit <- train(Class ~ ., data = trainingData,
#                  method = "knn",
#                  trControl= TrainingParameters,
#                  tuneGrid = expand.grid(k = seq(1, 10, length = 10)),
#                  preProcess = c("scale","center"),
#                  na.action = na.omit
# )
# fit
# fit$bestTune
#
# ## Testing
# pred <- predict(fit, testData)
#
# ## Evaluation
# confusionMatrix(pred, testData$Class)
#
# ## Rank terms by importance
# importance <- varImp(fit, scale=FALSE)
# plot(importance)
```

Classification using SVM

```r
set.seed(9001)
## Using linear kernel

## Training
fit.lSVM <- train(Class ~ ., data = trainingData,
                  method = "svmLinear",
                  metric = "ROC",
                  trControl= TrainingParameters,
                  tuneGrid = expand.grid(C = seq(0.1, 1, length = 9)),
                  preProcess = c("scale","center"),
                  na.action = na.omit
)
```

```
## maximum number of iterations reached 0.0009052221 0.0008988652maximum number of iterations reached 0
```

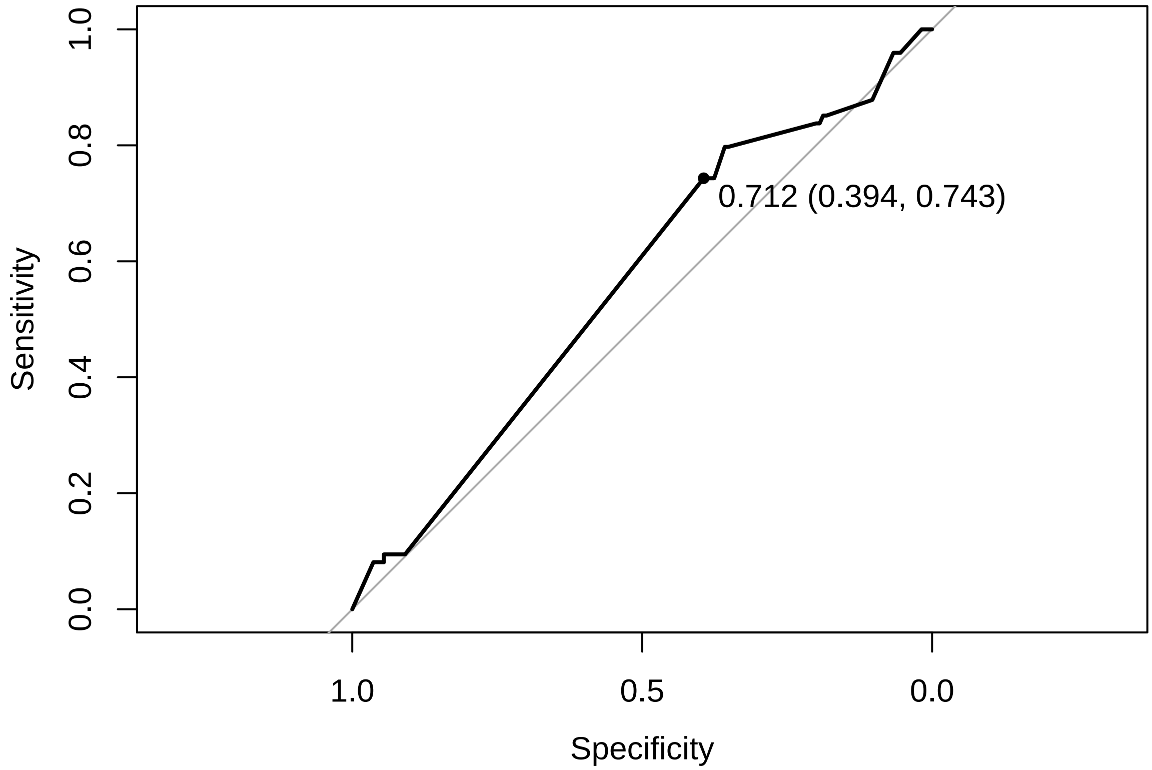
```r
fit.lSVM
```

```
## Support Vector Machines with Linear Kernel
##
## 563 samples
##   6 predictor
##   2 classes: 'Democrats', 'Republican'
##
## Pre-processing: scaled (6), centered (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 507, 507, 507, 507, 507, 506, ...
## Resampling results across tuning parameters:
##
##   C       ROC        Sens  Spec
##   0.1000  0.4957998  1     0
##   0.2125  0.4745358  1     0
##   0.3250  0.4898983  1     0
##   0.4375  0.5328424  1     0
##   0.5500  0.4858082  1     0
##   0.6625  0.4530615  1     0
##   0.7750  0.4991108  1     0
##   0.8875  0.5218775  1     0
##   1.0000  0.5133029  1     0
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.4375.
```

```r
fit.lSVM$bestTune
```

```
##        C
## 4 0.4375
```

```r
## Testing
pred.lSVM <- predict(fit.lSVM, testData, type="prob")

## Evaluation
#confusionMatrix(pred, testData$Class)
result.roc <- roc(testData$Class, pred.lSVM$Democrats)
plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")
```




```
result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accuracy
print(result.coords)
```

```
##           threshold  accuracy
## threshold 0.7122004 0.5020921
```

```
# set.seed(9002)
# ## Using polynomial kernel
#
# ## Training
# ## Due to CPU performance, cannot apply a grid to tune parameters
# fit.pSVM <- train(Class ~ ., data = trainingData,
#                   method = "svmPoly",
#                   metric = "ROC",
#                   trControl= TrainingParameters,
#                   preProcess = c("scale","center"),
#                   na.action = na.omit
# )
# fit.pSVM
# fit.pSVM$bestTune
#
# ## Testing
# pred.pSVM <- predict(fit.pSVM, testData)
#
# ## Evaluation
```

```r
# #confusionMatrix(pred, testData$Class)
# result.roc <- roc(testData$Class, pred.pSVM$Democrats)
# plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")
# result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accura
# print(result.coords)
```

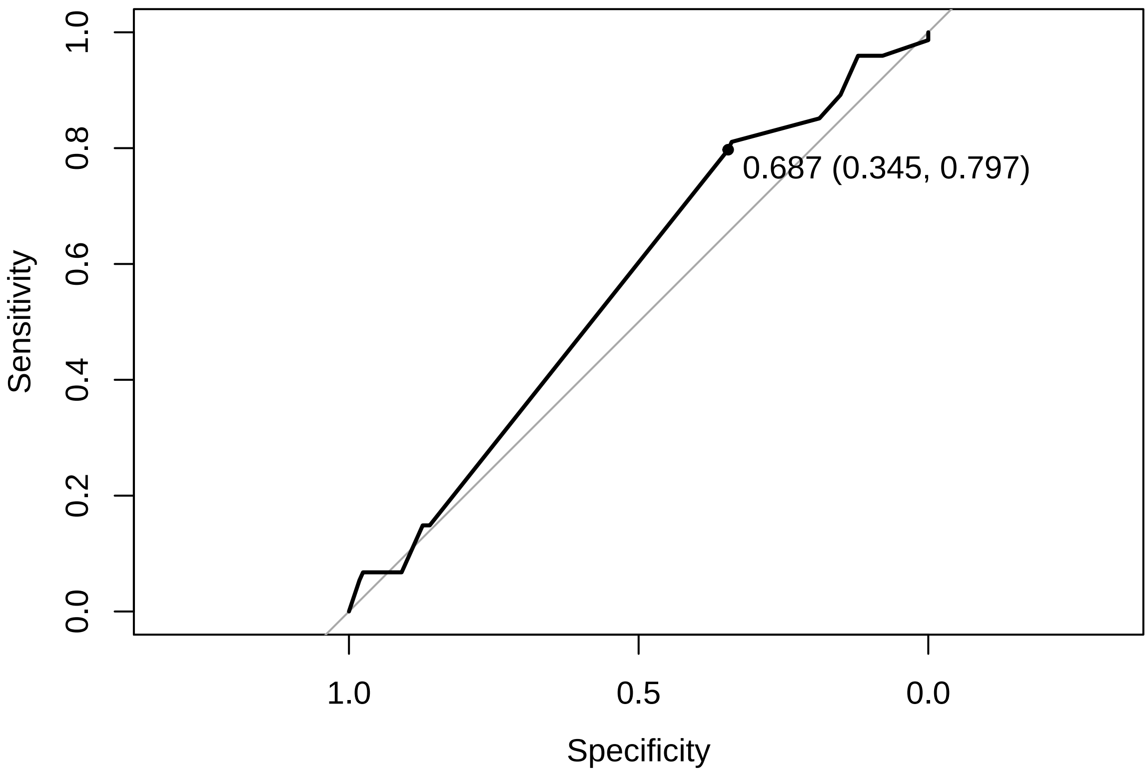
```r
set.seed(9003)
## Using radial basis kernel

## Training
## Due to CPU performance, cannot apply a grid to tune parameters
fit.rSVM <- train(Class ~ ., data = trainingData,
                  method = "svmRadial",
                  metric = "ROC",
                  trControl= TrainingParameters,
                  preProcess = c("scale","center"),
                  na.action = na.omit
)
fit.rSVM
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 563 samples
##   6 predictor
##   2 classes: 'Democrats', 'Republican'
##
## Pre-processing: scaled (6), centered (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 507, 506, 507, 508, 507, 507, ...
## Resampling results across tuning parameters:
##
##   C     ROC        Sens       Spec
##   0.25  0.4787260  0.9923077  0.005882353
##   0.50  0.5353951  0.9820513  0.005882353
##   1.00  0.5433448  0.9897436  0.005555556
##
## Tuning parameter 'sigma' was held constant at a value of 0.1043378
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1043378 and C = 1.
```

```r
fit.rSVM$bestTune
```

```
##       sigma C
## 3 0.1043378 1
```

```r
## Testing
pred.rSVM <- predict(fit.rSVM, testData, type="prob")

## Evaluation
#confusionMatrix(pred, testData$Class)
result.roc <- roc(testData$Class, pred.rSVM$Democrats)
plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")
```




```
result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accuracy
print(result.coords)
```

```
##           threshold  accuracy
## threshold 0.6872721 0.4853556
```

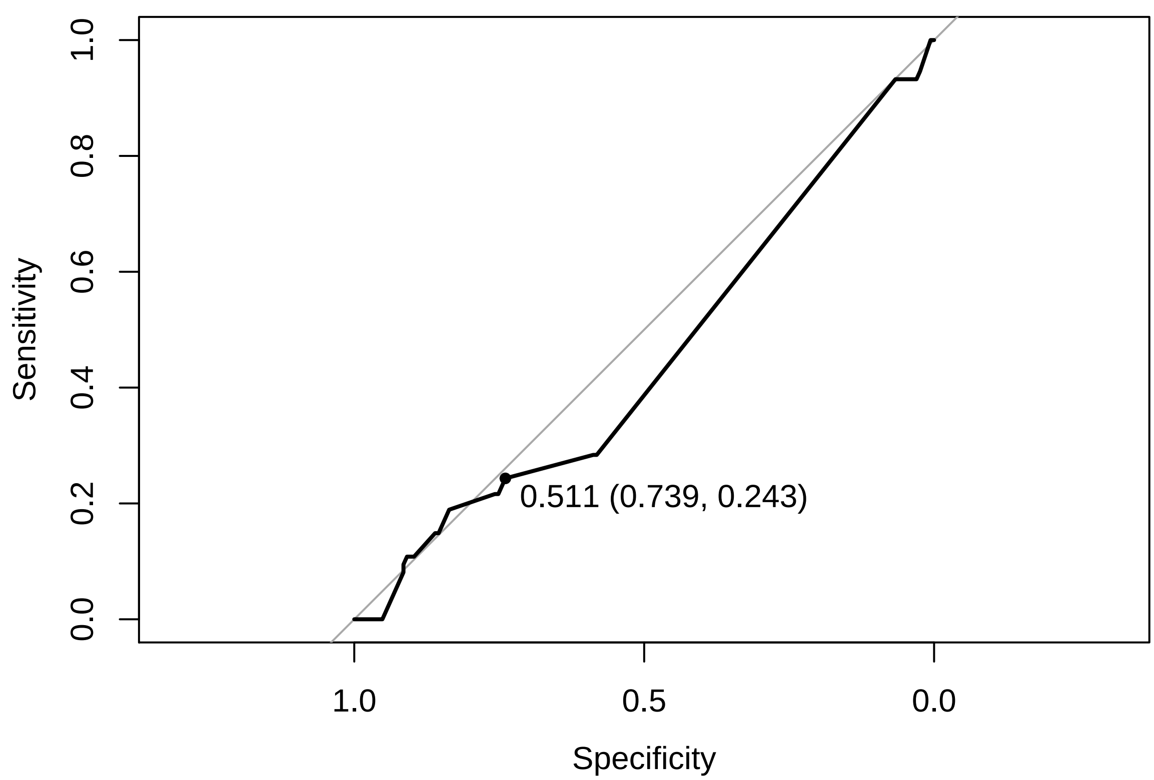
Classification using Naive Bayes

```
set.seed(9004)
## Training
fit.NB <- train(trainingData[,-ncol(trainingData)], trainingData$Class,
                    method = "nb",
                    metric = "ROC",
                    preProcess=c("scale","center"),
                    trControl= TrainingParameters,
                    na.action = na.omit
)
fit.NB
```

```
## Naive Bayes
##
## 563 samples
##   6 predictor
##   2 classes: 'Democrats', 'Republican'
##
```

```
## Pre-processing: scaled (6), centered (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 507, 507, 506, 507, 508, 506, ...
## Resampling results across tuning parameters:
##
##   usekernel  ROC        Sens       Spec
##   FALSE      0.6128860  0.3142375  0.8866013
##    TRUE      0.6080343  0.4427800  0.7294118
##
## Tuning parameter 'fL' was held constant at a value of 0
## Tuning
##  parameter 'adjust' was held constant at a value of 1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were fL = 0, usekernel = FALSE and adjust
##  = 1.
```

```r
## Testing
pred.NB <- predict(fit.NB, testData, type="prob")

## Evaluation
#confusionMatrix(pred, testData$Class)
result.roc <- roc(testData$Class, pred.NB$Democrats)
plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")
```



```r
result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accuracy
print(result.coords)
```

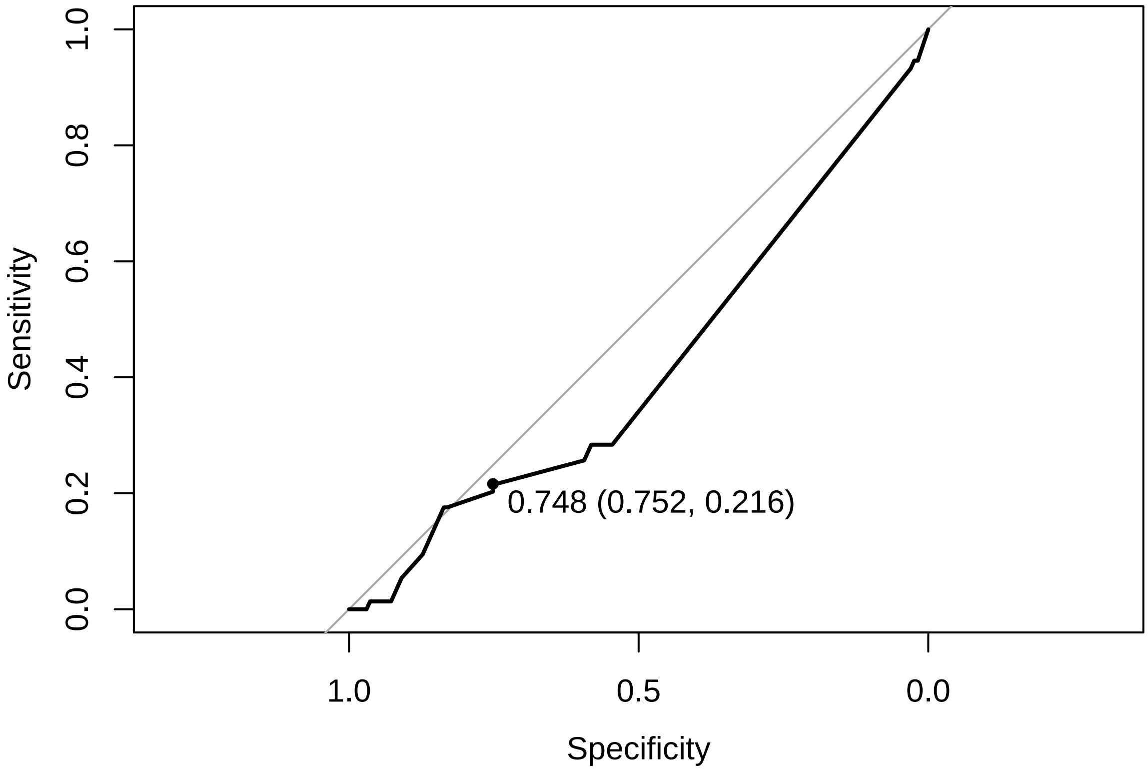
```
##           threshold  accuracy
## threshold  0.510868 0.5857741
```

Classification using Logistic Regression

```r
set.seed(9005)
## Training
fit.LR <- train(Class ~ ., data = trainingData,
                    method = "glm",
                    metric = "ROC",
                    preProcess=c("scale","center"),
                    trControl= TrainingParameters,
                    na.action = na.omit
)
fit.LR
```

```
## Generalized Linear Model
##
## 563 samples
##   6 predictor
##   2 classes: 'Democrats', 'Republican'
##
## Pre-processing: scaled (6), centered (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 508, 506, 506, 506, 507, 507, ...
## Resampling results:
##
##   ROC       Sens       Spec
##   0.606487  0.9717949  0.05196078
```

```r
## Testing
pred.LR <- predict(fit.LR, testData, type="prob")

## Evaluation
#confusionMatrix(pred, testData$Class)
result.roc <- roc(testData$Class, pred.LR$Democrats)
plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")
```




```
result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accuracy
print(result.coords)
```

```
##           threshold  accuracy
## threshold 0.7477121 0.5857741
```

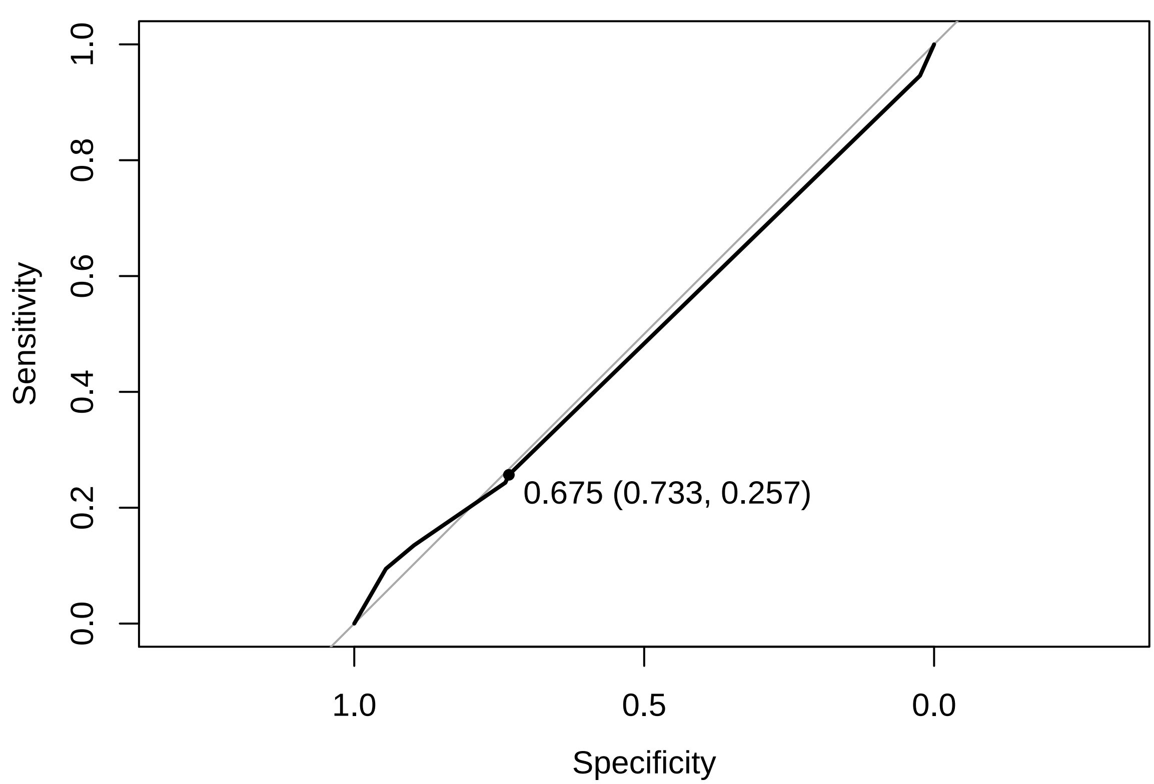
Classification using Decision Tree

```
set.seed(9006)
## Training
fit.DT <- train(Class ~ ., data = trainingData,
                    method = "rpart",
                    metric = "ROC",
                    preProcess=c("scale","center"),
                    trControl= TrainingParameters,
                    na.action = na.omit
)
fit.DT
```

```
## CART
##
## 563 samples
##   6 predictor
##   2 classes: 'Democrats', 'Republican'
##
```

```
## Pre-processing: scaled (6), centered (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 506, 507, 506, 507, 507, 506, ...
## Resampling results across tuning parameters:
##
##   cp           ROC        Sens       Spec
##   0.000000000  0.5459914  0.9973684  0.011437908
##   0.001142857  0.5459914  0.9973684  0.011437908
##   0.002285714  0.5227008  0.9973684  0.005555556
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.001142857.
```

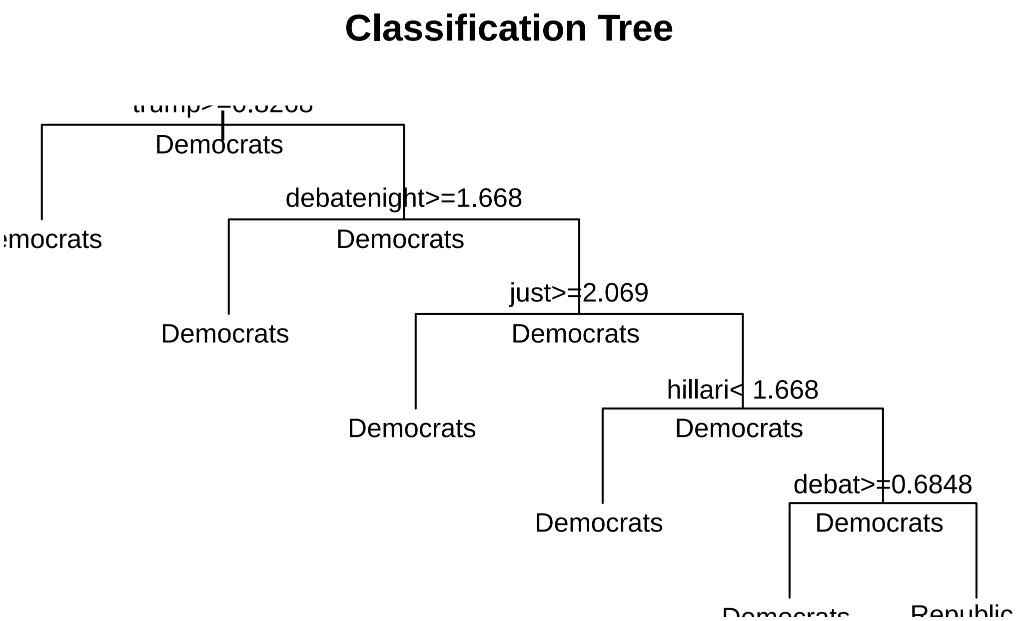
```
## Testing
pred.DT <- predict(fit.DT, testData, type="prob")

## Evaluation
#confusionMatrix(pred, testData$Class)
result.roc <- roc(testData$Class, pred.DT$Democrats)
plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")
```




```
result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accuracy
print(result.coords)
```

```
##           threshold  accuracy
## threshold 0.6754386 0.5857741
```

```r
plot(fit.DT$finalModel, uniform=TRUE, main="Classification Tree")
text(fit.DT$finalModel, use.n.=TRUE, all=TRUE, cex=.8)
```



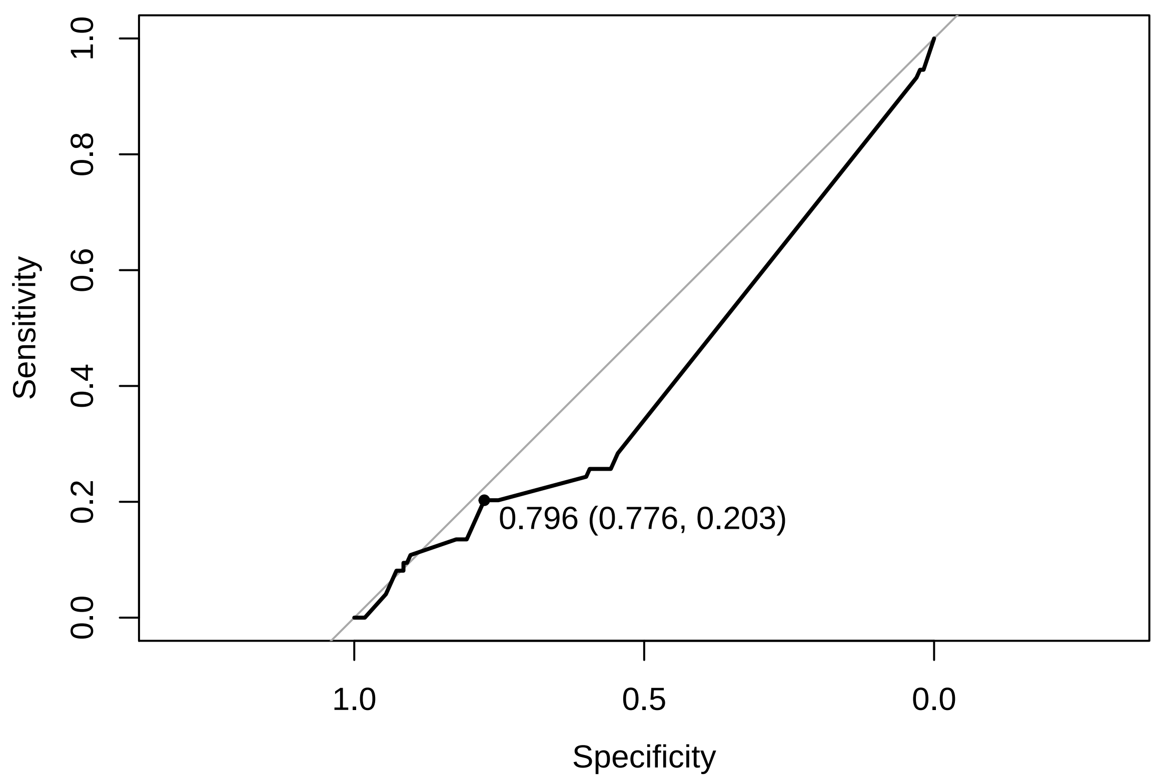
Classification using AdaBoost

```r
set.seed(9007)
## Training
fit.ADA <- train(trainingData[,-ncol(trainingData)], trainingData$Class,
                 method = "ada",
                 metric = "ROC",
                 preProcess=c("scale","center"),
                 trControl= TrainingParameters,
                 na.action = na.omit
)
fit.ADA
```

```
## Boosted Classification Trees
##
## 563 samples
##   6 predictor
##   2 classes: 'Democrats', 'Republican'
##
## Pre-processing: scaled (6), centered (6)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 507, 507, 506, 506, 507, 507, ...
```

```
## Resampling results across tuning parameters:
##
##   maxdepth  iter  ROC        Sens       Spec
##   1          50   0.5784609  1.0000000  0.000000000
##   1         100   0.5987609  1.0000000  0.000000000
##   1         150   0.6035772  1.0000000  0.000000000
##   2          50   0.6015338  1.0000000  0.000000000
##   2         100   0.6083345  0.9948718  0.005882353
##   2         150   0.6089374  0.9844804  0.022549020
##   3          50   0.6120182  0.9974359  0.000000000
##   3         100   0.6137247  0.9818489  0.028104575
##   3         150   0.6141977  0.9818489  0.028104575
##
## Tuning parameter 'nu' was held constant at a value of 0.1
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were iter = 150, maxdepth = 3 and nu = 0.1.
```

```r
## Testing
pred.ADA <- predict(fit.ADA, testData, type="prob")

## Evaluation
#confusionMatrix(pred, testData$Class)
result.roc <- roc(testData$Class, pred.ADA$Democrats)
plot(result.roc, print.thres="best", print.thres.best.method="closest.topleft")
```



```r
result.coords <- coords(result.roc, "best", best.method="closest.topleft", ret=c("threshold", "accuracy
print(result.coords)
```

```
##           threshold  accuracy
## threshold 0.7963398 0.5983264
```