557_Project

Ben Straub

4/11/2017

Data overview

Mining activity has long been associated with mining hazards, such as fires, floods, and toxic contaminants (Dozolme, P., 2016). Among these hazards, seismic hazards are the hardest to detect and predict (Sikora & Wróbel, 2010). Minimizing loss from seismic hazards requires both advanced data collection and analysis. In recent years, more and more advanced seismic and seismoacoustic monitoring systems have come about. Still, the disproportionate number of low-energy versus high-energy seismic phenomena (e.g. $> 10^4$J) renders traditional analysis methods insufficient.

In this project, we used the seismic-bumps dataset provided by Sikora & Wróbel (2010), found in the UCI Machine Learning Repository. This seismic-bumps dataset comes from a coal mine located in Poland and contains 2584 observations of 19 attributes. Each observation summarizes seismic activity in the rock mass within one 8-hour shift. Note that the decision attribute, named "class", has values 1 and 0. This variable is the response variable we use in this project. A class value of "1" is categorized as "hazardous state", which essentially indicates a registered seismic bump with high energy ($>10^4$J) in the next shift. A class value "0" represents non-hazardous state in the next shift. According to Bukowska (2006), a number of factors having an effect on seismic hazard occurrence were proposed. Among other factors, the occurrence of tremors with energy $> 10^4$J was listed. The purpose is to find whether and how the other 18 variables can be used to determine the hazard status of the mine.

Table 1. Attribute information of the seismic-bumps dataset

Data Attributes	Description
seismic	result of shift seismic hazard assessment: 'a' - lack of hazard, 'b' - low hazard, 'c' - high hazard, 'c
seismoacoustic	result of shift seismic hazard assessment
shift	type of a shift: 'W' - coal-getting, 'N' - preparation shift
genergy	seismic energy recorded within previous shift by active geophones (GMax) monitoring the longwal
gpuls	number of pulses recorded within previous shift by GMax
gdenergy	deviation of recorded energy within previous shift from average energy recorded during eight prev
gdpuls	deviation of recorded pulses within previous shift from average number of pulses recorded during e
ghazard	result of shift seismic hazard assessment by the seismoacoustic method based on registration comi
nbumps	the number of seismic bumps recorded within previous shift
nbumps$i$, $i \in \{1, \ldots, 5\}$	the number of seismic bumps ($10^i - 10^{i+1}$ J) registered within previous shift
energy	total energy of seismic bumps registered within previous shift
maxenergy	maximum energy of the seismic bumps registered within previous shift
class	the decision attribute: '1' - high energy seismic bump occurred in the next shift ('hazardous state

Exploratory Data Analysis

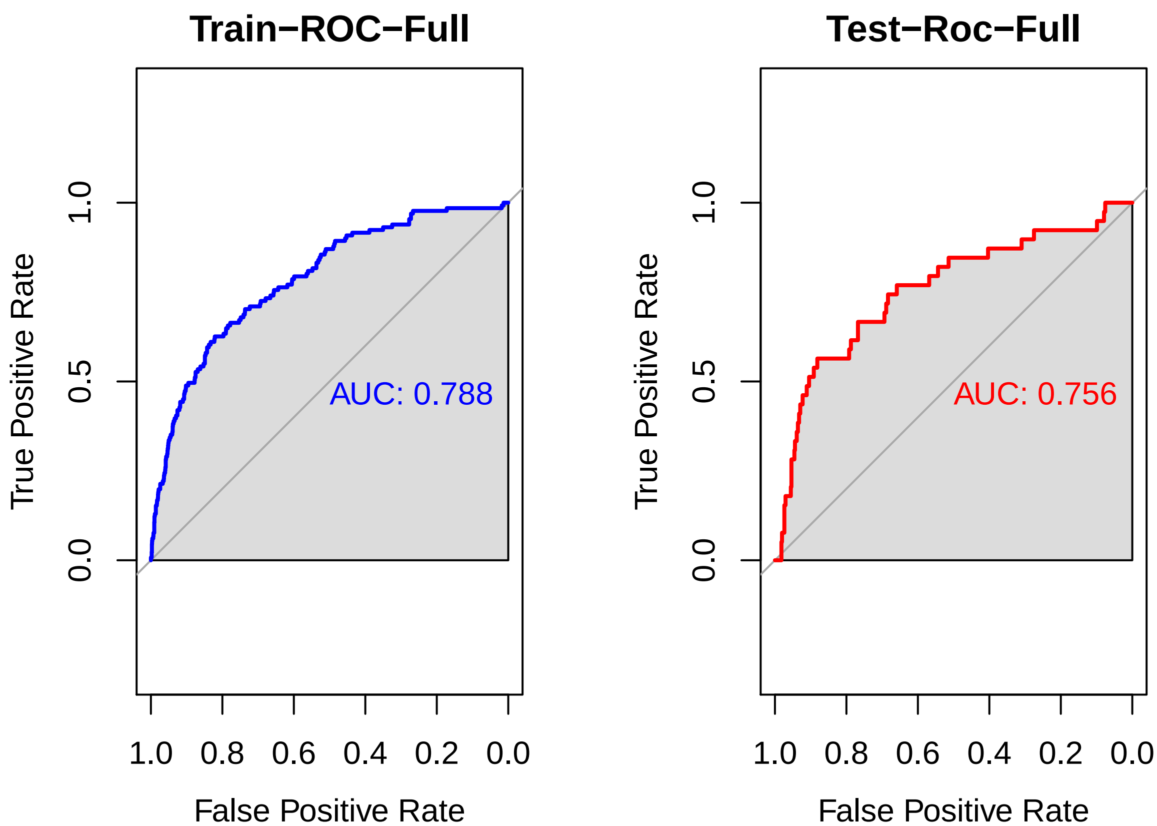
The state of the mine was indeed deemed hazardous infrequently − only 170 shifts out of 2584 − a difficult problem in our analyses. We want to examine which observations of seismic activity can help in the prediction of the hazard state of the mine during the next shift. Regression diagnostics indicate that the data, in general, meet most assumptions. However, we see that that data are somewhat skewed right, and there is severe multicollinearity (VIF > 10) between some of the covariates, as shown below.

Classification before Variable Selection

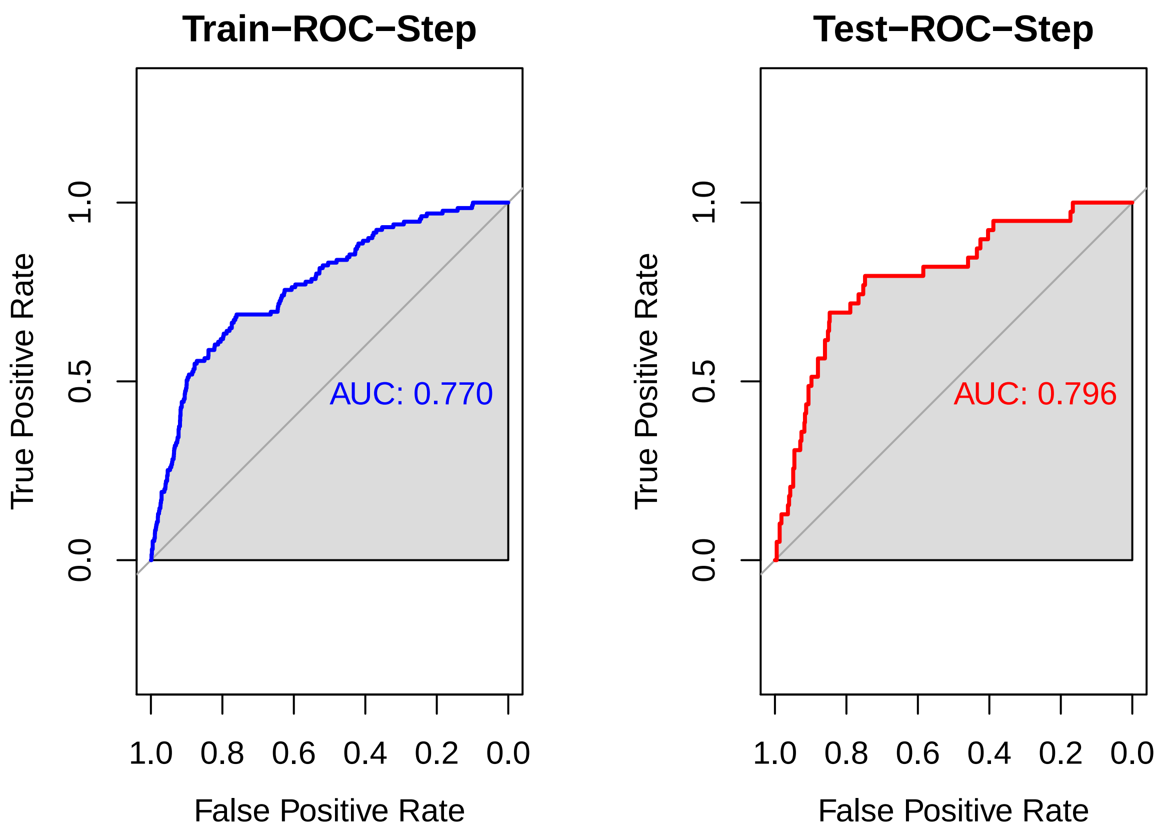
We first take the seismic-bumps dataset and partition the data into training (75%) and test (25%) datasets. The next steps involve examining multiple classification methods on the training and test datasets separately. The goal is to examine which classification method outputs comparatively better prediction for seismic hazards based on available predictors.

Logistic Regression

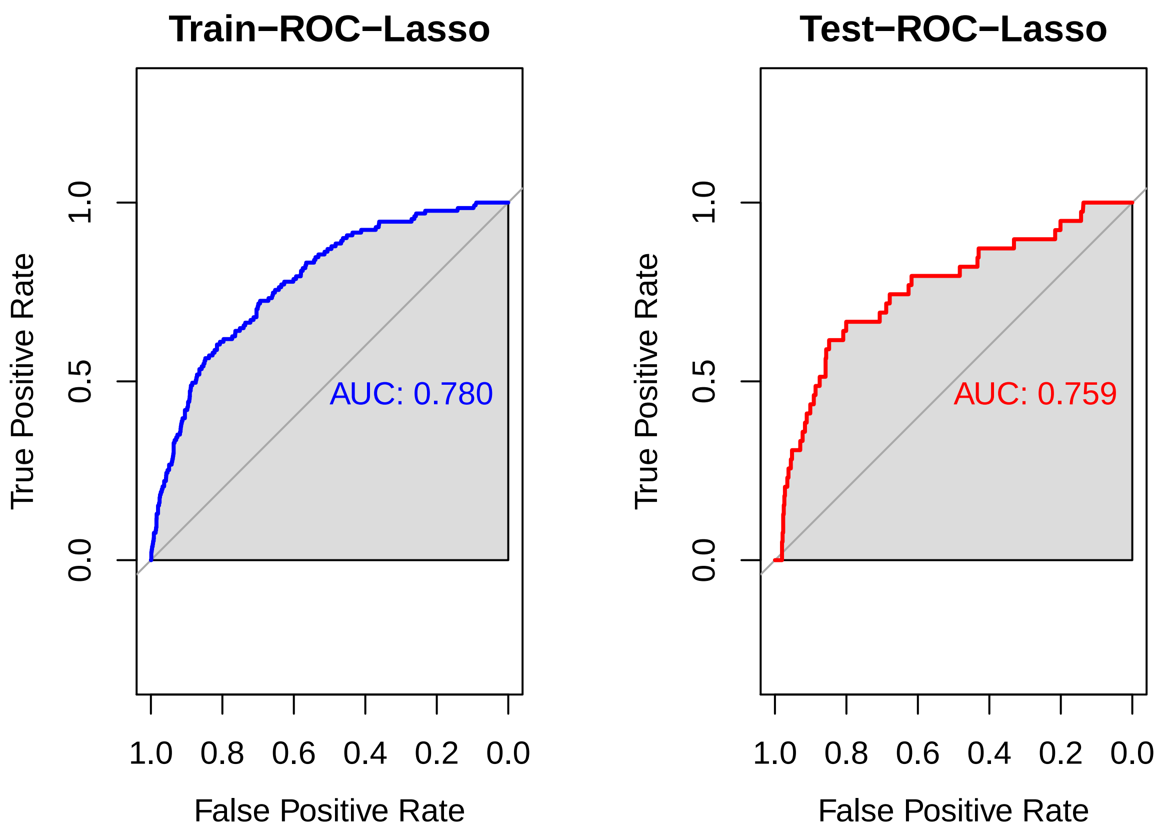
Full Model



Logistic Regression - Step Model



Logistic Regression - Lasso Model



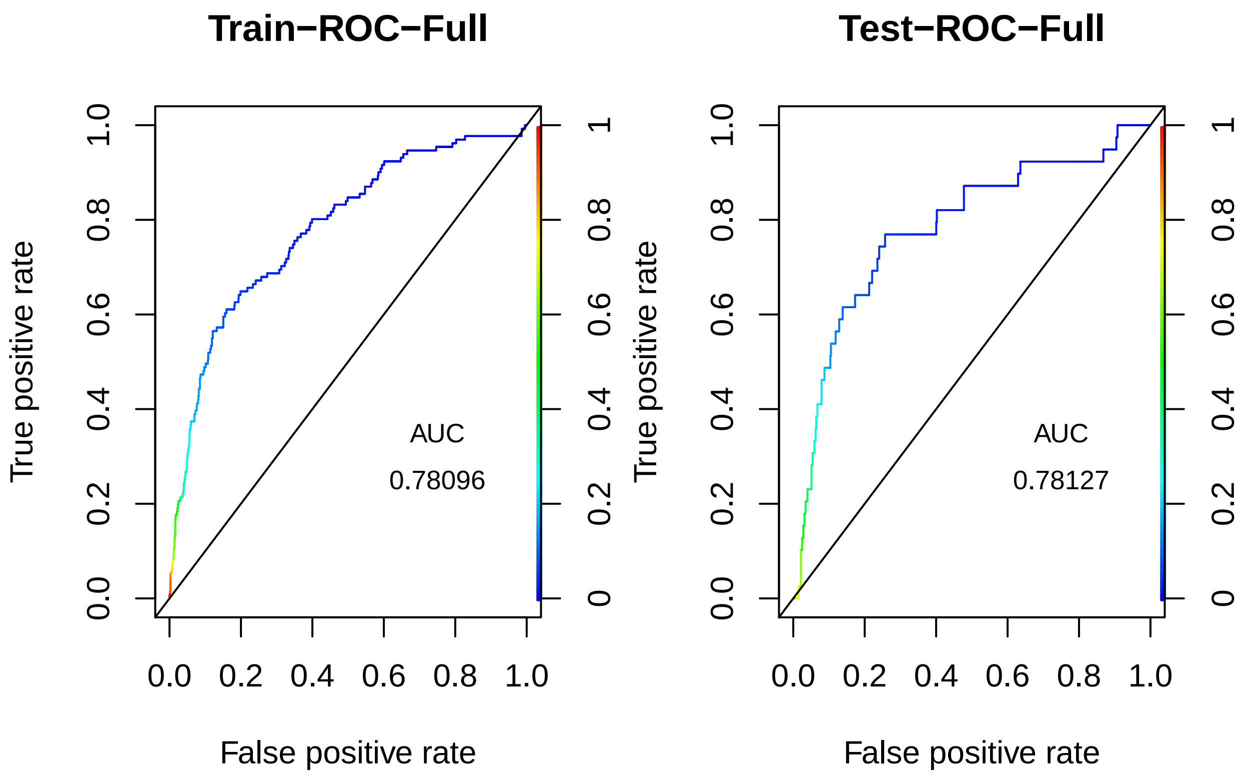

```
        time1 time2 time3
elapsed 0.124 0.205 0.071
```

```
     rate1.train rate3.train rate5.train
[1,]       0.067        0.07       0.068
```

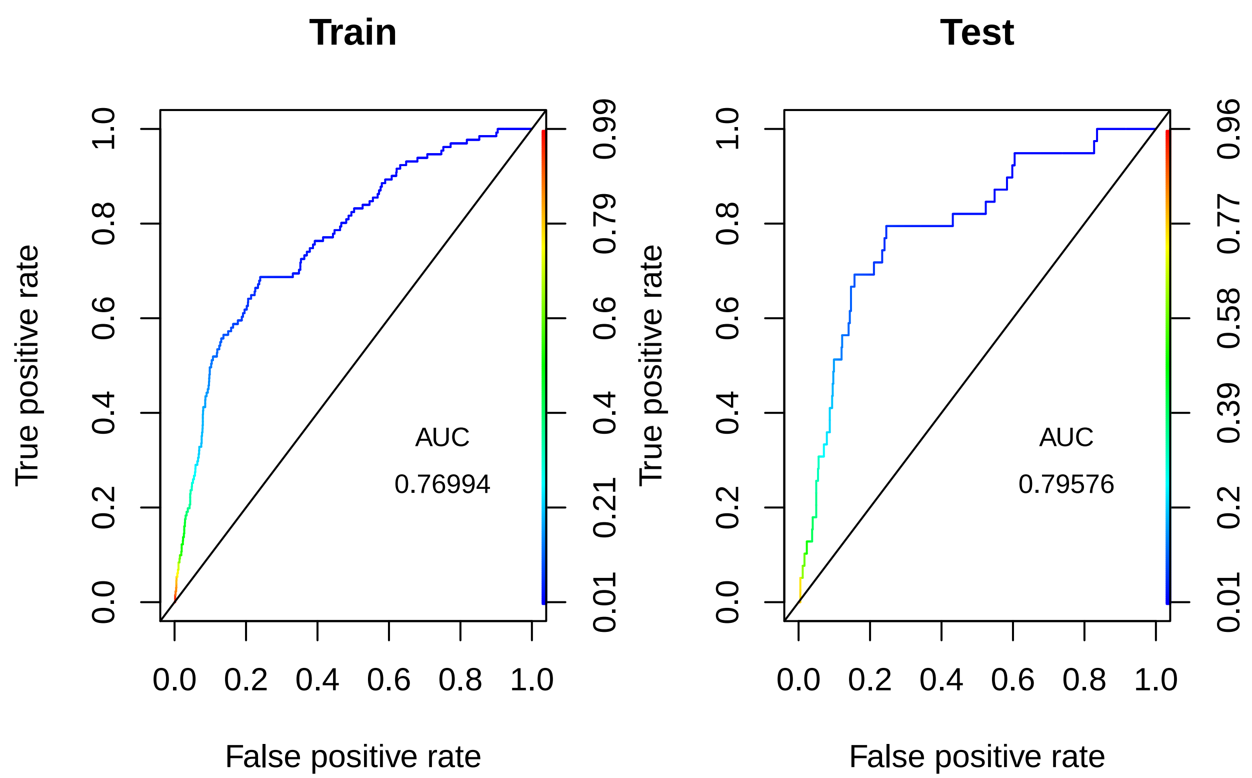
```
     rate2.test rate4.test rate6.test
[1,]      0.065      0.062      0.062
```

Linear Discriminant Analysis

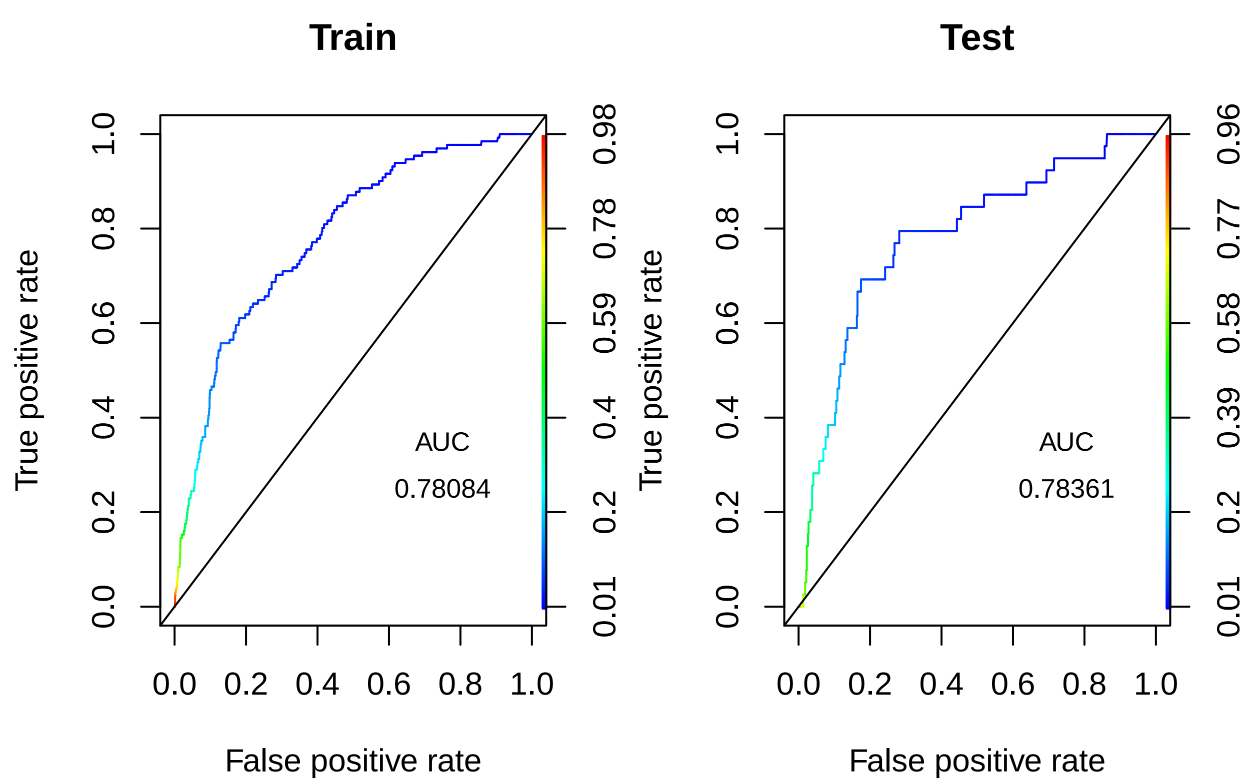
Full Model



Linear Discriminant Analysis - Step



Linear Discriminant Analysis - Lasso



```
        time1 time2 time3
elapsed  0.81 0.748 0.844
```

```
     rate1.train rate3.train rate5.train
[1,]       0.074       0.081       0.077
```

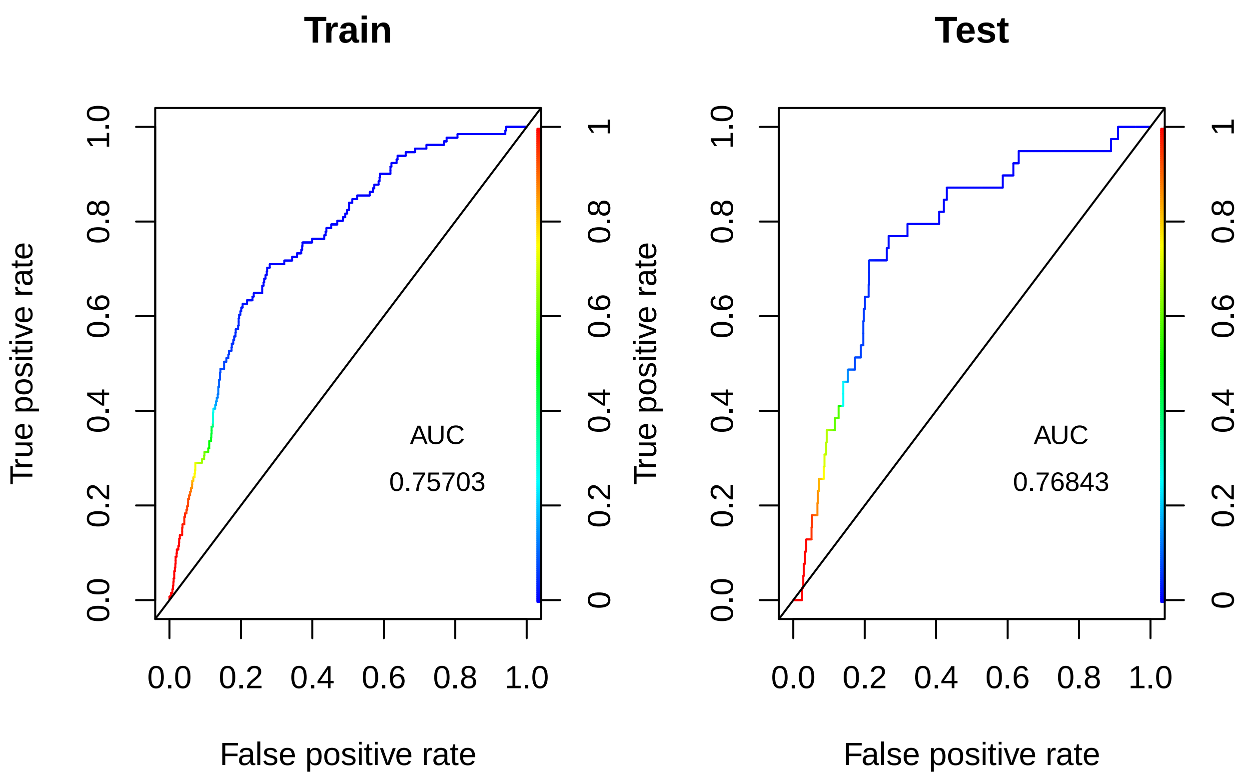
```
     rate2.test rate4.test rate6.test
[1,]      0.077      0.076      0.076
```

Quadratic Discriminant Analysis

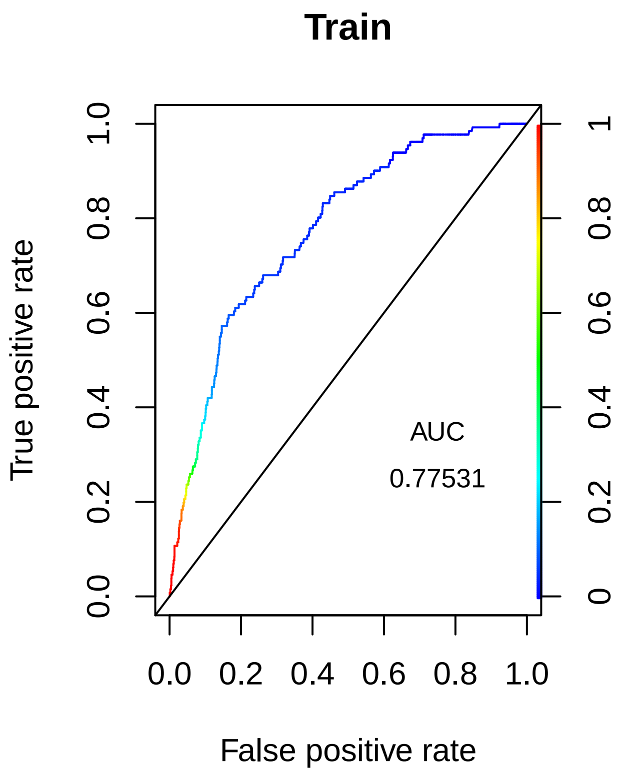
Full Model

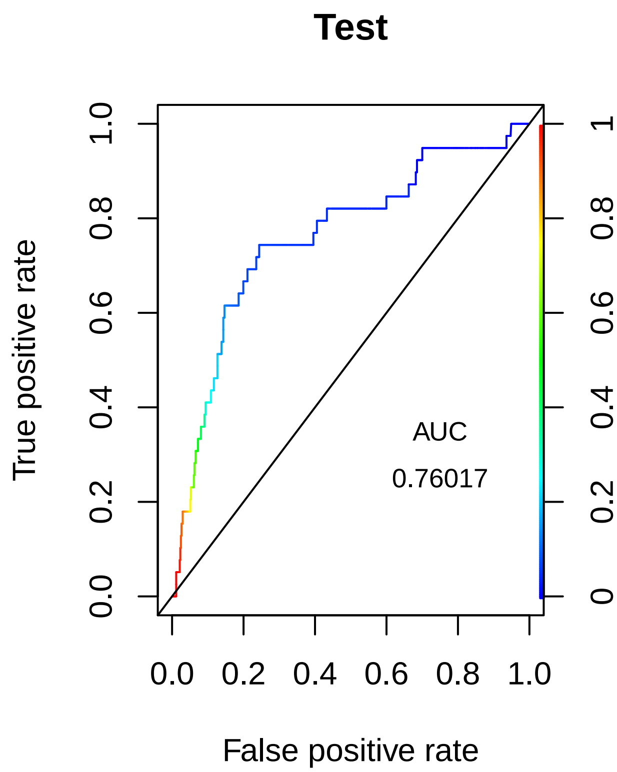
Full Model not able to handle the multicollinearity of the data.

Quadratic Discriminant Analysis - Step



Quadratic Discriminant Analysis - LASSO



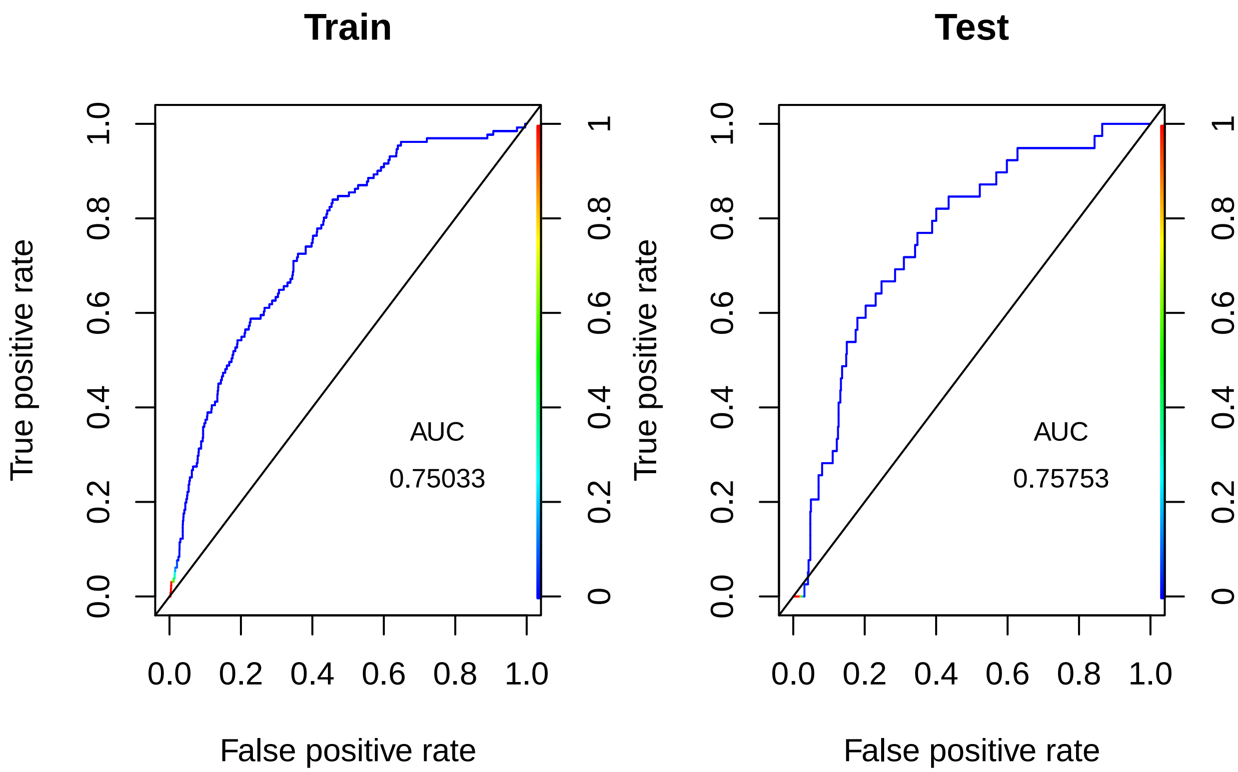



```
        time1 time2
elapsed 0.761 0.803
```

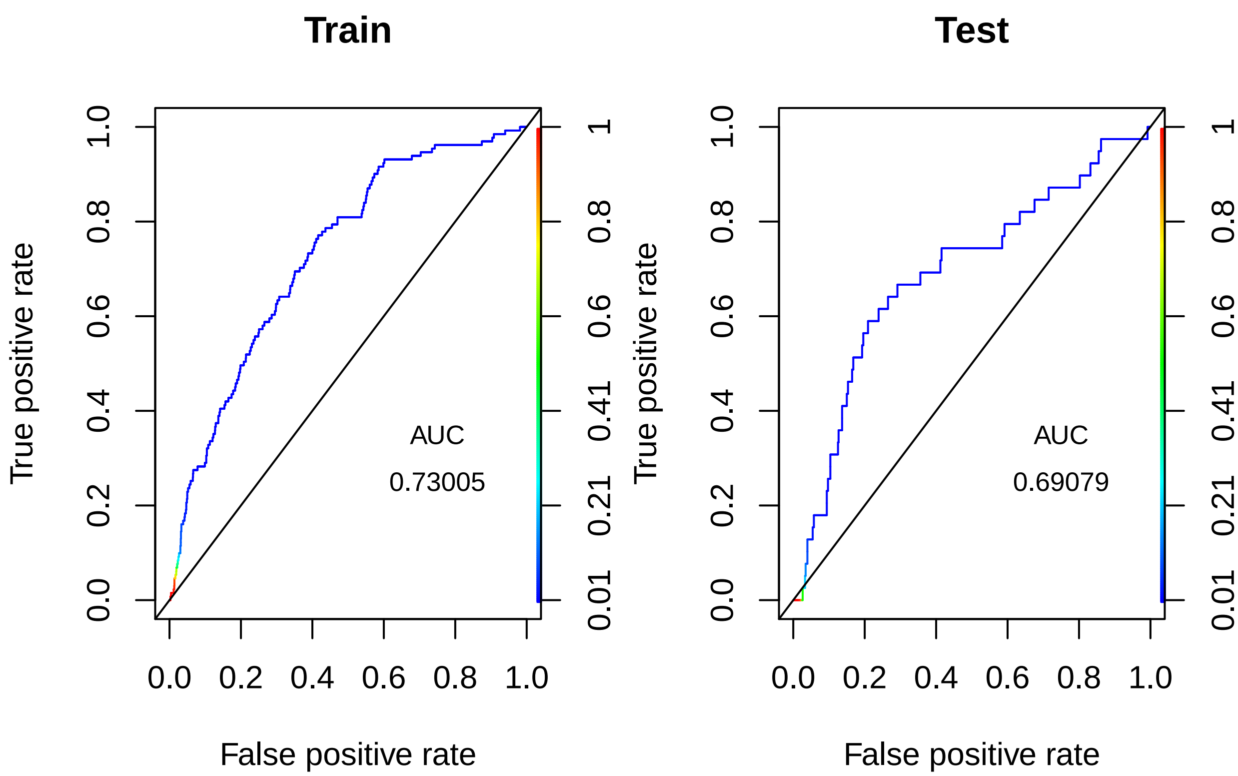
```
     rate1.train rate3.train rate5.train
[1,]       0.149       0.109       0.077
```

```
     rate2.test rate4.test rate6.test
[1,]      0.159      0.107      0.076
```

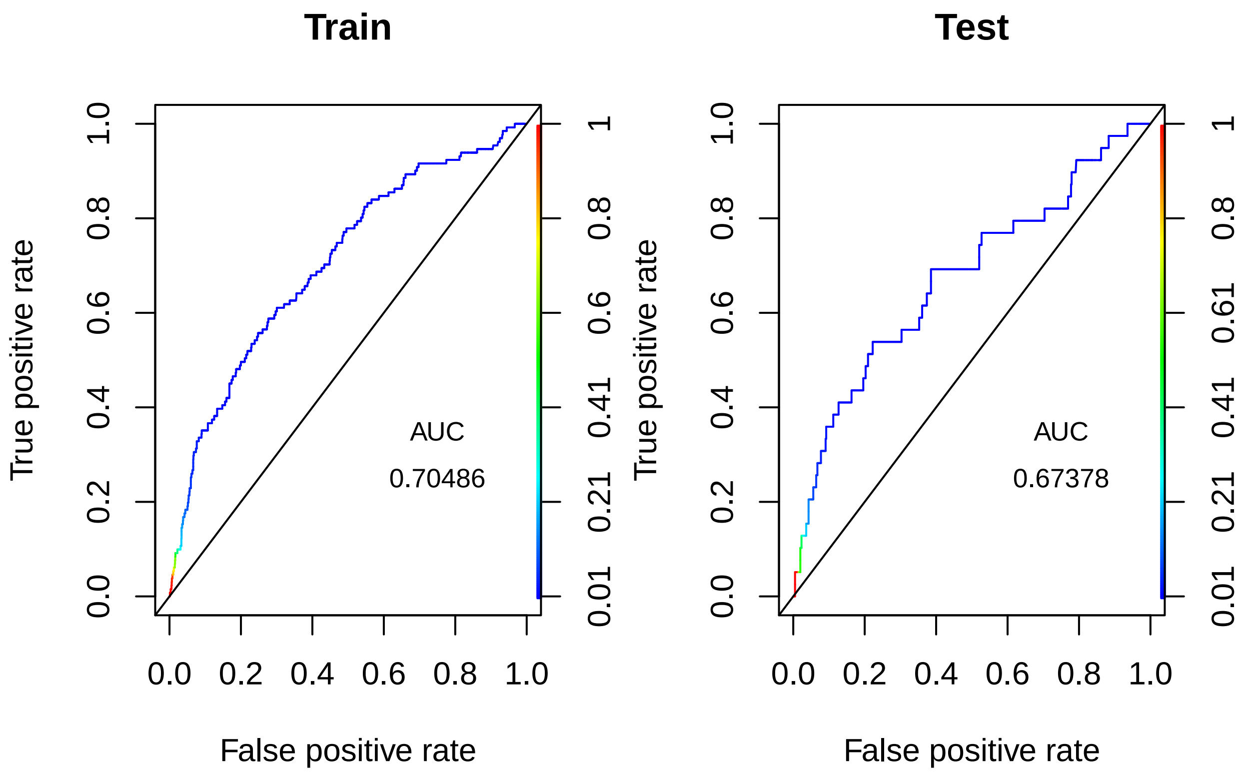
Regularized



Regularized Discriminant Analysis - Step



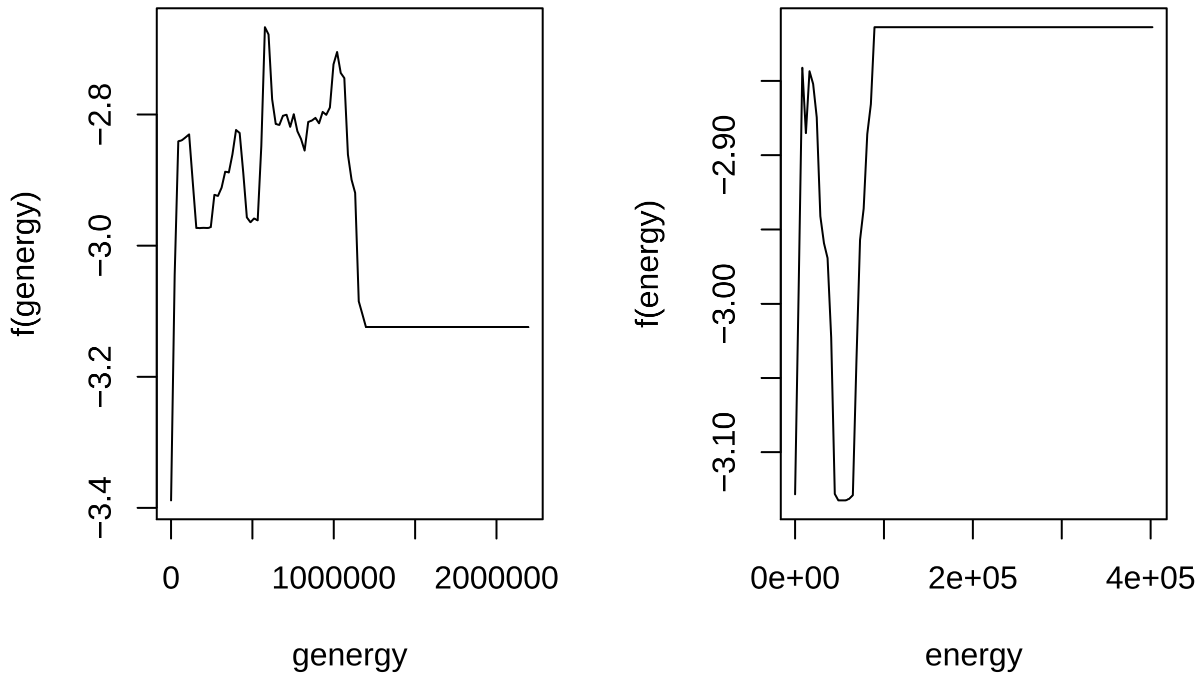
Regularized Discriminant Analysis - Lasso

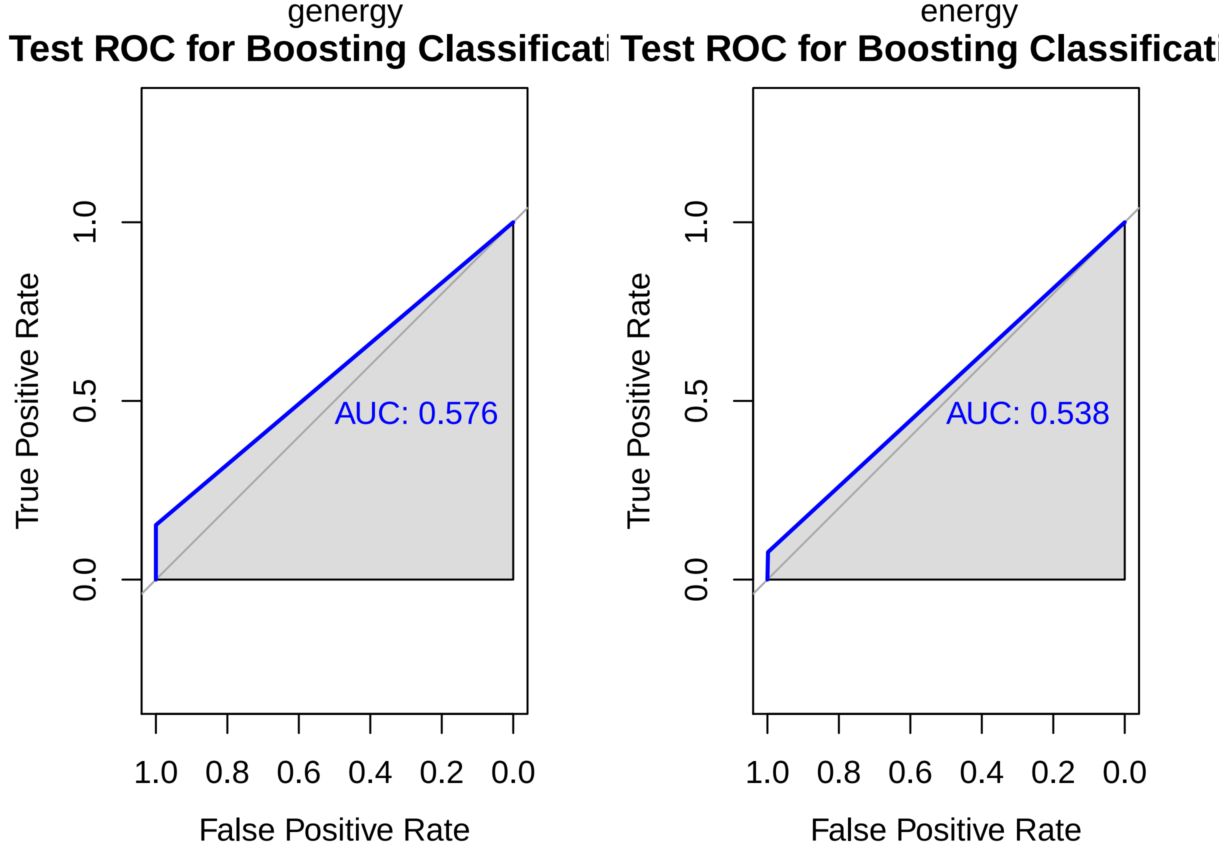



```
        time1 time2
elapsed 3.416 1.672

     rate1.train rate3.train rate5.train
[1,]       0.076       0.082       0.077

     rate2.test rate4.test rate6.test
[1,]      0.082      0.085      0.074
```

Boosting before variable selection

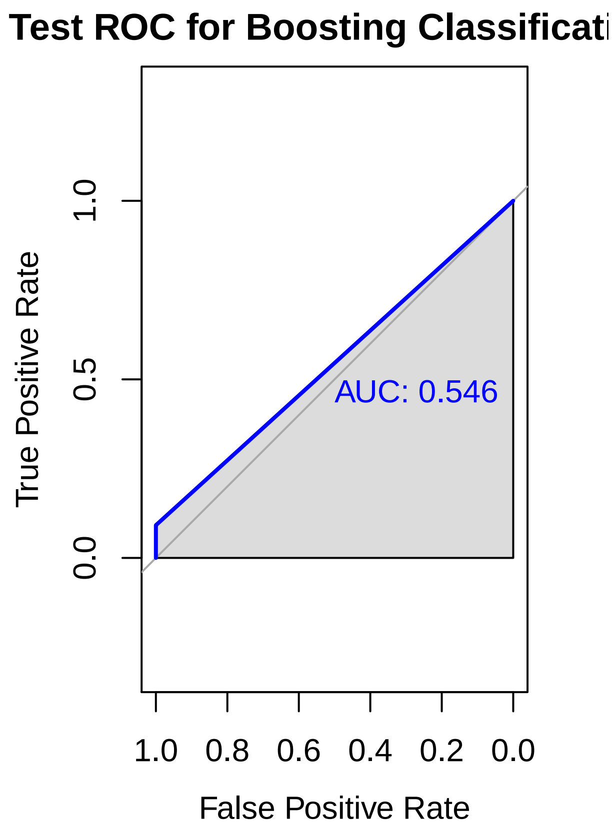
```
elapsed
  7.969
```

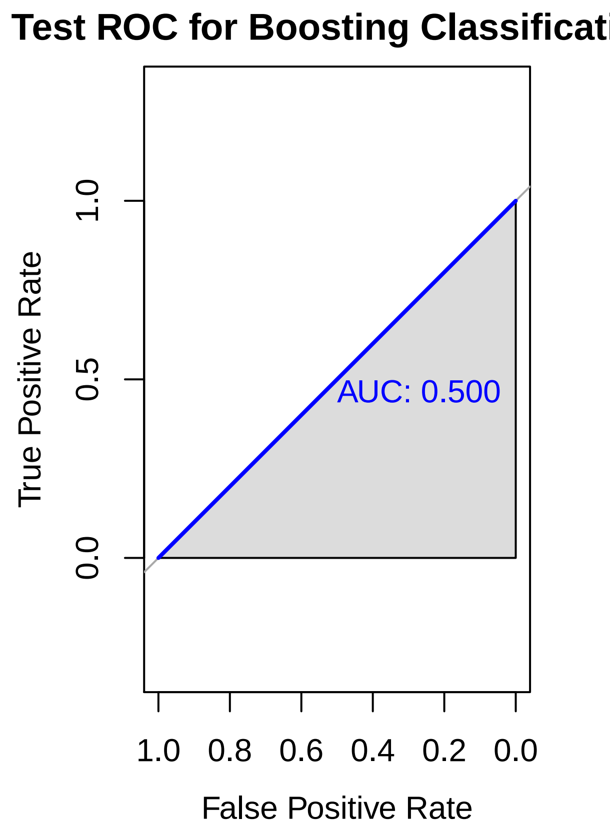




Boosting after variable selection

```
elapsed
  3.776
```

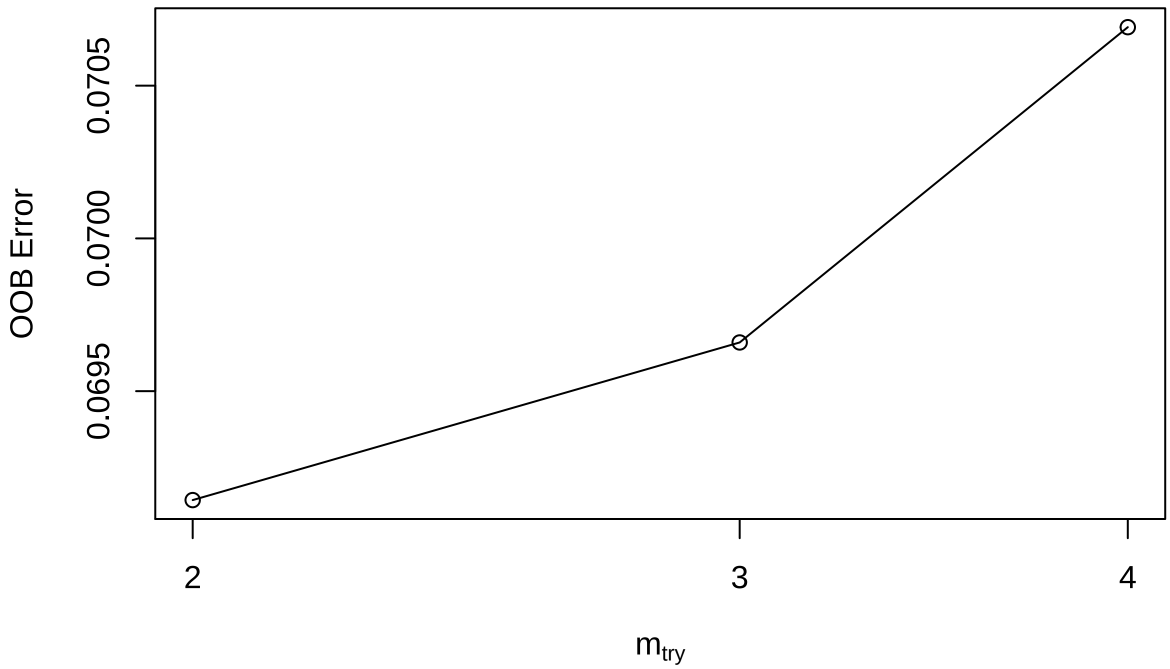




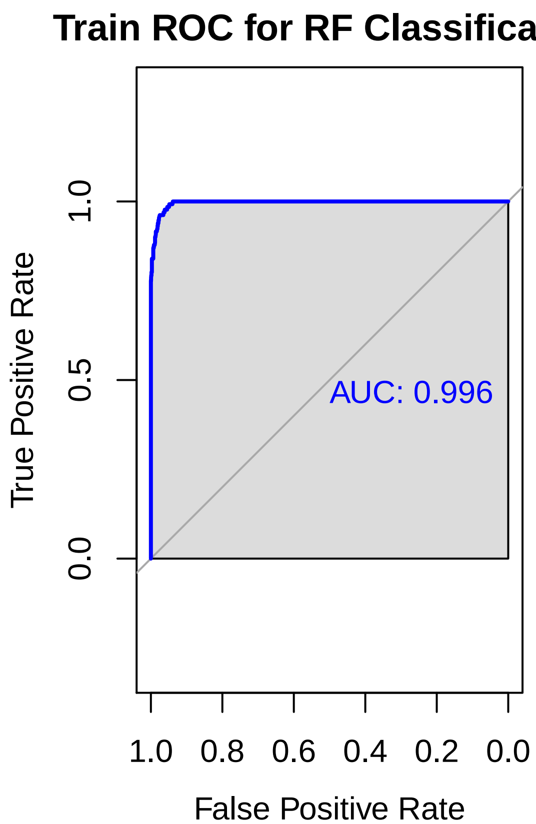
Random Forests Classification

RF Classification BEFORE Variable Selection

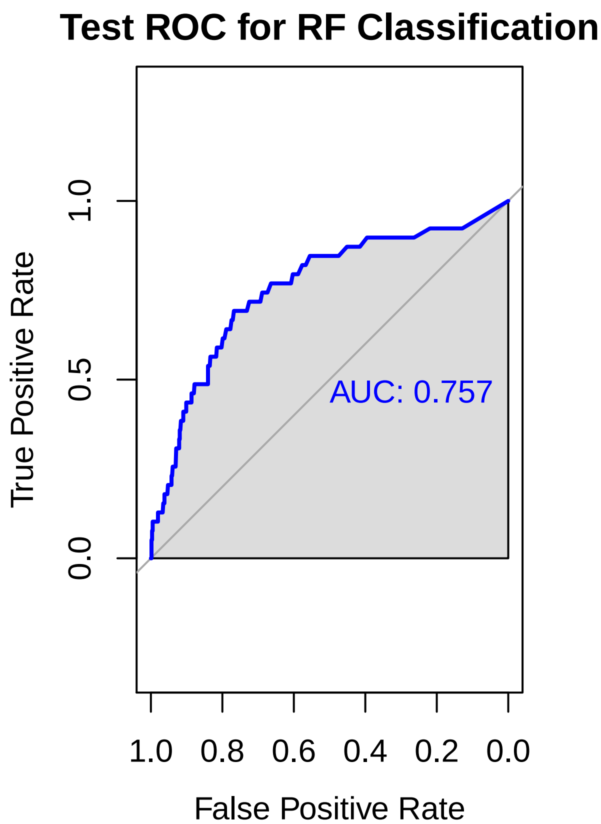
```
mtry = 3  OOB error = 6.97%
Searching left ...
mtry = 2    OOB error = 6.91%
0.007407407 0.01
Searching right ...
mtry = 4    OOB error = 7.07%
-0.01481481 0.01
```

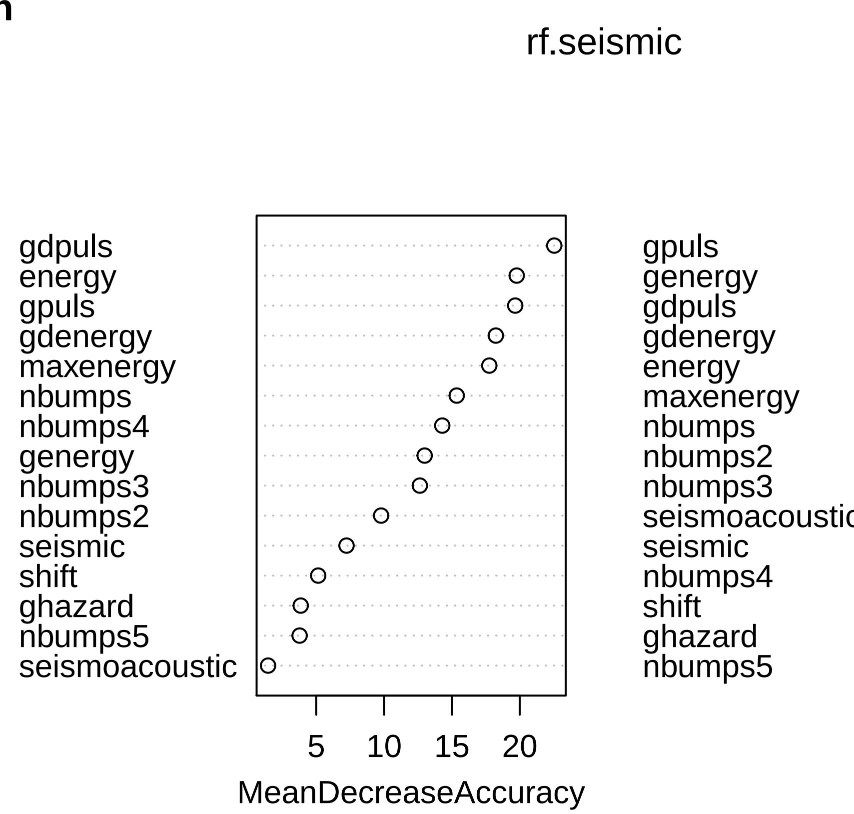



```
[1] 0

[1] 0.2274488

[1] 0.2120743
```

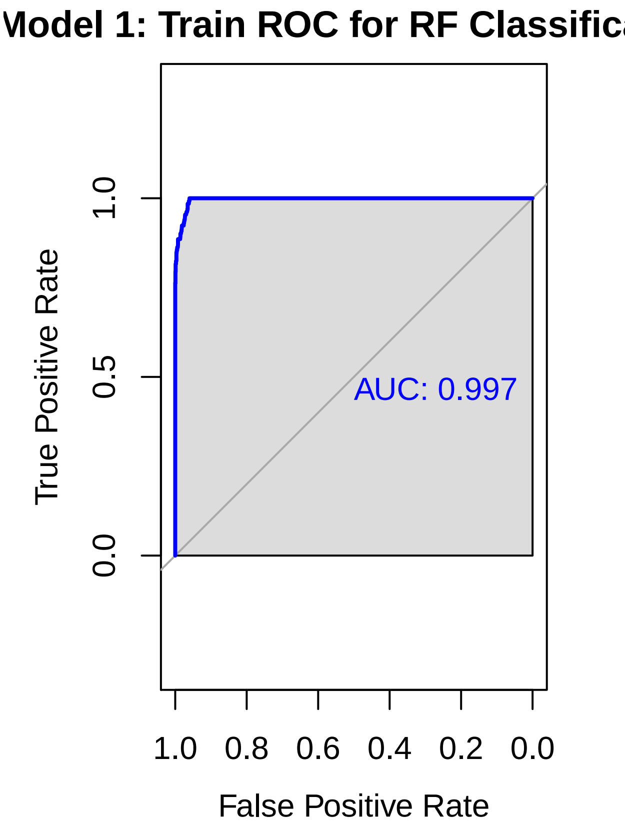


```
[1] 0

[1] 0.1285008

[1] 0.120743

                      0           1 MeanDecreaseAccuracy
seismic        4.942147  6.15726161             7.231775
seismoacoustic 1.638320 -0.03122369             1.447841
shift          4.468044  0.11448538             5.143586
genergy       11.000662  2.95986633            12.989644
gpuls         17.489423  8.47867312            19.665174
gdenergy      20.136379 -8.01313089            18.237778
gdpuls        23.892430 -8.21408587            22.544829
ghazard        5.523303 -4.74394607             3.855002
nbumps        14.795107  2.26357609            15.354170
nbumps2        7.560342  8.01727857             9.782091
nbumps3       11.230001  3.64309300            12.635703
nbumps4       16.289664 -10.49765231           14.289190
nbumps5        4.558268 -3.60950032             3.774679
energy        19.517128 -4.13354759            19.775304
maxenergy     17.385435 -5.66891463            17.755819
               MeanDecreaseGini
seismic               4.2166606
seismoacoustic        4.3595971
shift                 2.5076001
genergy              25.0274156
gpuls                26.1374496
gdenergy             20.9172228
gdpuls               21.3015730
ghazard               1.9574330
nbumps               11.6841679
nbumps2               8.5743069
nbumps3               7.2958276
nbumps4               2.7711741
nbumps5               0.3347043
energy               18.0817341
maxenergy            14.3337436
```

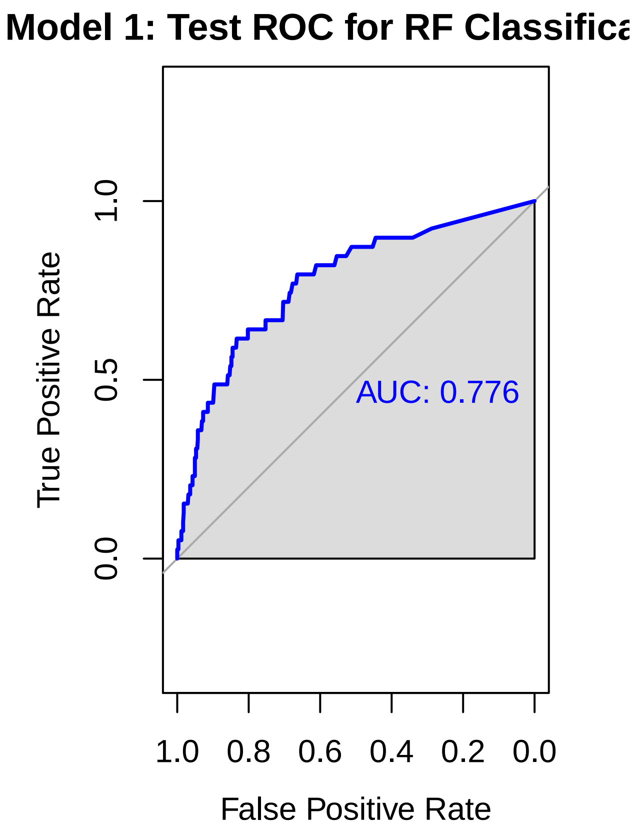

Test ROC for RF Classification
True Positive Rate
0.0
0.5
1.0
AUC: 0.757
1.0
0.8
0.6
0.4
0.2
0.0
False Positive Rate

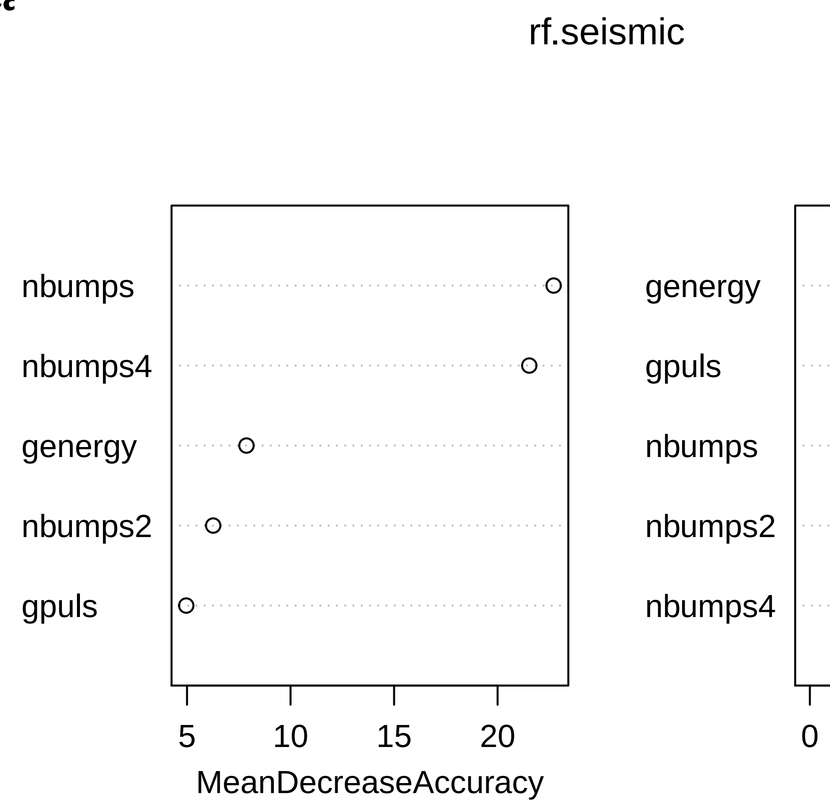

rf.seismic
gdpuls
energy
gpuls
gdenergy
maxenergy
nbumps
nbumps4
genergy
nbumps3
nbumps2
seismic
shift
ghazard
nbumps5
seismoacoustic
5
10
15
20
MeanDecreaseAccuracy
gpuls
genergy
gdpuls
gdenergy
energy
maxenergy
nbumps
nbumps2
nbumps3
seismoacousti
seismic
nbumps4
shift
ghazard
nbumps5

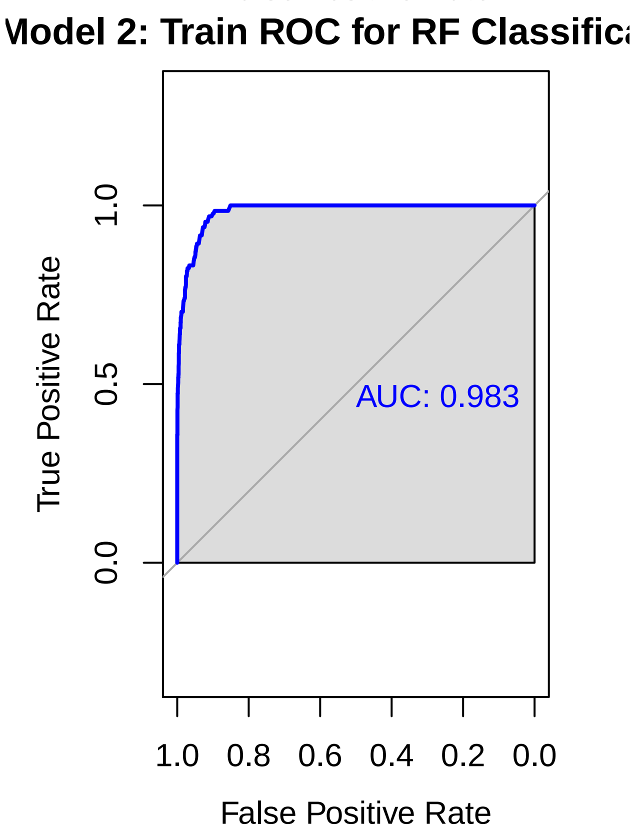
RF Classification AFTER Variable Selection



	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
genergy	5.3648618	6.002146	7.870972	68.670832
gpuls	-0.9452219	21.379661	4.961309	66.667769
nbumps	15.3751653	31.517801	22.706583	25.713553
nbumps2	2.4034881	9.756071	6.262233	14.451982
nbumps4	23.7743816	-7.967046	21.530947	6.908167

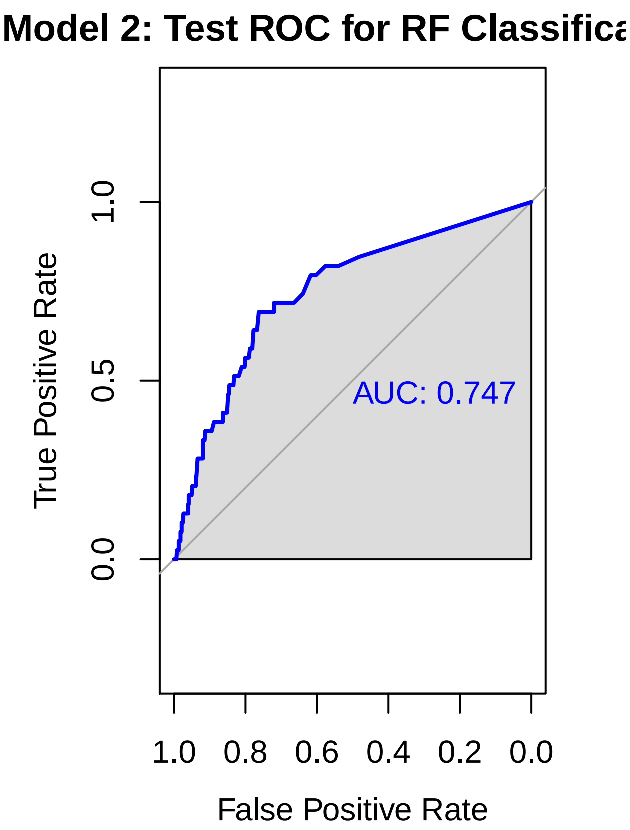


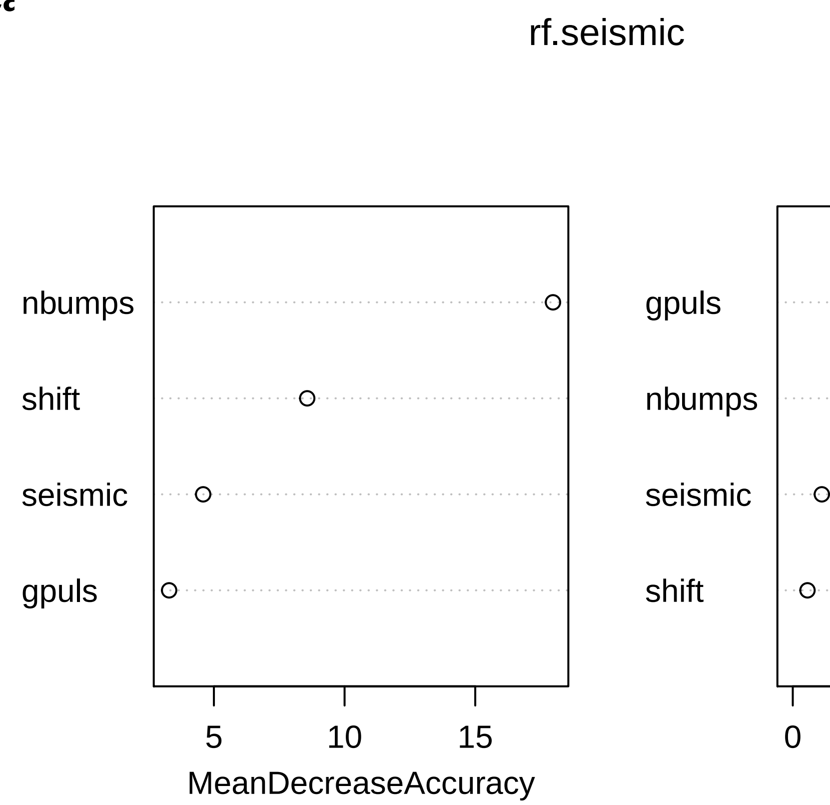





```
0           1 MeanDecreaseAccuracy MeanDecreaseGini
```

seismic	2.128852	7.294074	4.589075	5.767247
shift	10.559603	-10.690270	8.569901	2.923482
gpuls	1.018808	6.321819	3.285896	76.185253
nbumps	12.192454	21.390967	17.974127	27.048122





Support vector classifier and support vector machine

```
# Variable selection and refitting
#model1 = genergy + gpuls + nbumps + nbumps2 + nbumps4 #Step
#model2 = seismic + shift + gpuls + nbumps #Lasso

library(e1071)

# We can modify this by using kernel = radial, which involves changing choice of gamma
# or by choosing kernel = polynomial, where we can also modify the degree
# Using a linear kernel is technically a support vector classifier

#--------------------------------------------------
# Get the ROC curve for svm
library(ROCR)

rocplot <- function(pred, truth, ...){
  predob <- prediction(pred, truth)
  perf <- performance(predob, "tpr", "fpr")
```

```r
  plot(perf,...)
}
#--------------------------------------------------

#--------------------------------------------------
# Start with just the linear kernel
#--------------------------------------------------

##
## Model 1
##

start.time <- proc.time()

tune.out <- tune(svm, factor(class)~genergy + gpuls + nbumps + nbumps2 + nbumps4, data = seismic[train,]

# Look for a best model
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.001
##
## - best performance: 0.06761391
##
## - Detailed performance results:
##    cost      error dispersion
## 1 0.001 0.06761391 0.02215124
## 2 0.010 0.06761391 0.02215124
## 3 0.100 0.06761391 0.02215124
## 4 1.000 0.06761391 0.02215124
## 5 5.000 0.06761391 0.02215124
```

```r
bestmod <- tune.out$best.model
summary(bestmod)
```

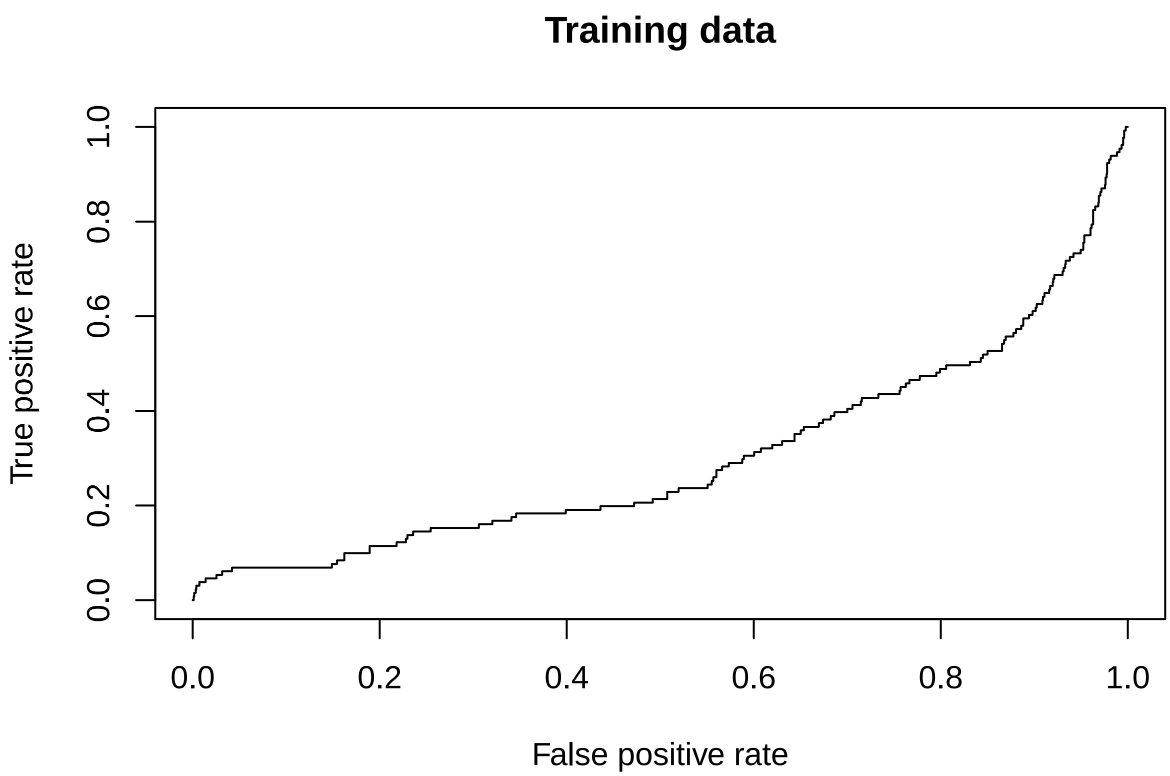
```
##
## Call:
## best.tune(method = svm, train.x = factor(class) ~ genergy + gpuls +
##     nbumps + nbumps2 + nbumps4, data = seismic[train, ], ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.001
```

```
##       gamma:  0.2
##
## Number of Support Vectors:  268
##
##  ( 137 131 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

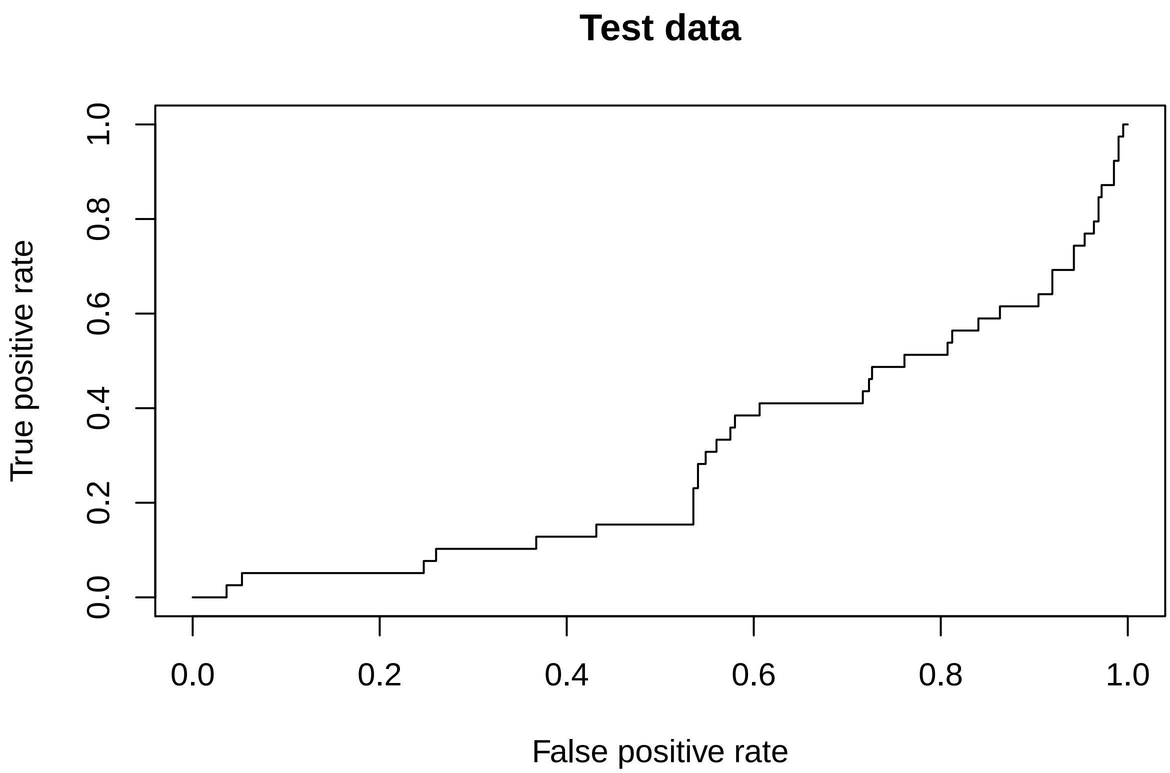
```
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

```
##        truth
## predict   0   1
##       0 607  39
##       1   0   0
```

```
svmfit.best1 <- svm(factor(class)~genergy + gpuls + nbumps + nbumps2 + nbumps4, data = seismic[train,],
fitted1 <- attributes(predict(svmfit.best1, seismic[train,], decision.values = T))$decision.values
fitted.test1 <- attributes(predict(svmfit.best1, seismic[-train,], decision.values = T))$decision.values

# It is unsurprising that this doesn't work well, because we are using a linear classifier
# However, we have reason to believe that a non-linear classifier would be more appropriate
rocplot(fitted1, seismic[train,"class"], main = "Training data")
```



```r
rocplot(fitted.test1, seismic[-train,"class"], main = "Test data")
```




```r
total.time <- proc.time() - start.time
time1 <- total.time[3]

##
## Model 2
##

start.time <- proc.time()

tune.out <- tune(svm, factor(class)~seismic + shift + gpuls + nbumps, data = seismic[train,], kernel =

# Look for a best model
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##   cost
##  0.001
##
## - best performance: 0.06760323
##
## - Detailed performance results:
```

```
##    cost      error dispersion
## 1 0.001 0.06760323 0.01549586
## 2 0.010 0.06760323 0.01549586
## 3 0.100 0.06760323 0.01549586
## 4 1.000 0.06760323 0.01549586
## 5 5.000 0.06760323 0.01549586
```

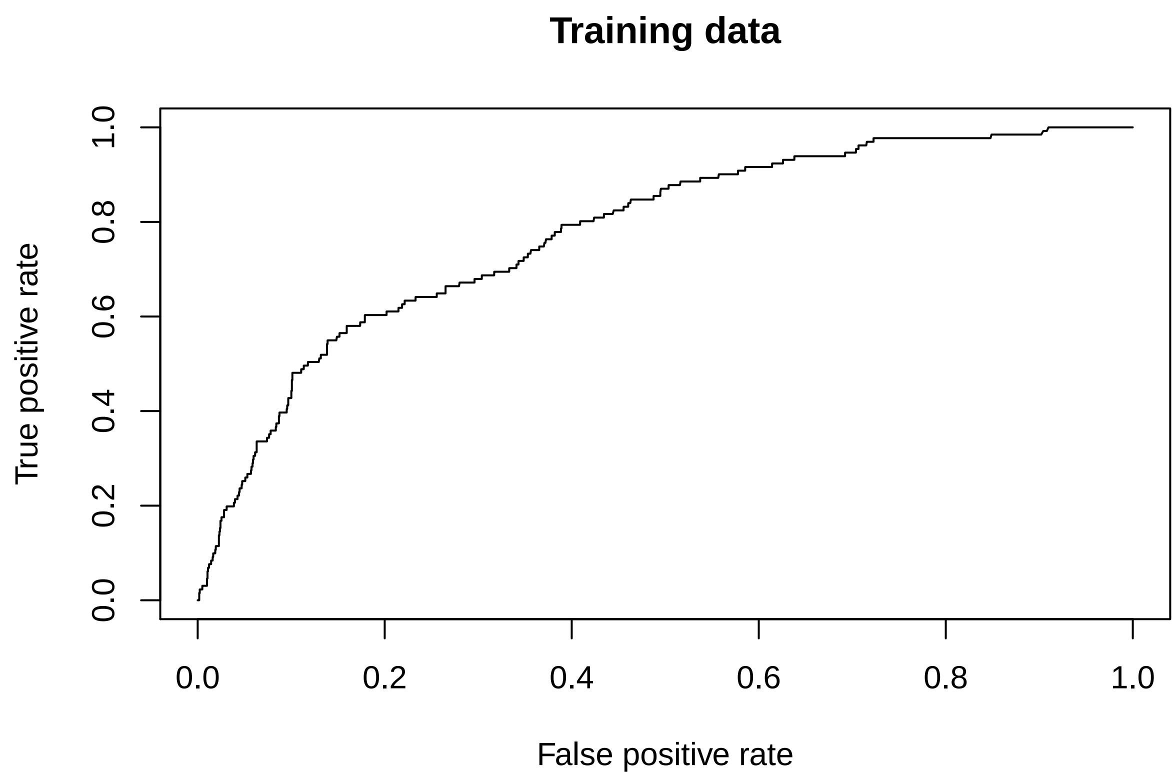
```r
bestmod <- tune.out$best.model
summary(bestmod)
```

```
##
## Call:
## best.tune(method = svm, train.x = factor(class) ~ seismic + shift +
##     gpuls + nbumps, data = seismic[train, ], ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.001
##       gamma:  0.25
##
## Number of Support Vectors:  265
##
##  ( 134 131 )
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

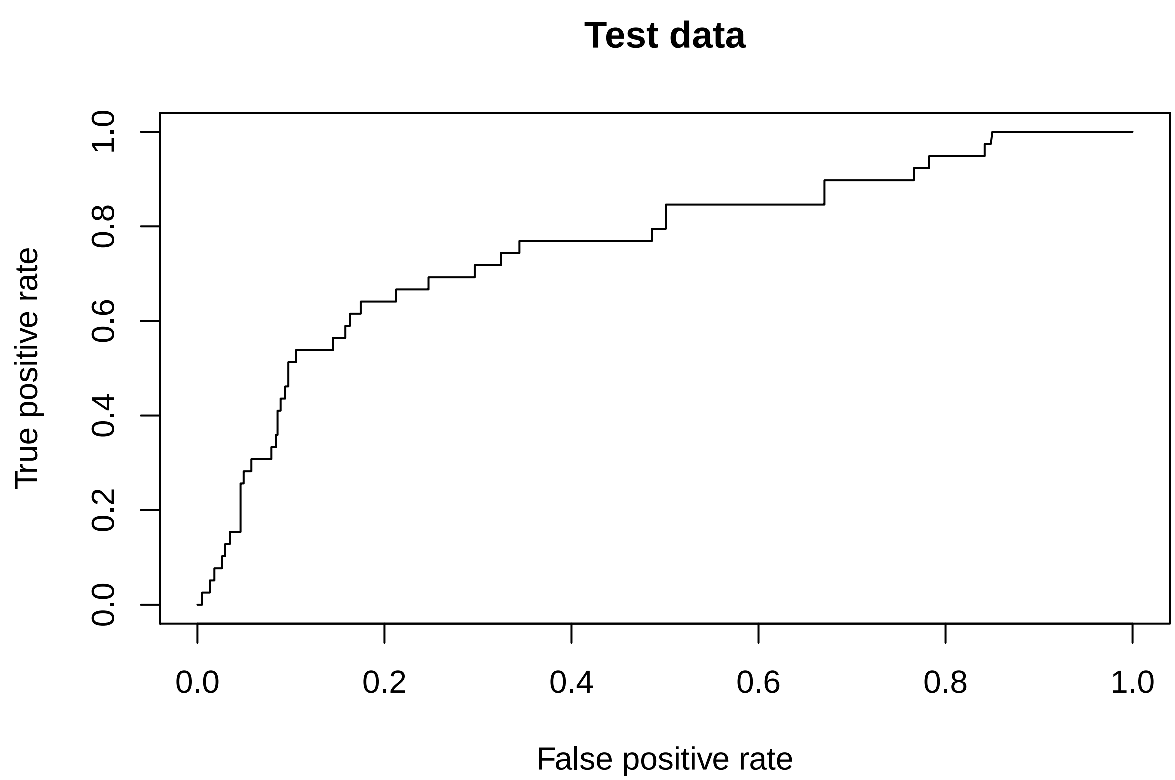
```r
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

```
##        truth
## predict   0   1
##       0 607  39
##       1   0   0
```

```r
svmfit.best2 <- svm(factor(class)~seismic + shift + gpuls + nbumps, data = seismic[train,], kernel = "li
fitted2 <- attributes(predict(svmfit.best2, seismic[train,], decision.values = T))$decision.values
fitted.test2 <- attributes(predict(svmfit.best2, seismic[-train,], decision.values = T))$decision.values

# This one shows a much better ROC curve
# But it still looks bad just from the original table produced
rocplot(fitted2, seismic[train,"class"], main = "Training data")
```



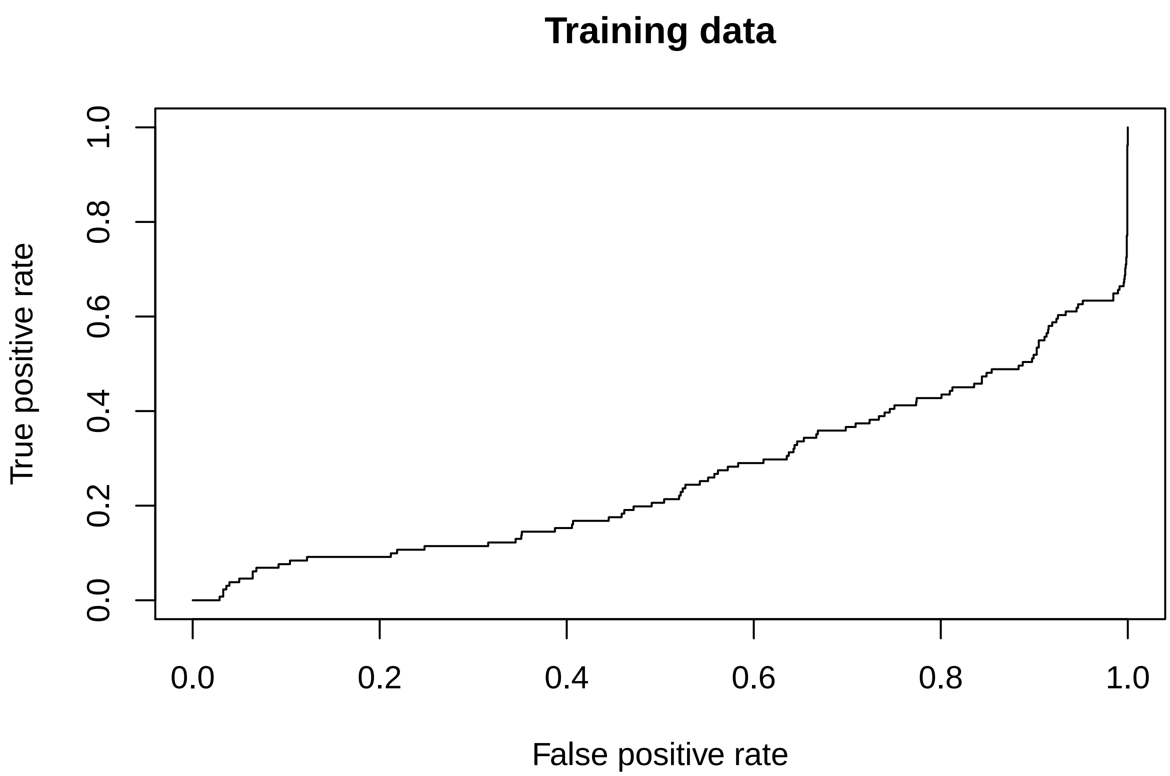

```r
rocplot(fitted.test2, seismic[-train,"class"], main = "Test data")
```



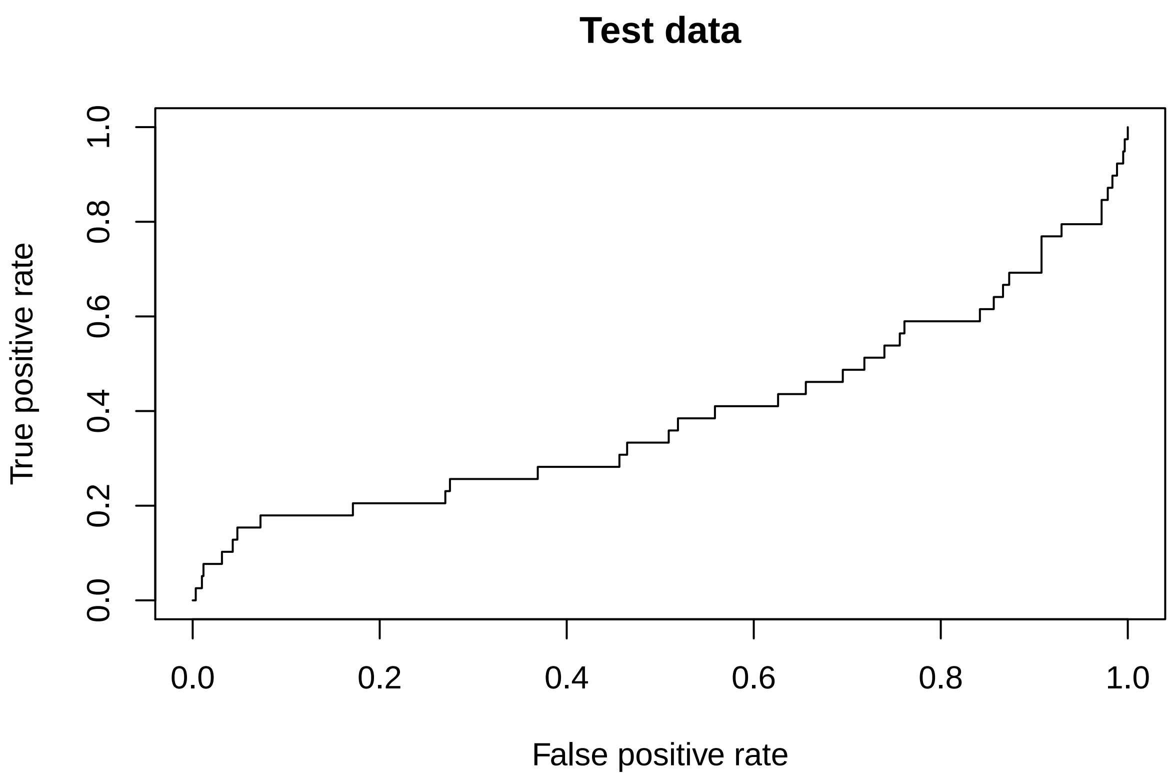
```r
total.time <- proc.time() - start.time
time2 <- total.time[3]

#---------------------------------------------------
# Implement with the radial kernel
#---------------------------------------------------

##
## Model 1
##

start.time <- proc.time()

tune.out2 <- tune(svm, factor(class)~genergy + gpuls + nbumps + nbumps2 + nbumps4, data = seismic[train

bestmod <- tune.out2$best.model
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

```
##        truth
## predict   0   1
##       0 607  39
##       1   0   0
```

```r
svmrad2 <- svm(factor(class)~genergy + gpuls + nbumps + nbumps2 + nbumps4, data = seismic[train,], kern
fitted2 <- attributes(predict(svmrad2, seismic[train,], decision.values = T))$decision.values
fitted.test2 <- attributes(predict(svmrad2, seismic[-train,],decision.values = T))$decision.values

rocplot(fitted2, seismic[train,"class"], main = "Training data")
```



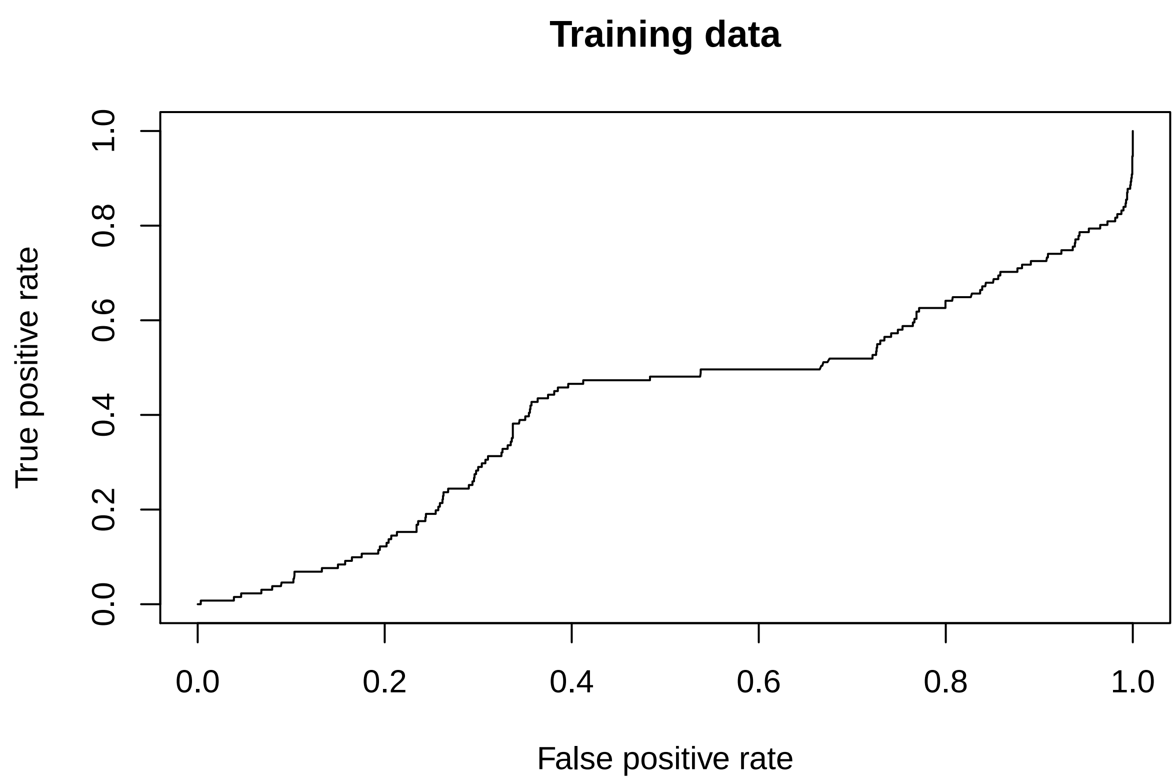

```
rocplot(fitted.test2, seismic[-train,"class"], main = "Test data")
```



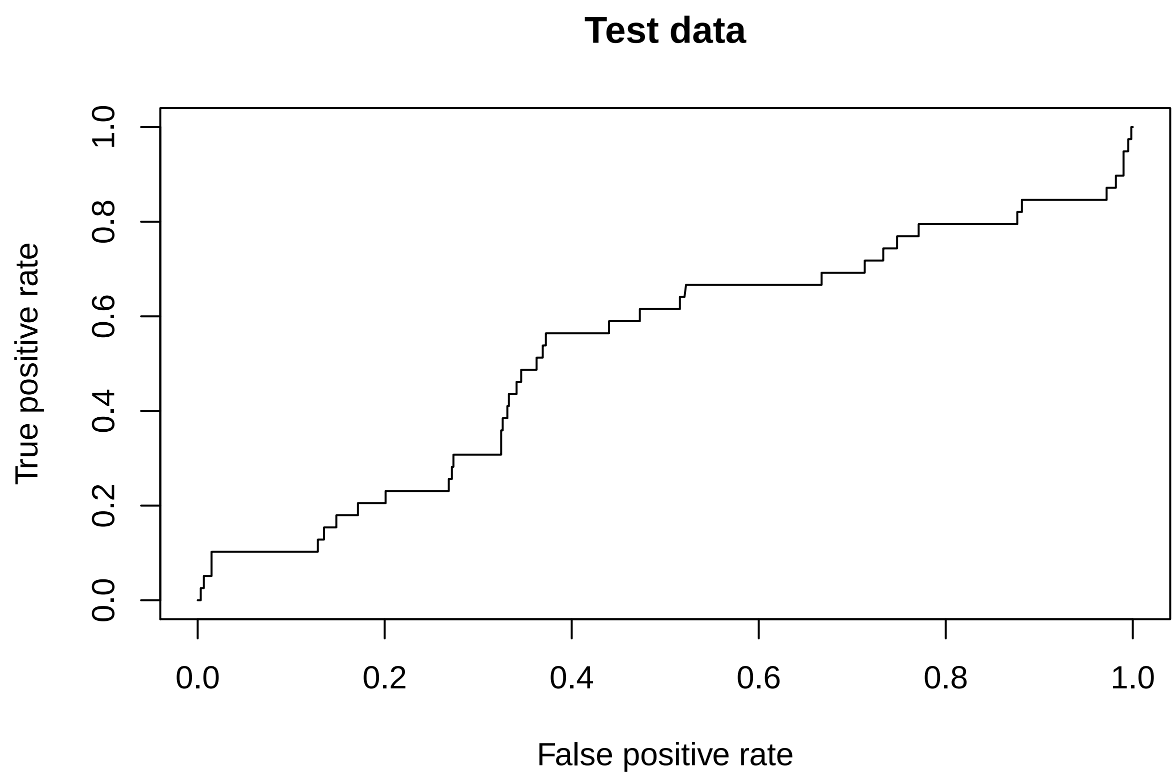
```
total.time <- proc.time() - start.time
time3 <- total.time[3]

##
## Model 2
##

start.time <- proc.time()

tune.out3 <- tune(svm, factor(class)~seismic + shift + gpuls + nbumps, data = seismic[train,], kernel =

bestmod <- tune.out3$best.model
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

```
##          truth
## predict   0   1
##       0 607  39
##       1   0   0
```

```
svmrad3 <- svm(factor(class)~seismic + shift + gpuls + nbumps, data = seismic[train,], kernel = "radial
fitted3 <- attributes(predict(svmrad3, seismic[train,], decision.values = T))$decision.values
fitted.test3 <- attributes(predict(svmrad3, seismic[-train,],decision.values = T))$decision.values

rocplot(fitted3, seismic[train,"class"], main = "Training data")
```



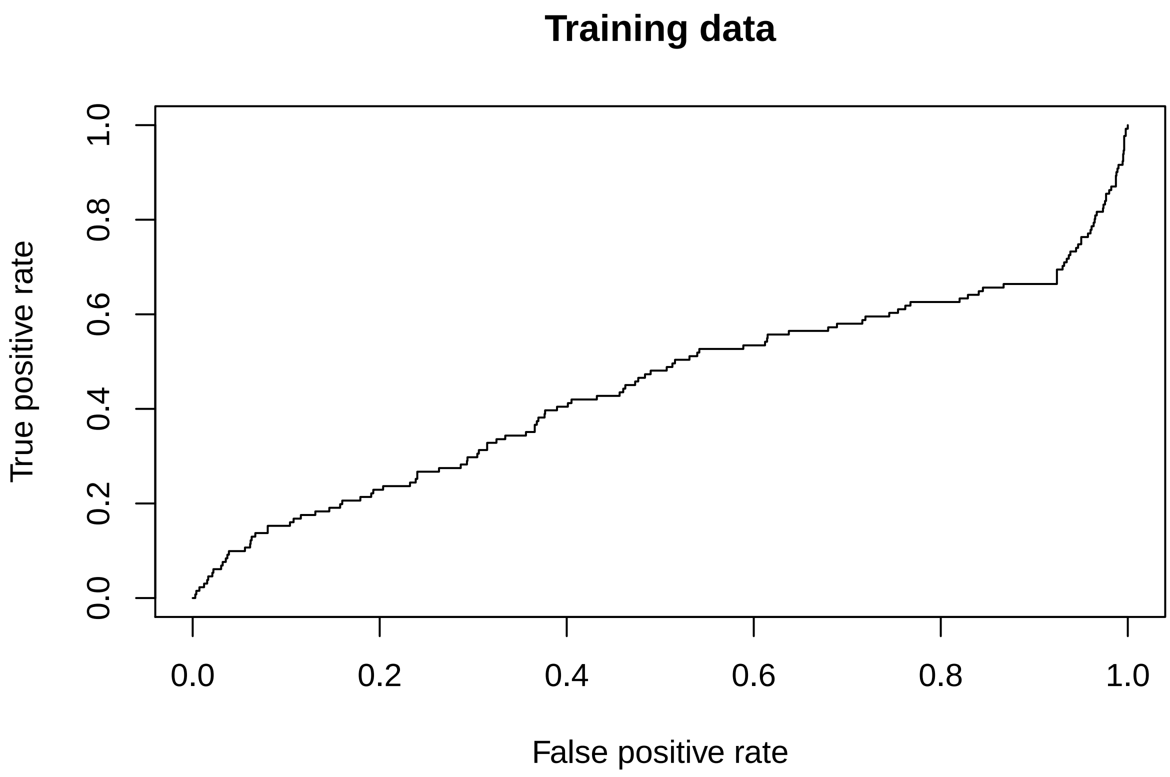
```r
rocplot(fitted.test3, seismic[-train,"class"], main = "Test data")
```



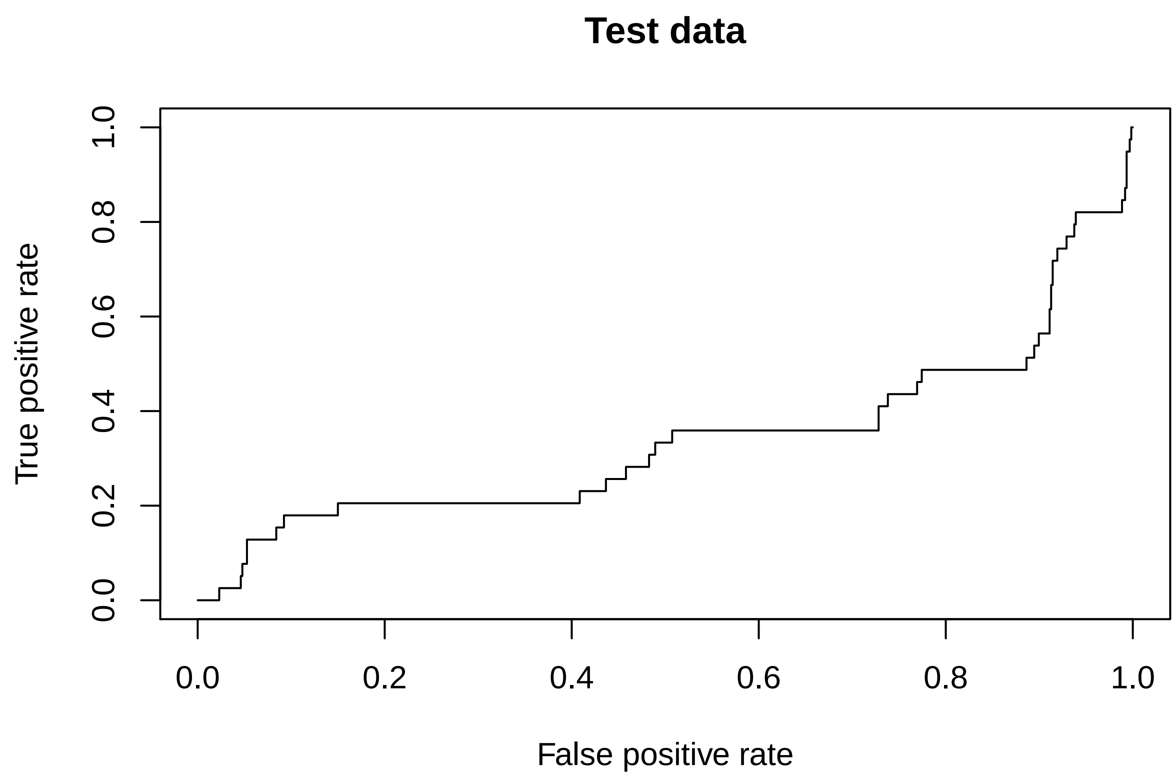

```r
total.time <- proc.time() - start.time
time4 <- total.time[3]

#--------------------------------------------------
# Implement with the polynomial kernel
#--------------------------------------------------

##
## Model 1
##

start.time <- proc.time()

tune.out4 <- tune(svm, factor(class)~genergy + gpuls + nbumps + nbumps2 + nbumps4, data = seismic[train

bestmod <- tune.out4$best.model
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

```
##        truth
## predict   0   1
##       0 607  39
##       1   0   0
```

```
svmpoly4 <- svm(factor(class)~genergy + gpuls + nbumps + nbumps2 + nbumps4, data = seismic[train,], kern
fitted4 <- attributes(predict(svmpoly4, seismic[train,], decision.values = T))$decision.values
fitted.test4 <- attributes(predict(svmpoly4, seismic[-train,],decision.values = T))$decision.values

rocplot(fitted4, seismic[train,"class"], main = "Training data")
```



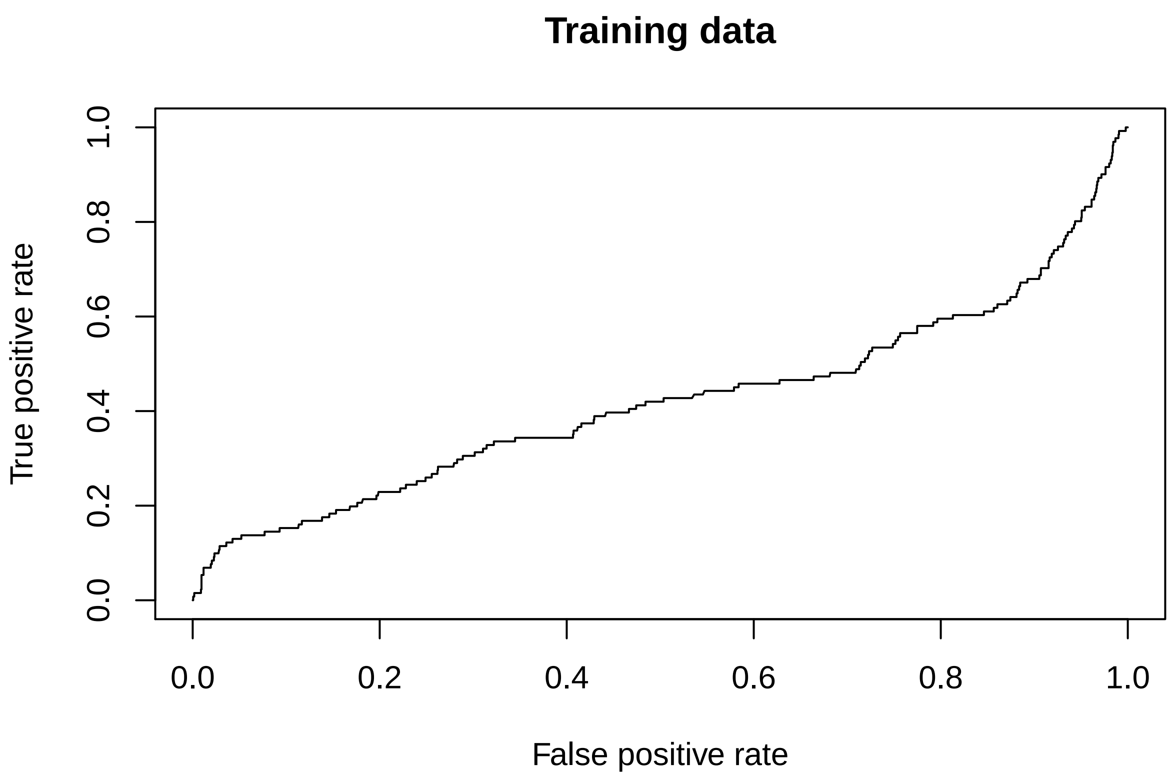

```
rocplot(fitted.test4, seismic[-train,"class"], main = "Test data")
```



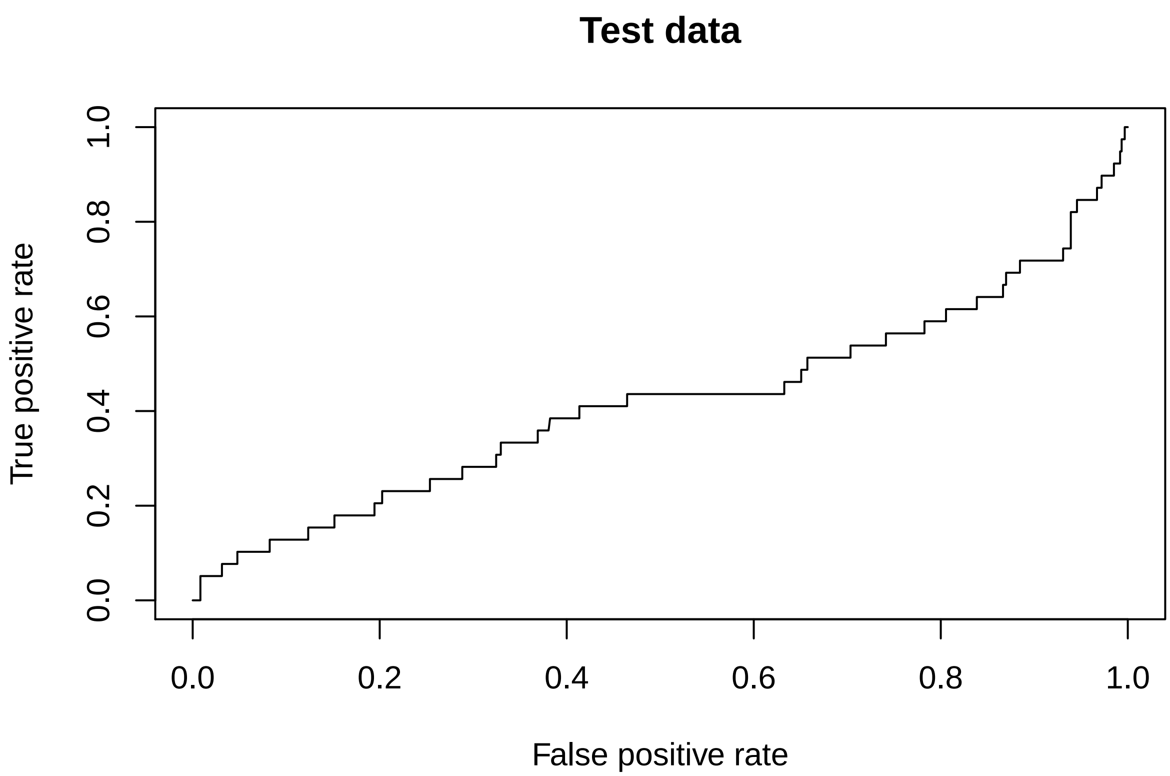

```
total.time <- proc.time() - start.time
time5 <- total.time[3]

##
## Model 2
##

start.time <- proc.time()

tune.out5 <- tune(svm, factor(class)~seismic + shift + gpuls + nbumps, data = seismic[train,], kernel =

bestmod <- tune.out5$best.model
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

```
##          truth
## predict   0   1
##       0 607  39
##       1   0   0
```

```
svmpoly5 <- svm(factor(class)~seismic + shift + gpuls + nbumps, data = seismic[train,], kernel = "polyno
fitted5 <- attributes(predict(svmpoly5, seismic[train,], decision.values = T))$decision.values
fitted.test5 <- attributes(predict(svmpoly5, seismic[-train,],decision.values = T))$decision.values

rocplot(fitted5, seismic[train,"class"], main = "Training data")
```




```r
rocplot(fitted.test5, seismic[-train,"class"], main = "Test data")
```



```r
total.time <- proc.time() - start.time
time6 <- total.time[3]
```

How to time your code!!!

```r
#-------------------------------------
# How to time your method
#-------------------------------------

# Put this before your method
start.time <- proc.time()

## the thing you are computing, like random forest or SVM goes here ##

total.time <- proc.time() - start.time

total.time[3] # the elapsed time
```

```
## elapsed
##   0.001
```