classification_Xinming

Xinming Liu

11/27/2020

Set A - Debate 1

```r
## SetA Preprocessing
users <- read.csv("user_setA/users.csv", stringsAsFactors = FALSE)
#head(a.users)
tweets <- read.csv("user_setA/tweets_debate3.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=="PA")
#head(data)
use <- data.frame(data$text, factor(data$party), stringsAsFactors = FALSE)
colnames(use) <- c("text", "party")
#head(use)
```

```r
## 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
```

```r
## 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
```

```r
## 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 0.001908255 0.001890649maximum number of iterations reached 0.0
```

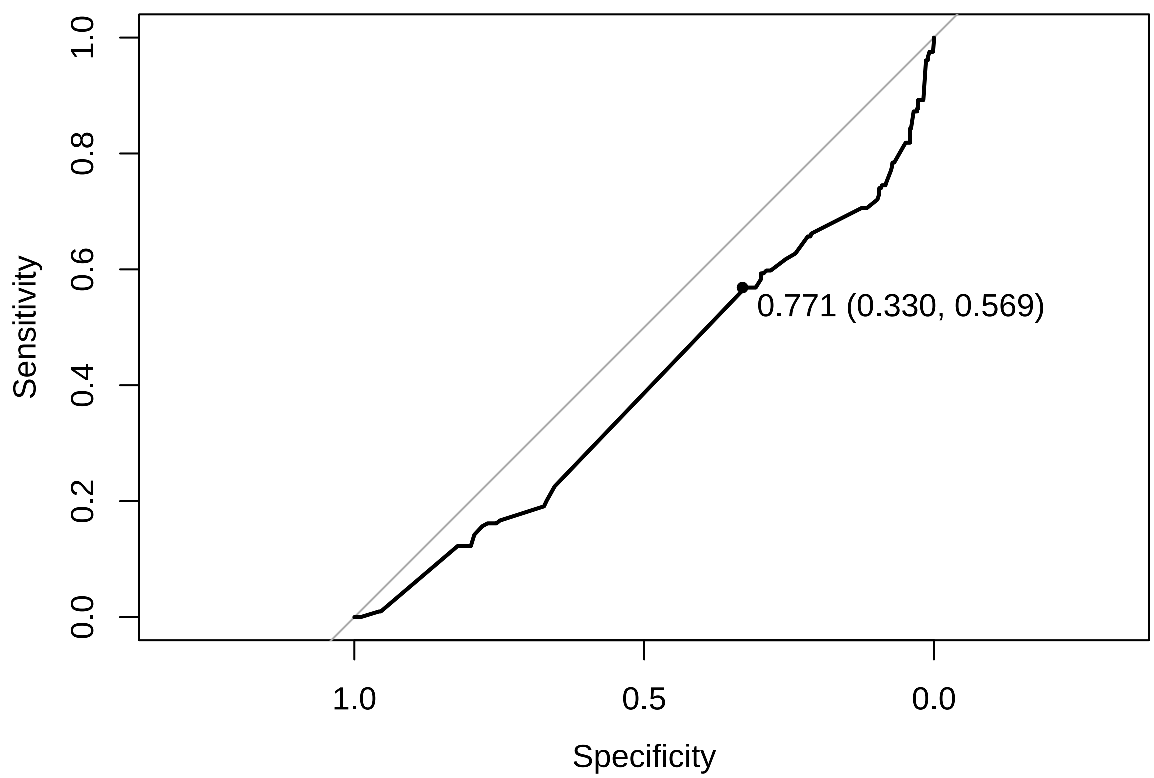
```r
fit.lSVM
```

```
## Support Vector Machines with Linear Kernel
##
## 2010 samples
##   10 predictor
##    2 classes: 'Democrats', 'Republican'
##
## Pre-processing: scaled (10), centered (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1809, 1810, 1809, 1808, 1809, 1808, ...
## Resampling results across tuning parameters:
##
##   C       ROC        Sens       Spec
##   0.1000  0.5216105  0.9980392  0.00000000
##   0.2125  0.5158934  0.9876114  0.01679965
##   0.3250  0.5496734  0.9876114  0.01679965
##   0.4375  0.5233665  0.9980392  0.00000000
##   0.5500  0.4952598  0.9921569  0.01041667
##   0.6625  0.5177286  0.9980392  0.00000000
##   0.7750  0.5553563  0.9980392  0.00000000
##   0.8875  0.5149111  0.9980392  0.00000000
##   1.0000  0.5243349  0.9980392  0.00000000
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.775.
```

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

```
##       C
## 7 0.775
```

```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  0.770811 0.3867596
```

```
# 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)
```

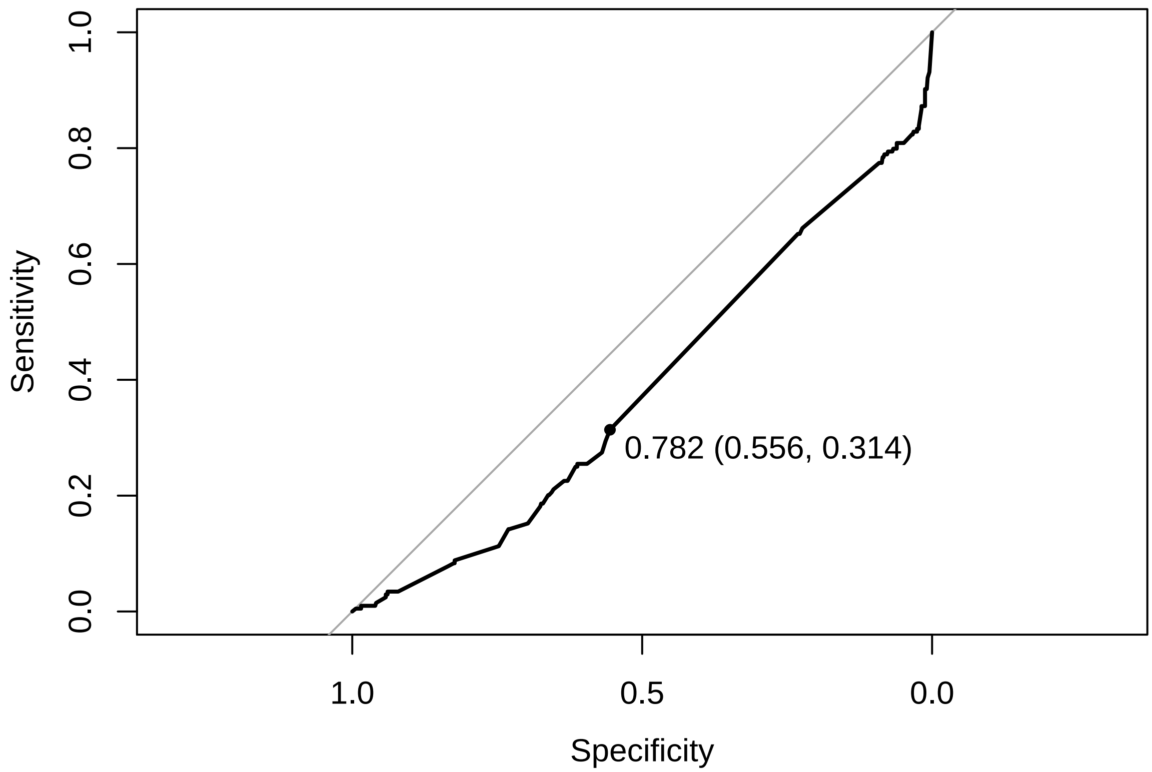
```r
set.seed(9003)
## Using radial basis kernel

## 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
##
## 2010 samples
##   10 predictor
##    2 classes: 'Democrats', 'Republican'
##
## Pre-processing: scaled (10), centered (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1810, 1808, 1809, 1810, 1809, 1809, ...
## Resampling results across tuning parameters:
##
##   C     ROC        Sens       Spec
##   0.25  0.5894976  0.9915372  0.07974291
##   0.50  0.5697093  0.9869833  0.09649823
##   1.00  0.5666541  0.9804728  0.12180851
##
## Tuning parameter 'sigma' was held constant at a value of 0.104267
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.104267 and C = 0.25.
```

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

```
##      sigma    C
## 1 0.104267 0.25
```

```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")
```




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

```
##           threshold  accuracy
## threshold 0.7820195 0.4982578
```

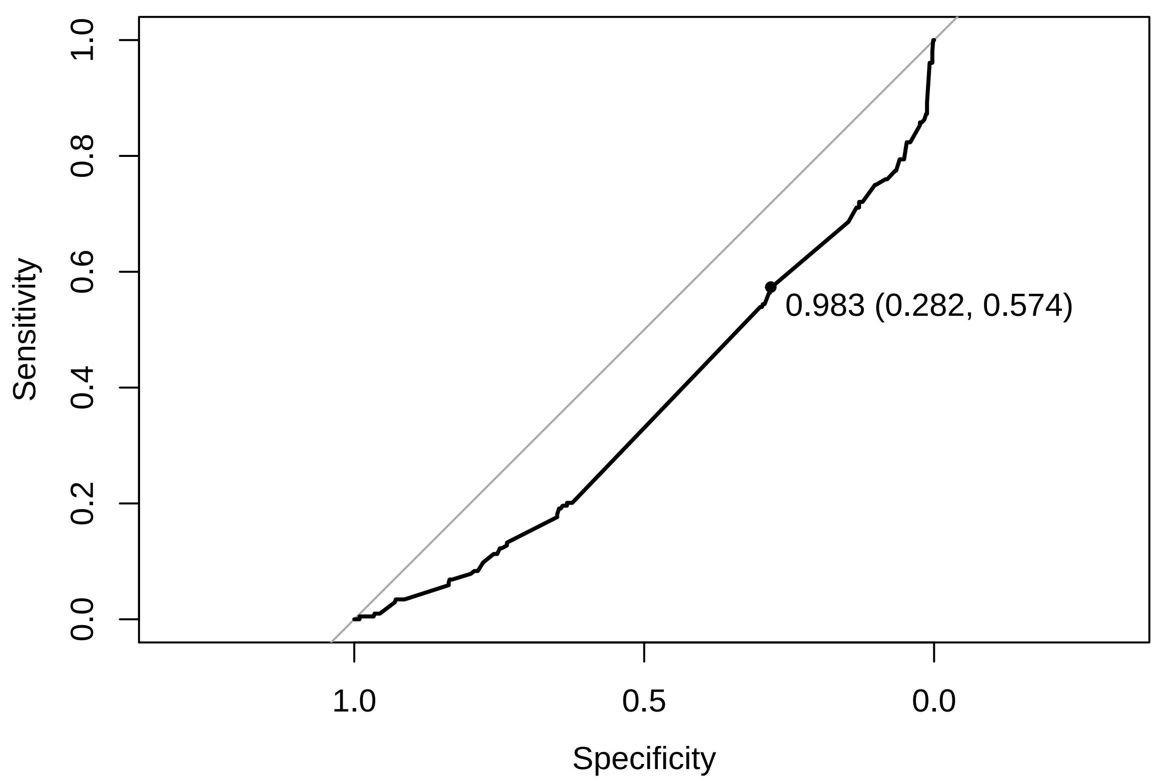
Classification using Naive Bayes

```r
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
##
## 2010 samples
##   10 predictor
##    2 classes: 'Democrats', 'Republican'
##
```

```
## Pre-processing: scaled (10), centered (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1809, 1809, 1808, 1810, 1809, 1808, ...
## Resampling results across tuning parameters:
##
##   usekernel  ROC        Sens       Spec
##   FALSE      0.6284958  0.8663908  0.2732713
##    TRUE      0.6362737  1.0000000  0.0000000
##
## 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 = TRUE 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)
```

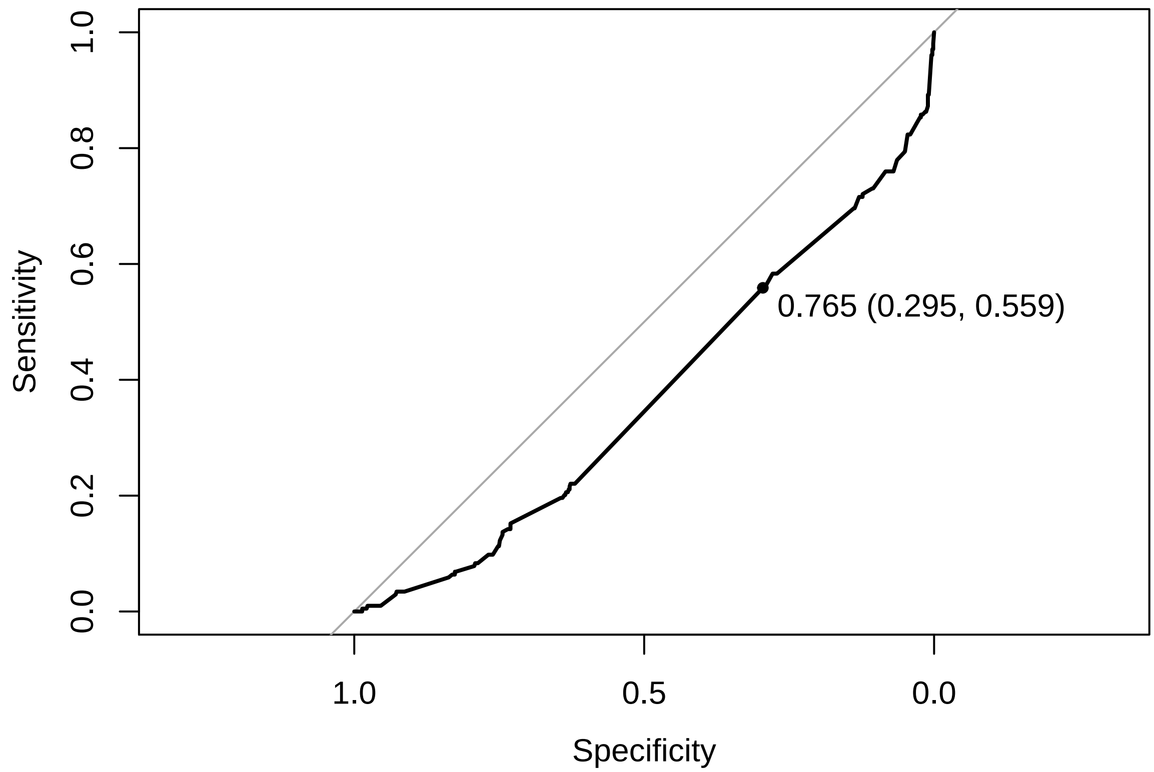
```
##           threshold  accuracy
## threshold 0.9827756 0.3507549
```

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
##
## 2010 samples
##   10 predictor
##    2 classes: 'Democrats', 'Republican'
##
## Pre-processing: scaled (10), centered (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1809, 1810, 1808, 1809, 1809, 1810, ...
## Resampling results:
##
##   ROC        Sens       Spec
##   0.6360117  0.9823954  0.05066489
```

```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 0.7647984 0.3577236
```

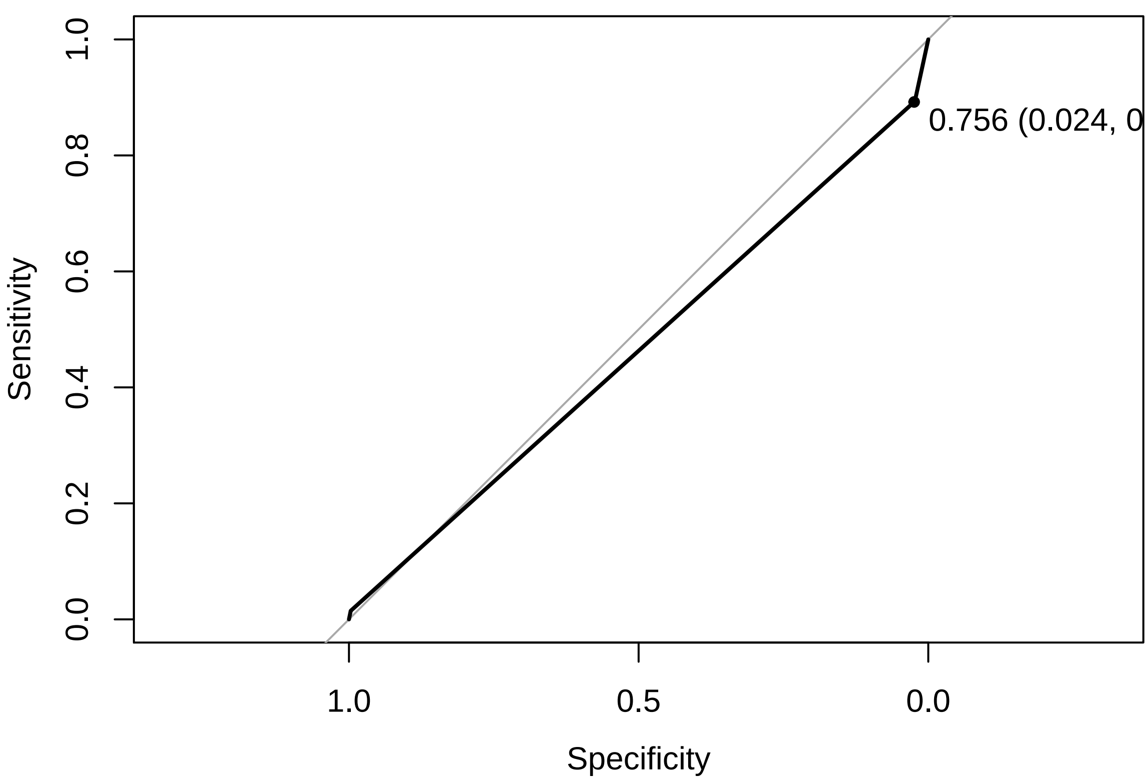
Classification using Decision Tree

```
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
##
## 2010 samples
##   10 predictor
##    2 classes: 'Democrats', 'Republican'
##
```

```
## Pre-processing: scaled (10), centered (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1808, 1810, 1808, 1809, 1809, 1809, ...
## Resampling results across tuning parameters:
##
##   cp          ROC        Sens       Spec
##   0.01260504  0.5258730  0.9771878  0.08204787
##   0.01470588  0.5167761  0.9791529  0.06121454
##   0.01785714  0.5090492  0.9817673  0.04228723
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.01260504.
```

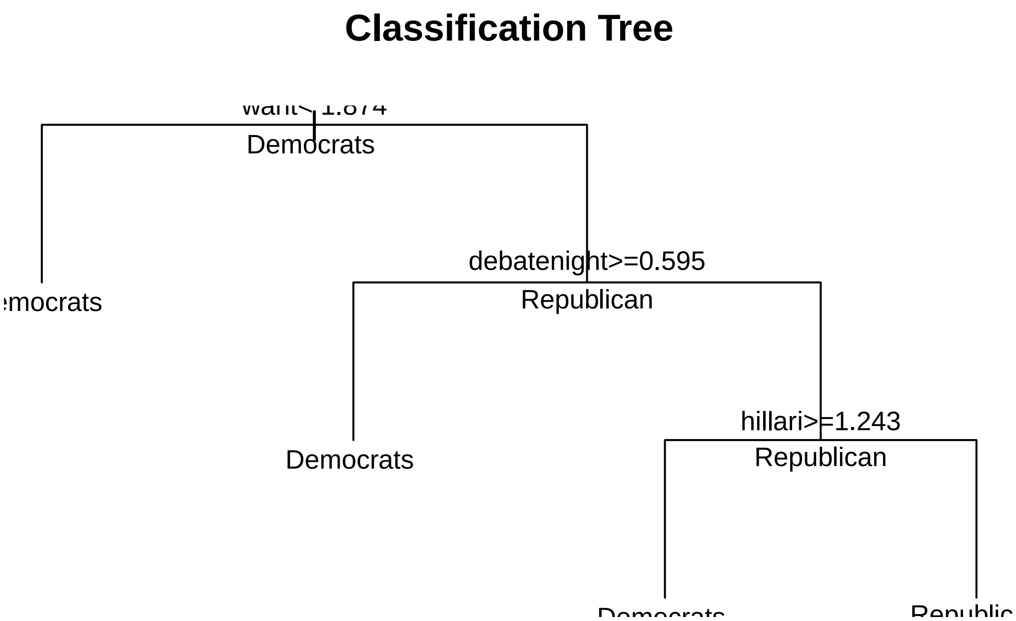
```
## Testing
pred.DT <- predict(fit.DT, testData, type="prob")

## 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 0.7564035 0.2299652
```

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



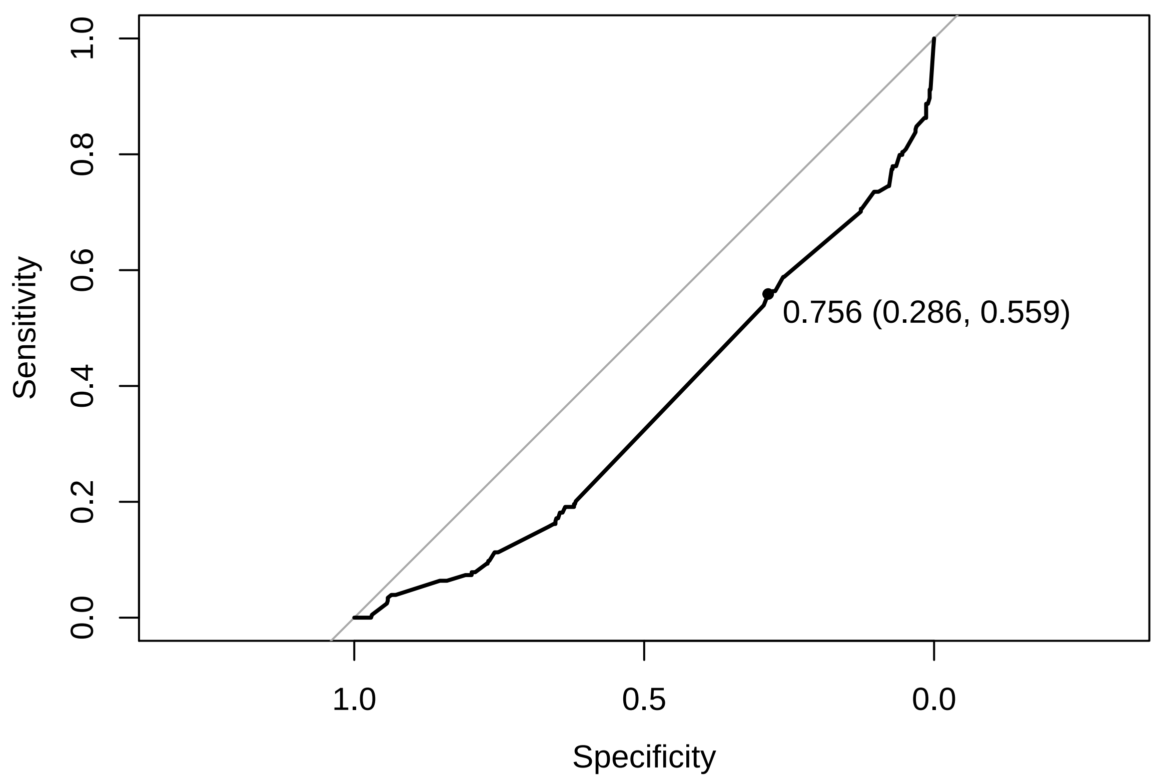
Classification using AdaBoost

```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
##
## 2010 samples
##   10 predictor
##    2 classes: 'Democrats', 'Republican'
##
## Pre-processing: scaled (10), centered (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1809, 1808, 1808, 1808, 1809, 1809, ...
```

```
## Resampling results across tuning parameters:
##
##   maxdepth  iter  ROC        Sens       Spec
##   1          50   0.6121298  1.0000000  0.000000000
##   1         100   0.6306340  0.9980435  0.004166667
##   1         150   0.6368190  0.9980435  0.008333333
##   2          50   0.6385099  0.9856634  0.050310284
##   2         100   0.6378876  0.9811094  0.084086879
##   2         150   0.6383334  0.9785035  0.098714539
##   3          50   0.6412266  0.9837026  0.102703901
##   3         100   0.6414884  0.9830490  0.109042553
##   3         150   0.6430482  0.9817503  0.111170213
##
## 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 0.7564355 0.3507549
```