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
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

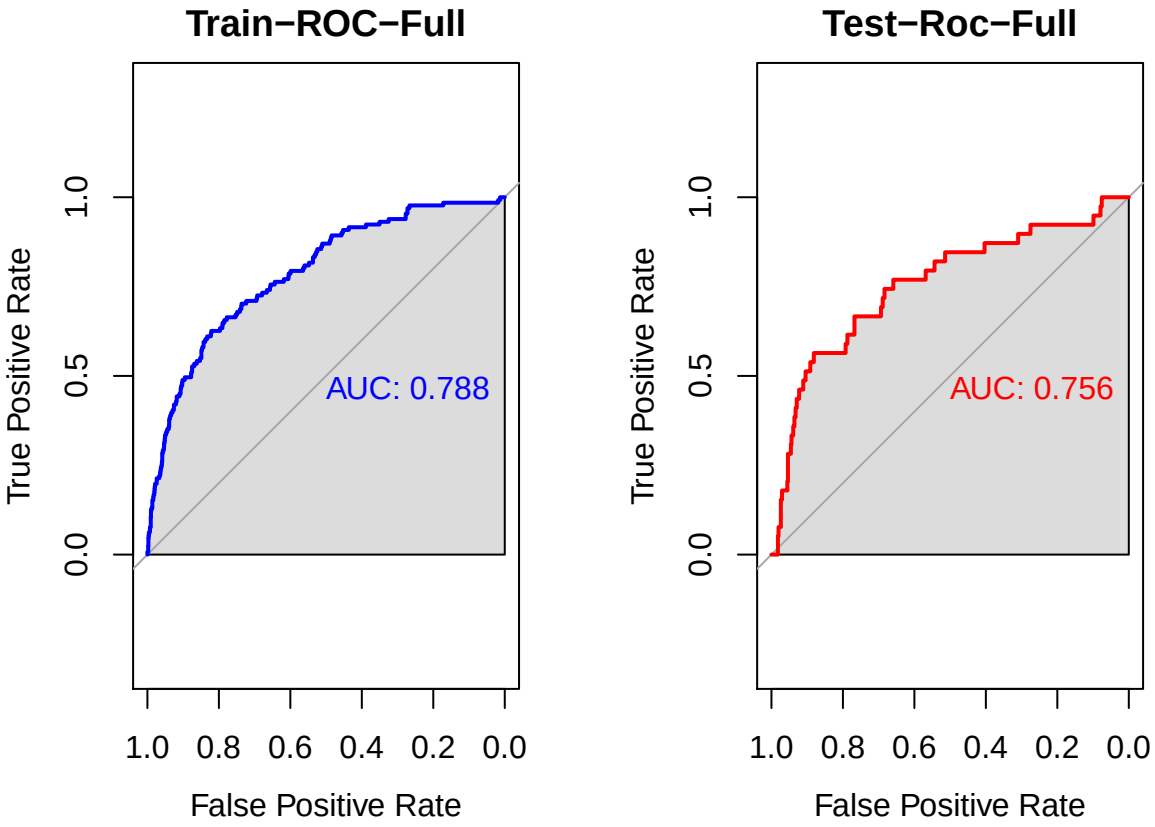
Classification before Variable Selection

multicollinearity (VIF > 10) between some of the covariates, as shown below.

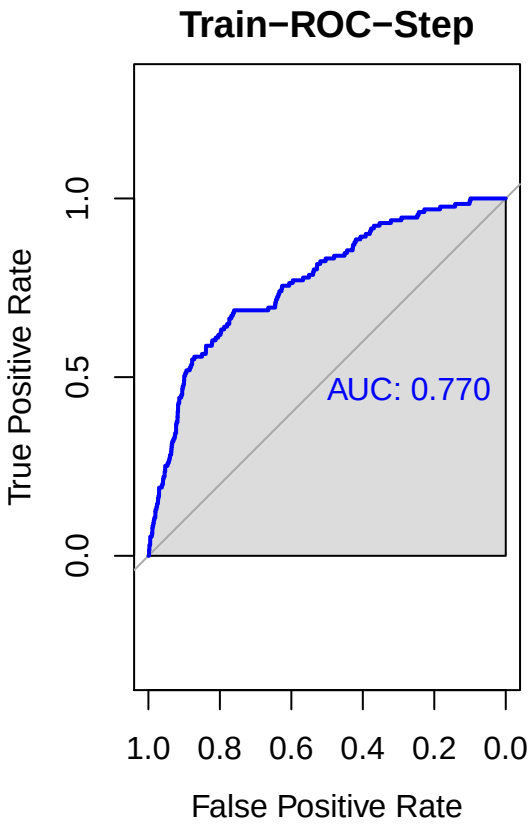
We first take the seismic-bumps dataset and partition the data into training (75%) and test (25%) datasets. The next steps involve examining multiple classification methods on the training and test datasets separately. The goal is to examine which classification method outputs comparatively better prediction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

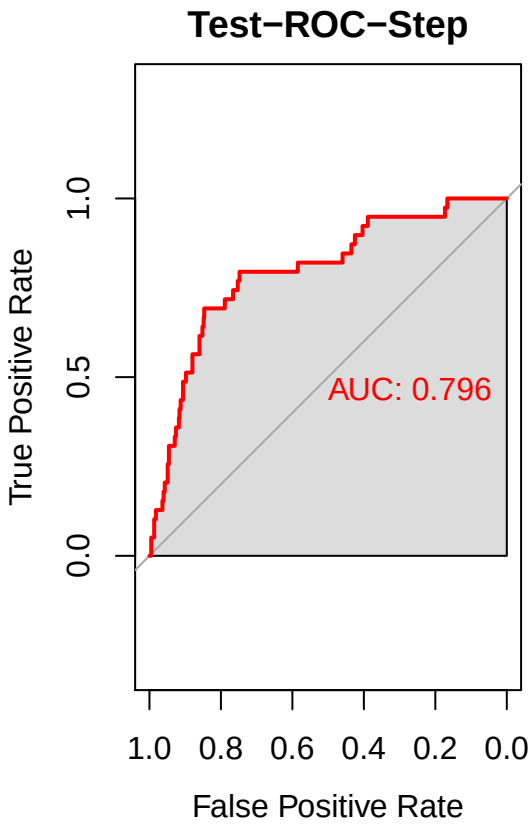
Logistic Regression

Full Model

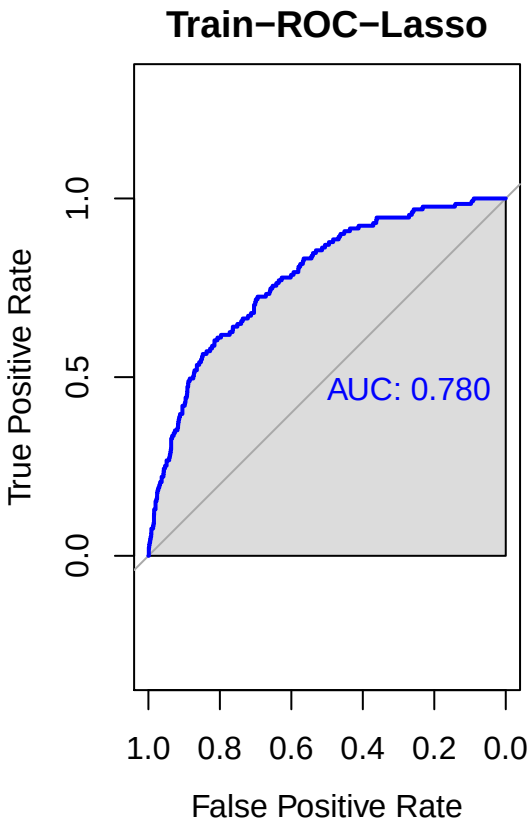


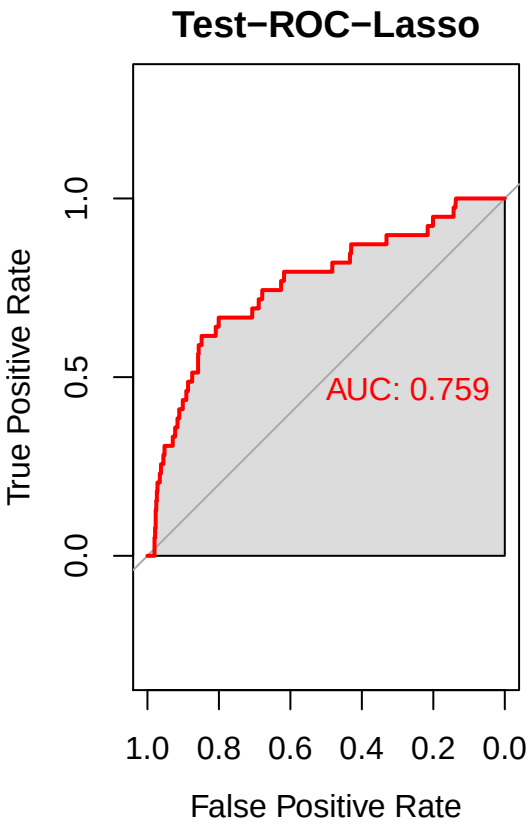
Logistic Regression - Step Model





Logistic Regression - Lasso Model





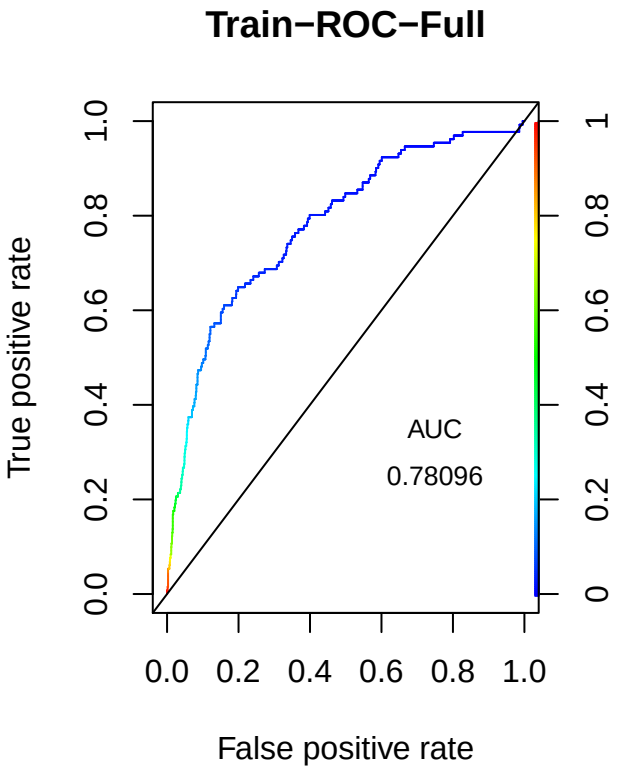

```
        time1 time2 time3
elapsed 0.125 0.206 0.078
```

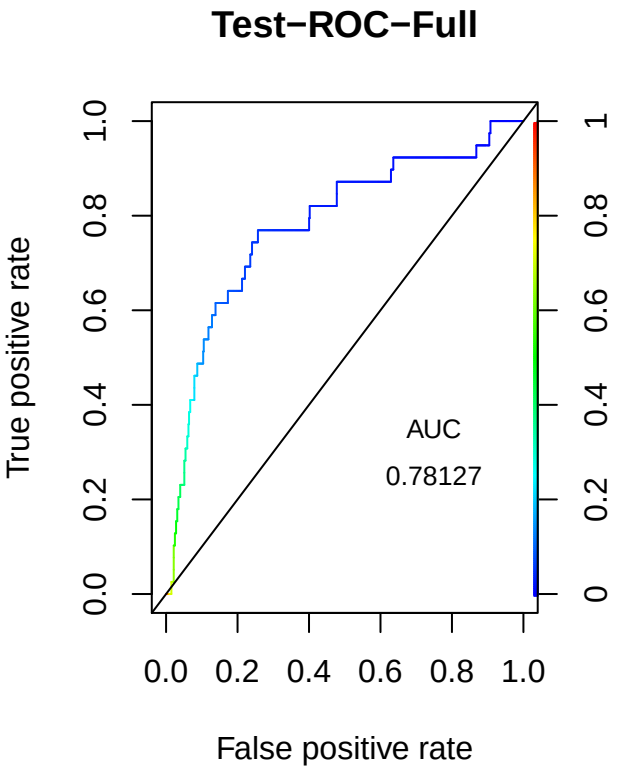
```
     rate1.train rate3.train rate5.train
[1,]       0.067        0.07       0.068
```

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

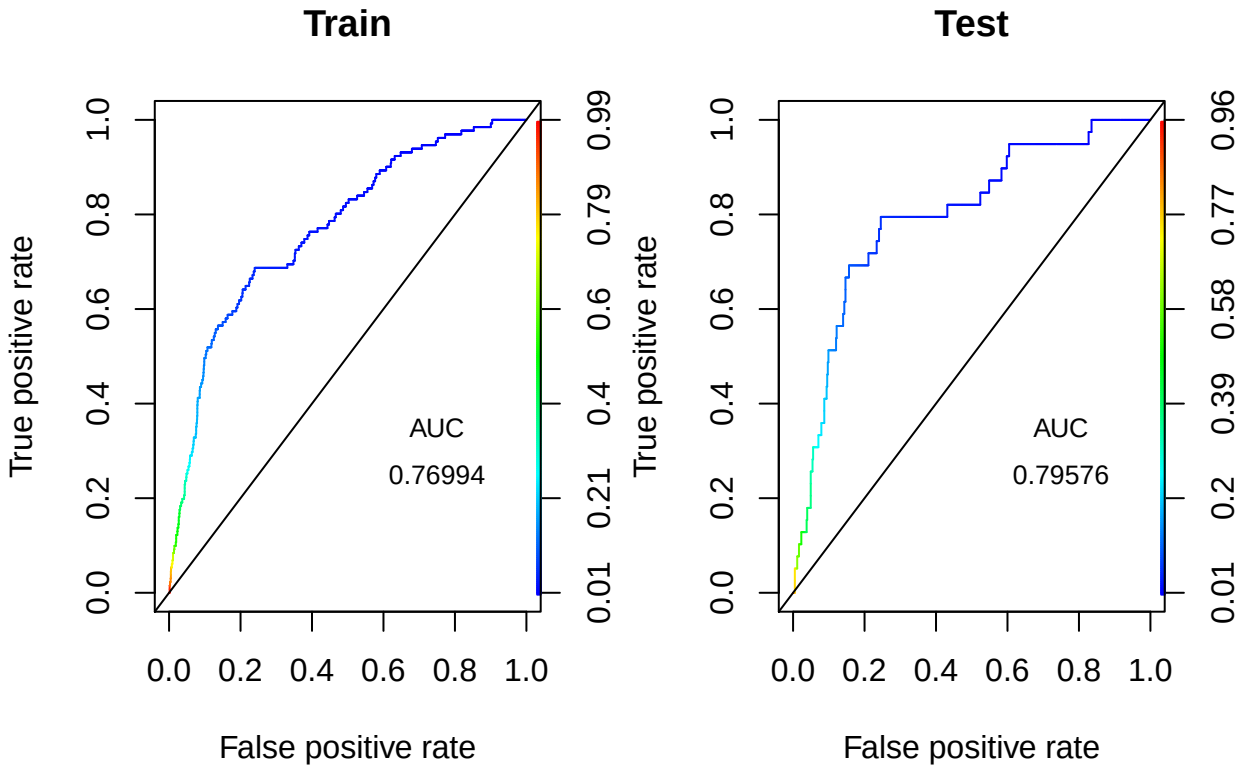
Linear Discriminant Analysis

Full Model

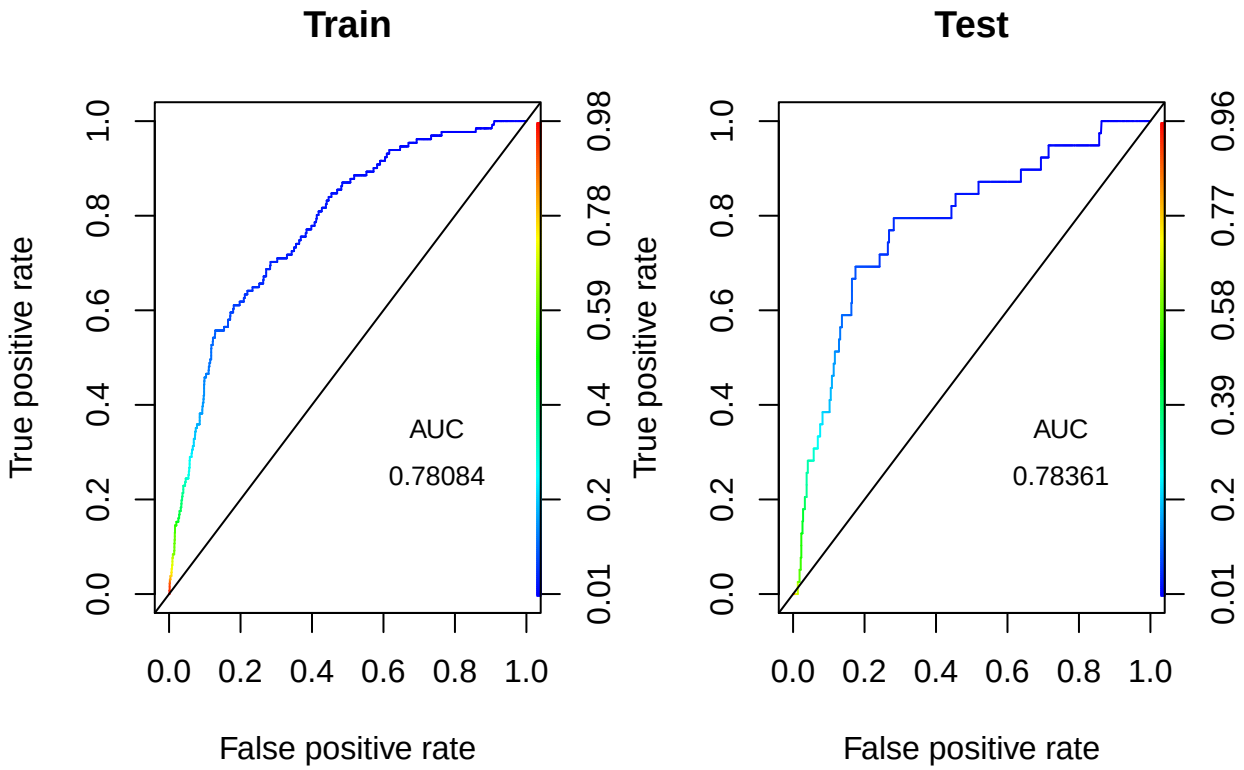




Linear Discriminant Analysis - Step



Linear Discriminant Analysis - Lasso



```
        time1 time2 time3
elapsed 0.862 0.808 0.807
```

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

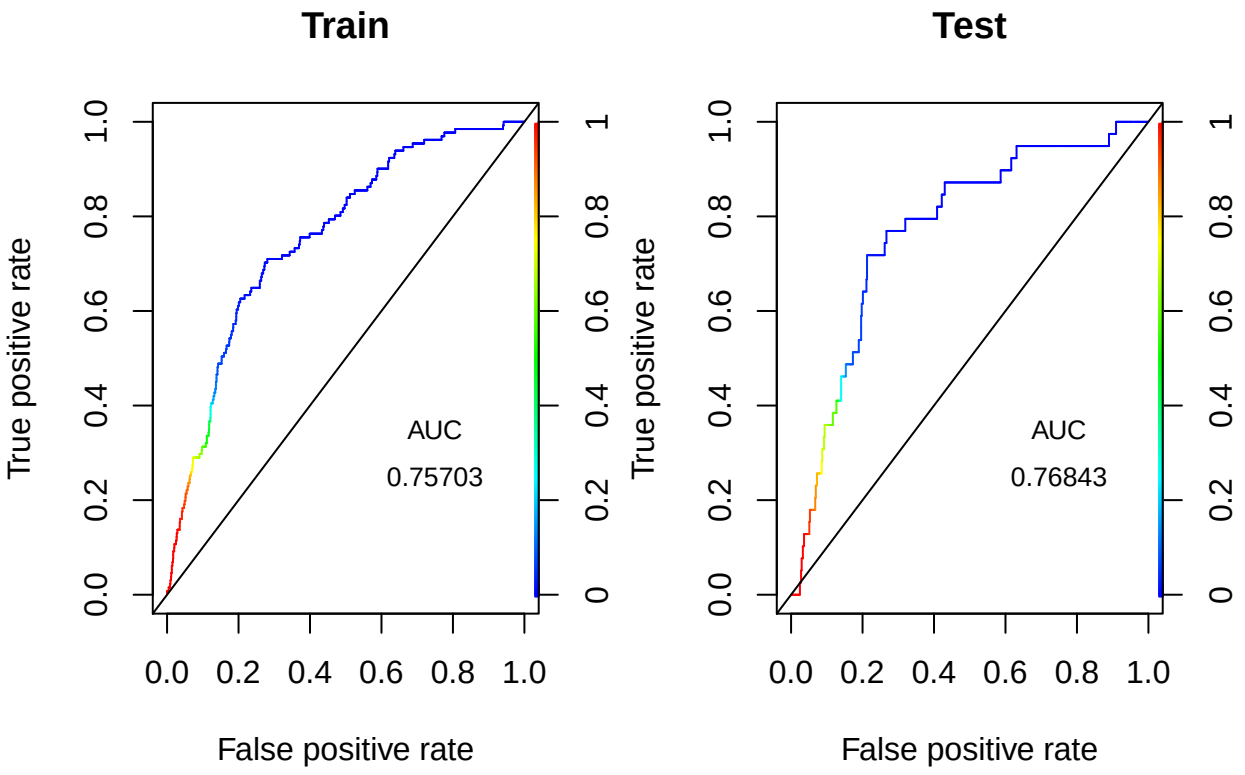
```
     rate2.test rate4.test rate6.test
[1,]      0.077      0.076      0.076
```

Quadratic Discriminant Analysis

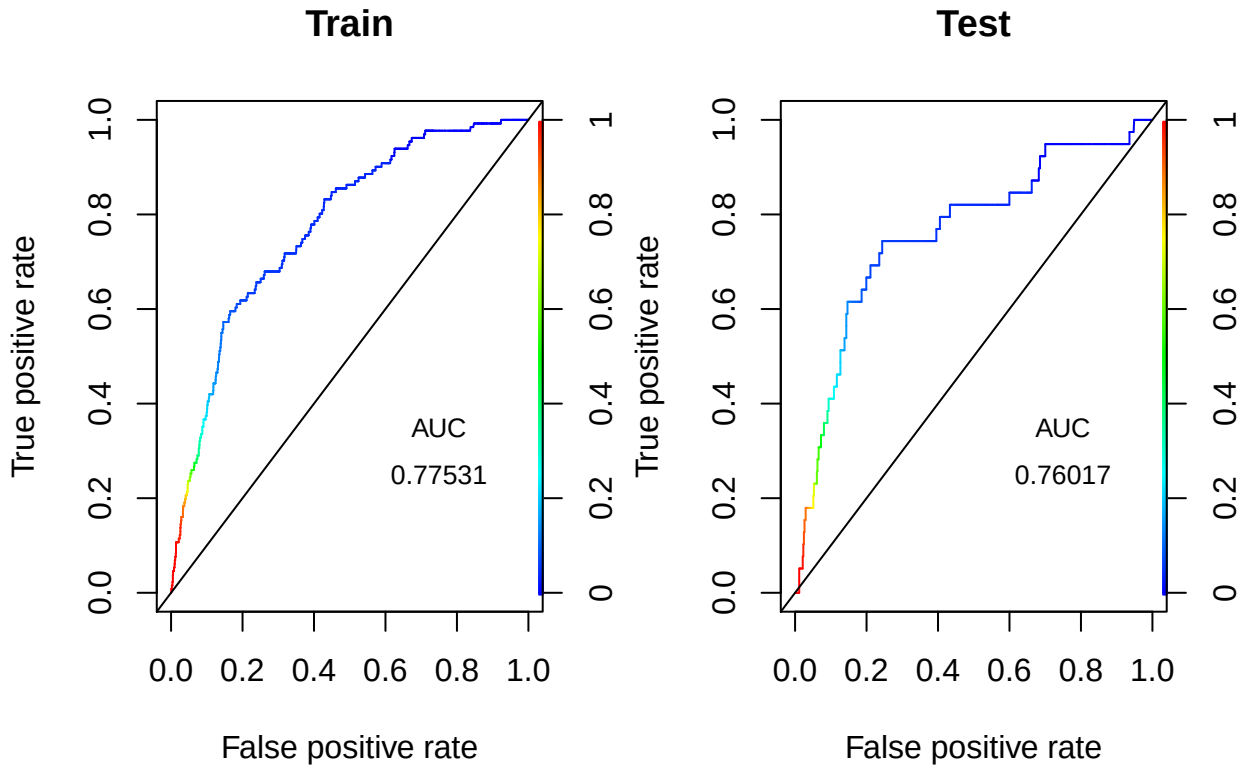
Full Model

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

Quadratic Discriminant Analysis - Step



Quadratic Discriminant Analysis - LASSO

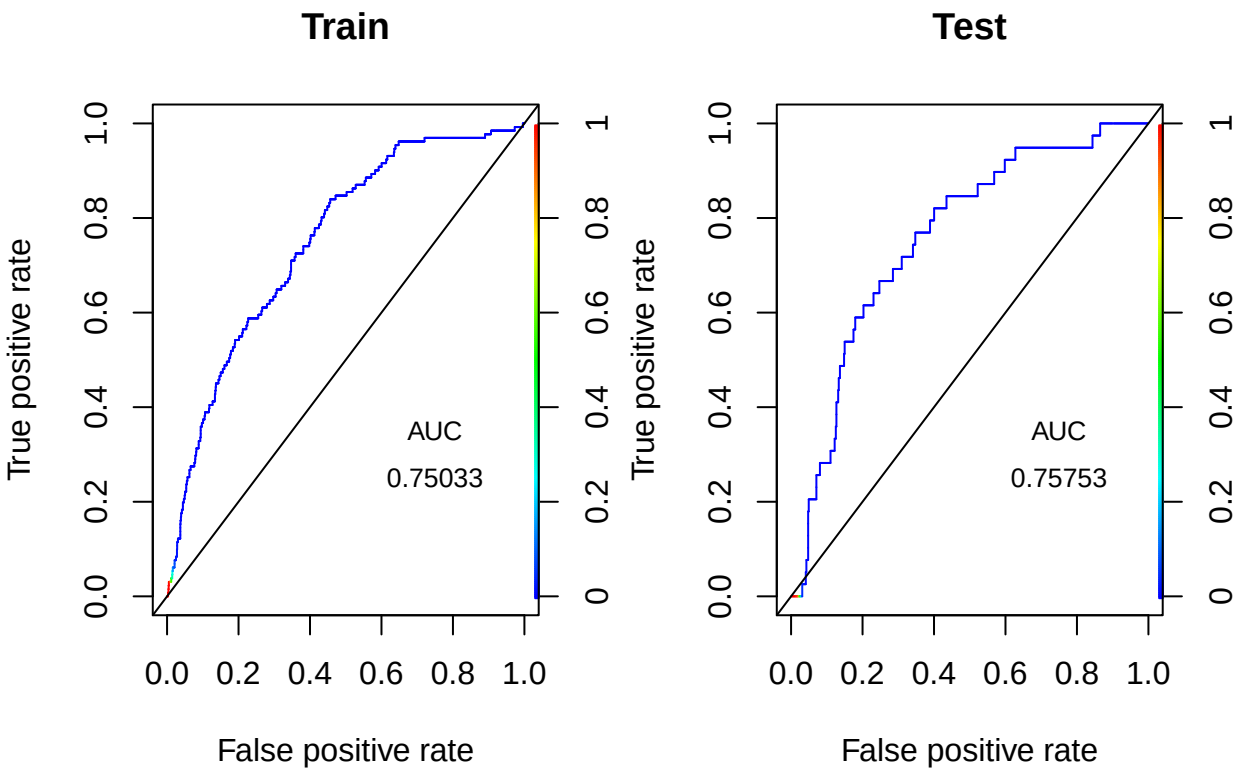



```
        time1 time2
elapsed 0.927 0.848
```

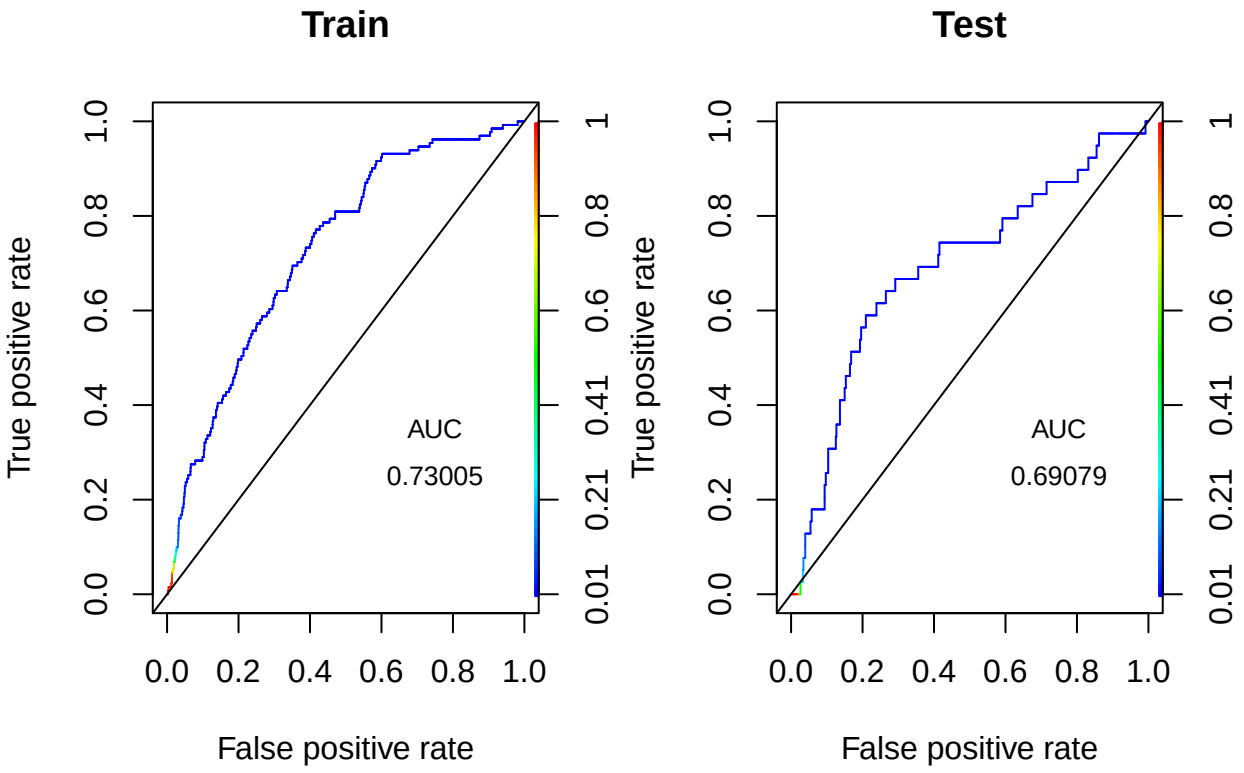
```
     rate1.train rate3.train rate5.train
[1,]       0.149       0.109       0.077
```

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

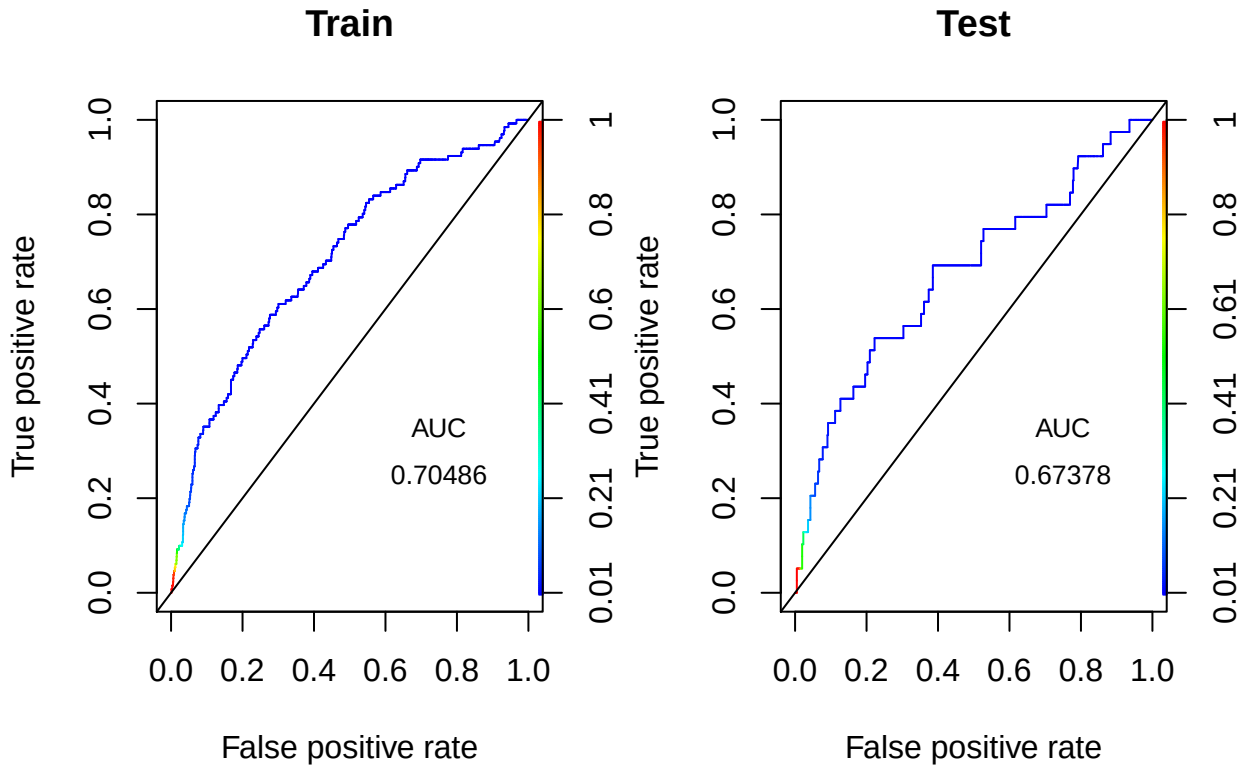
Regularized



Regularized Discriminant Analysis - Step



Regularized Discriminant Analysis - Step




```
        time1 time2
elapsed 3.497  1.62
```

```
     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
```

Random Forests Classifier

```
[1] 0.7251908
```

```
[1] 1
```

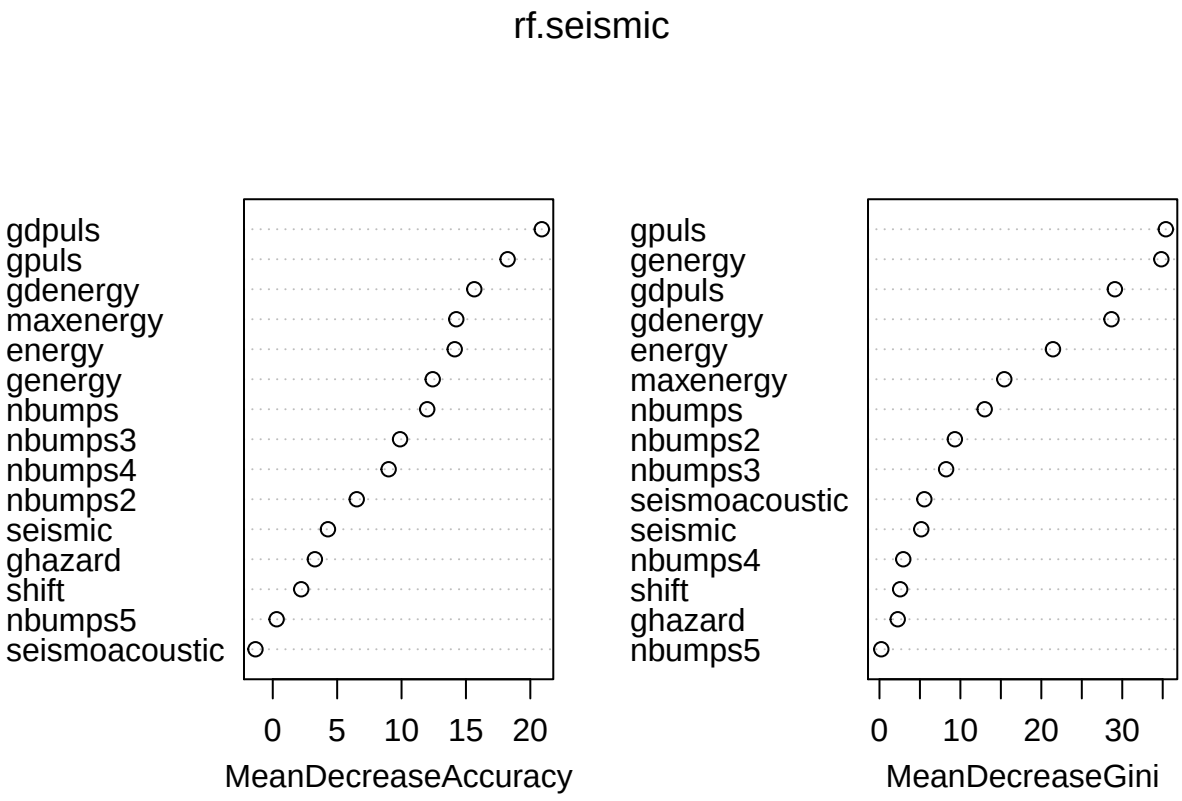
```
[1] 0.9814241
```

```
[1] 0.05128205
```

```
[1] 0.9967051
```

```
[1] 0.9396285
```

```
                     0         1 MeanDecreaseAccuracy MeanDecreaseGini
seismic          3.2423956  3.150342           4.2836798        5.1658789
seismoacoustic -0.2140636 -2.874863          -1.3453087        5.5443467
shift           2.2987072 -0.651549           2.2099663        2.5598187
genergy        12.0834949 -1.716464          12.4196823       34.8289273
gpuls          16.1967974  8.786552          18.2364527       35.3860259
gdenergy       16.8659522 -5.367971          15.6488467       28.6674898
gdpuls         21.7314330 -6.170429          20.8924846       29.0875596
ghazard         4.0067012 -2.243642           3.2692239        2.2569330
nbumps         11.6339632  1.348905          11.9932937       13.0017641
nbumps2         5.1541517  4.920132           6.5203257        9.3162064
nbumps3         9.0714791  2.047936           9.8848601        8.2345298
nbumps4        10.2972021 -7.715771           8.9967774        2.9418042
nbumps5         0.6613005 -1.412860           0.3007354        0.2264114
energy         14.0883490 -4.594831          14.1277273       21.4397149
maxenergy      14.0344626 -5.412141          14.2499697       15.4212607
```



boosting

Support vector classifier and support vector machine

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

```r
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")
  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: 0.06759788
##
## - Detailed performance results:
##    cost      error dispersion
## 1 0.001 0.06759788 0.01983582
## 2 0.010 0.06759788 0.01983582
## 3 0.100 0.06759788 0.01983582
## 4 1.000 0.06759788 0.01983582
## 5 5.000 0.06759788 0.01983582
```

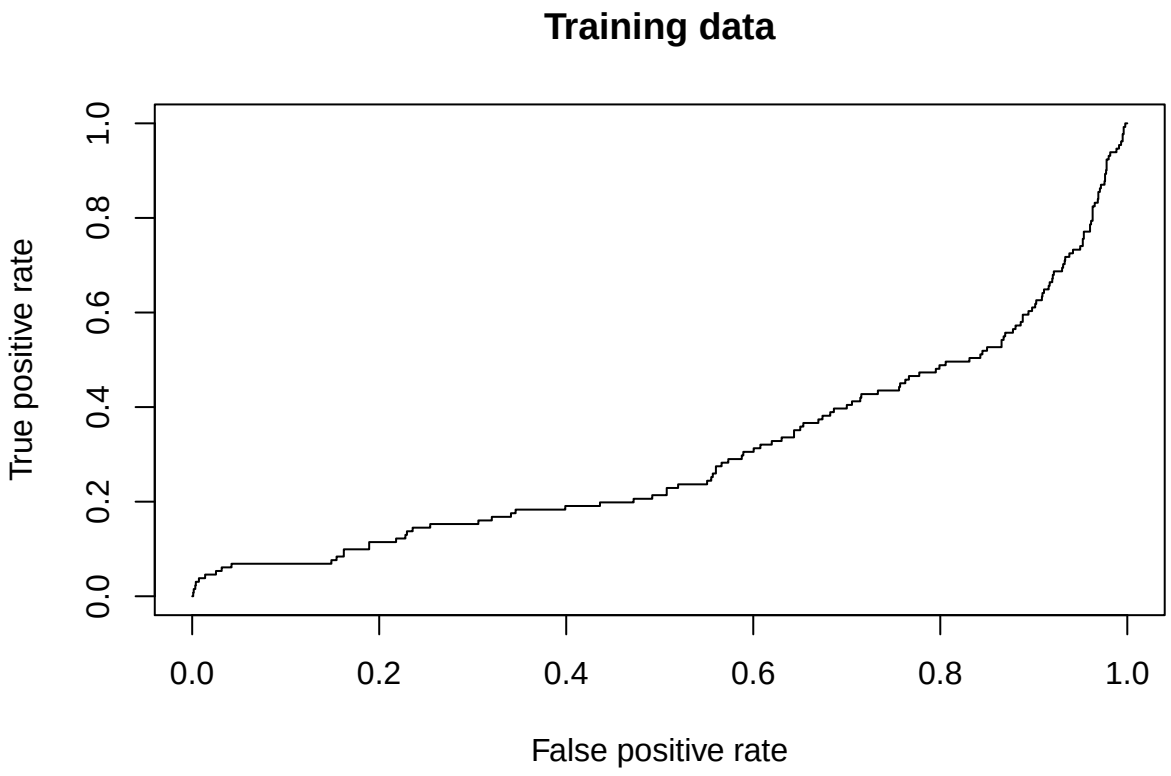
```r
bestmod <- tune.out$best.model
summary(bestmod)
```

```
##
## 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
```

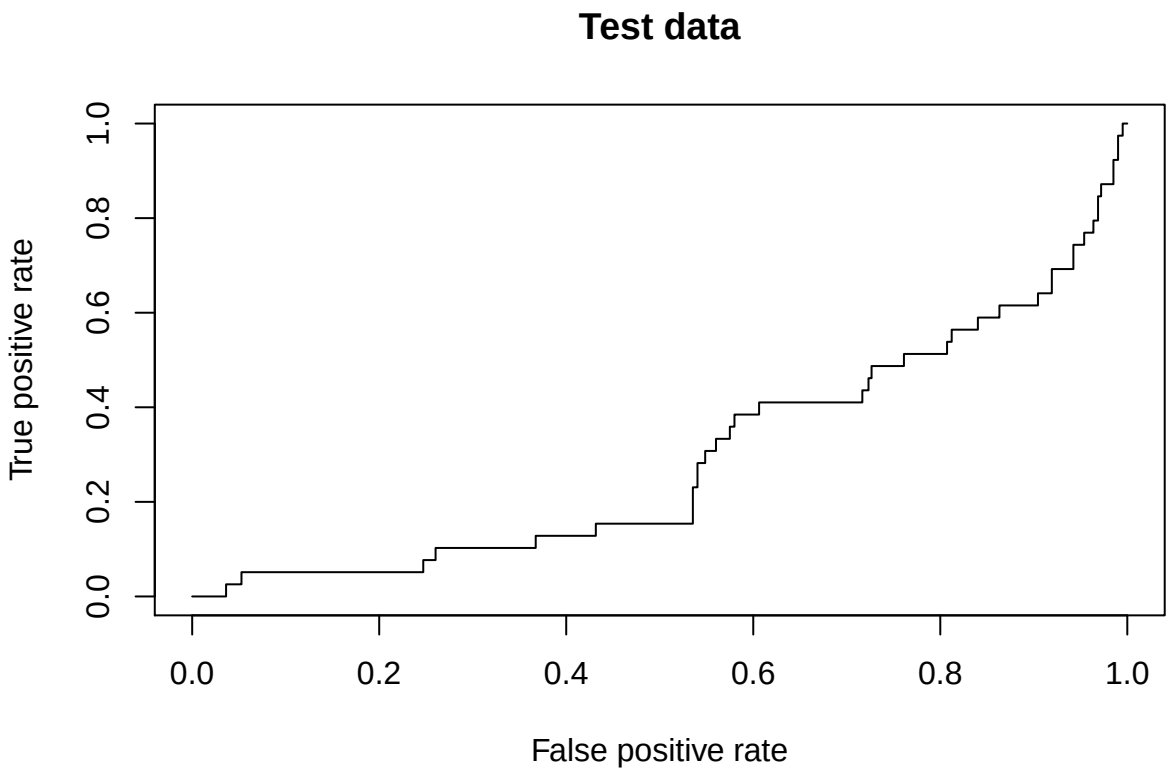
```r
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

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

```r
svmfit.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")
```




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



```
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: 0.06759521
##
## - Detailed performance results:
##    cost      error dispersion
## 1 0.001 0.06759521  0.0140538
## 2 0.010 0.06759521  0.0140538
## 3 0.100 0.06759521  0.0140538
## 4 1.000 0.06759521  0.0140538
## 5 5.000 0.06759521  0.0140538
```

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

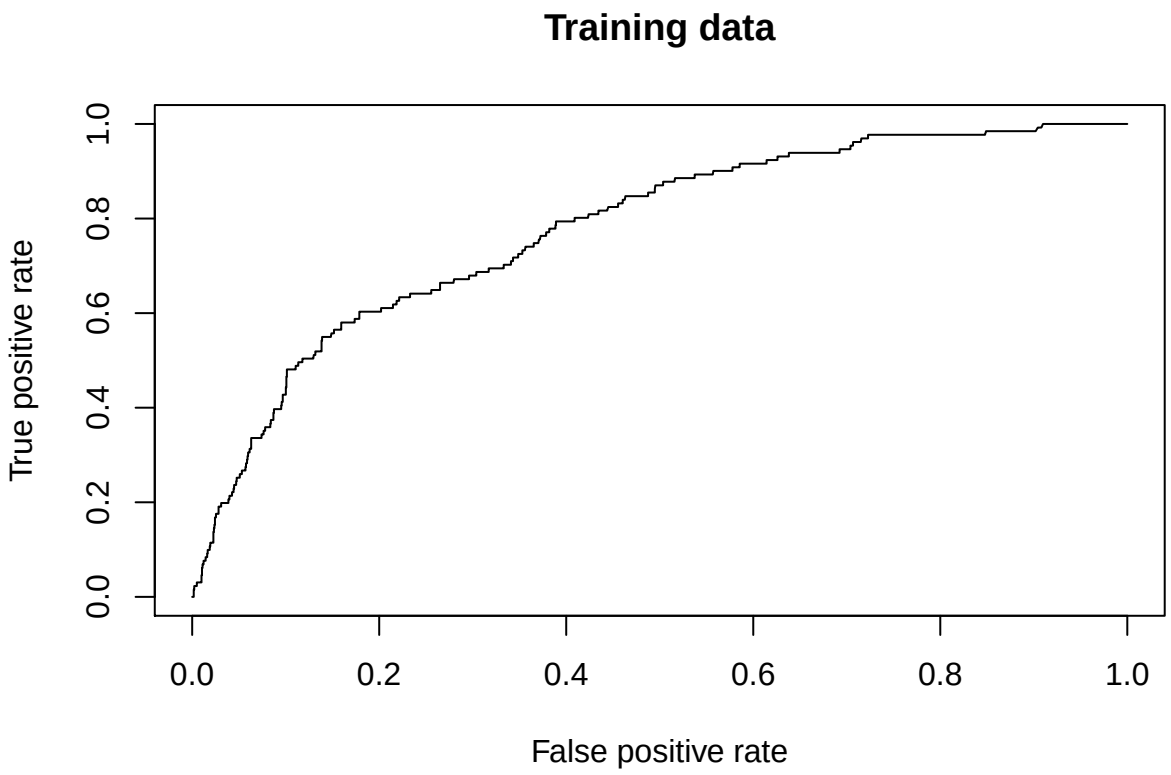
```
##
## Call:
## best.tune(method = svm, train.x = factor(class) ~ seismic + shift +
##     gpuls + nbumps, data = seismic[train, ], ranges = list(cost = c(0.001,
##     0.01, 0.1, 1, 5)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost: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

```
##
##
## Number of Classes:  2
##
## Levels:
##  0 1
```

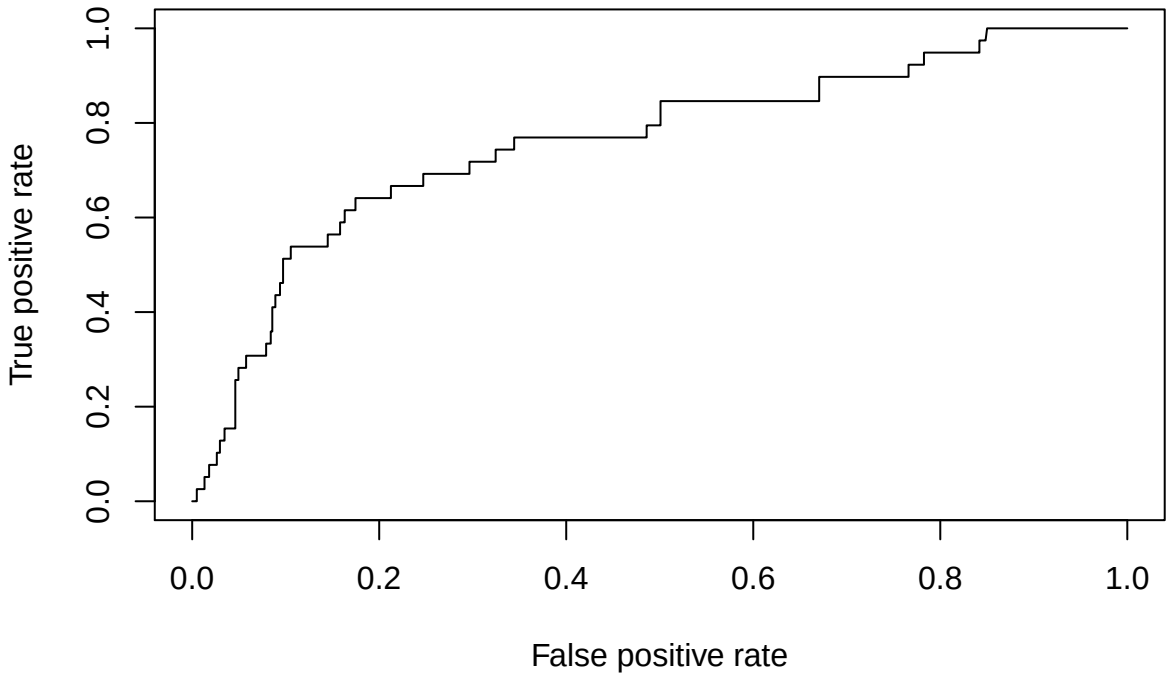
```
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

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

```
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")
```



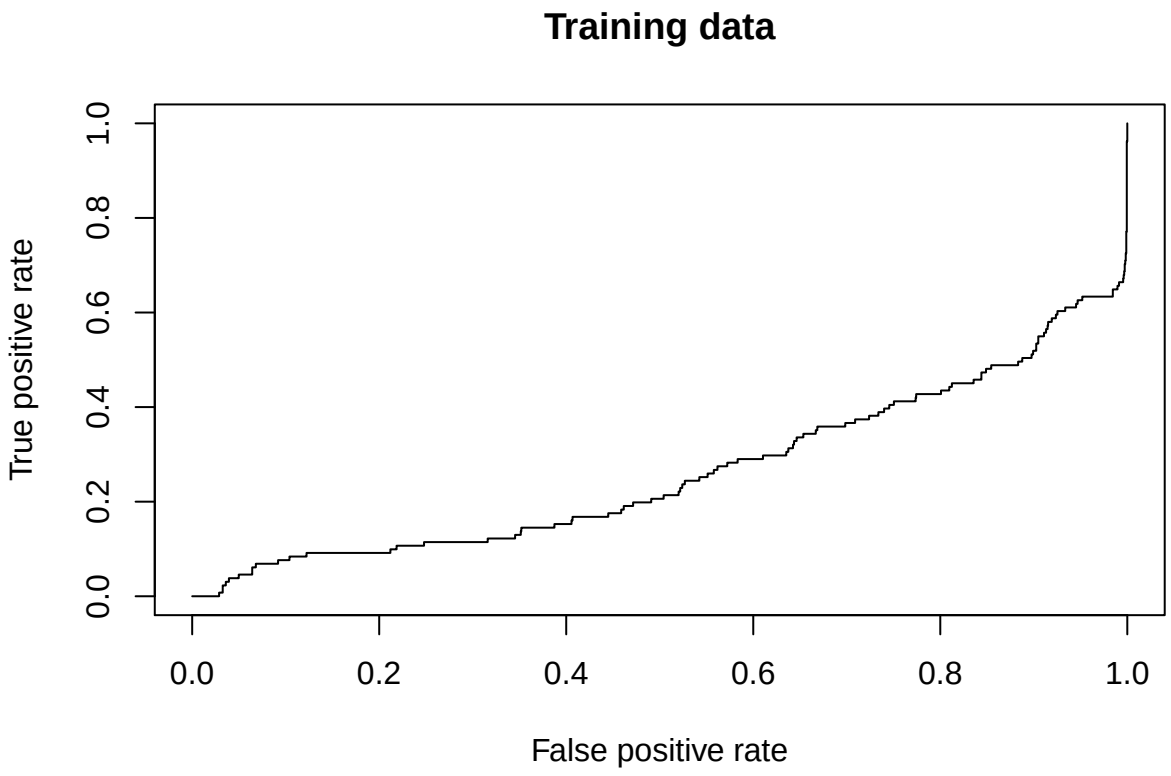

```
rocplot(fitted.test2, seismic[-train,"class"], main = "Test data")
```



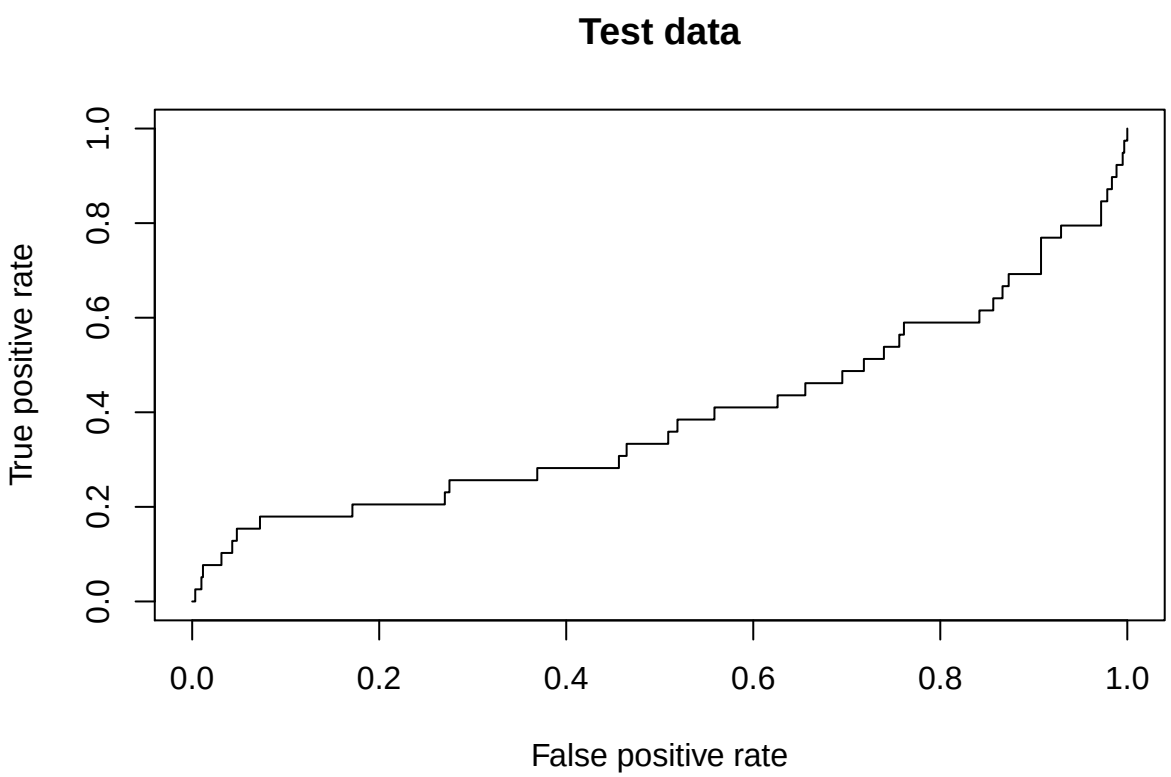

```
total.time <- proc.time() - start.time
time2 <- total.time[3]

#-------------------------------------------------
# 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
```

```
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")
```



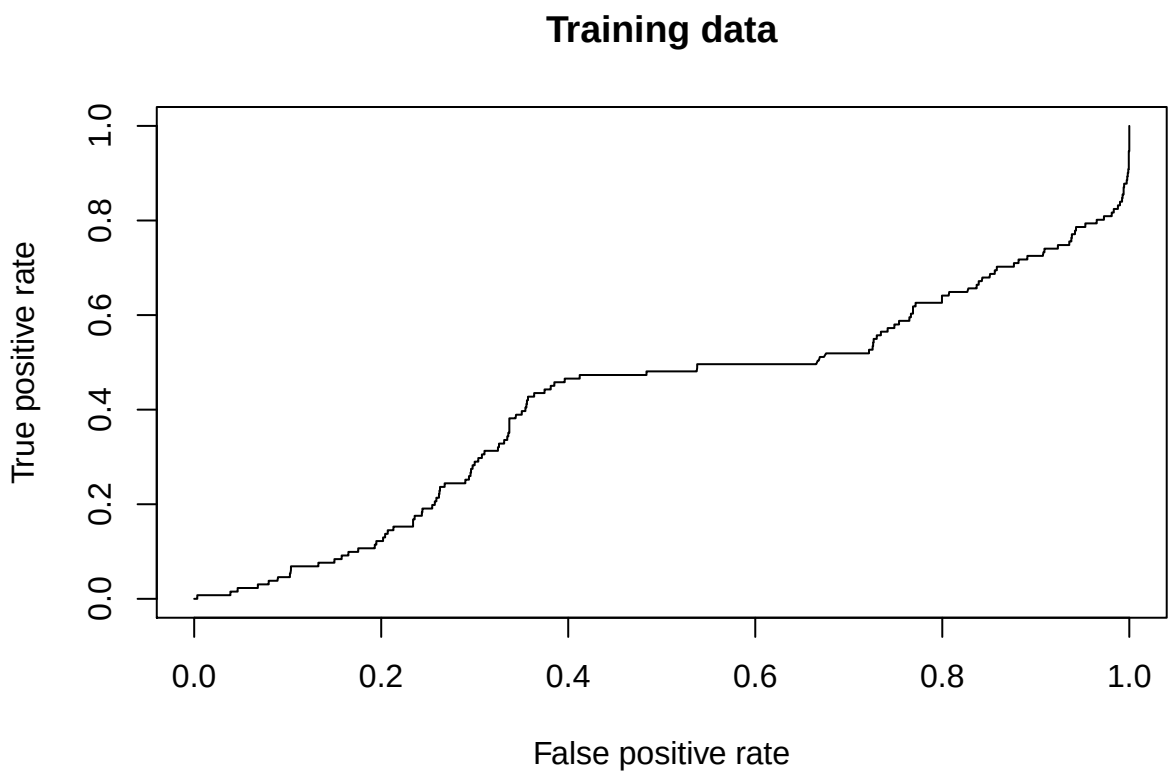

```
rocplot(fitted.test2, seismic[-train,"class"], main = "Test data")
```



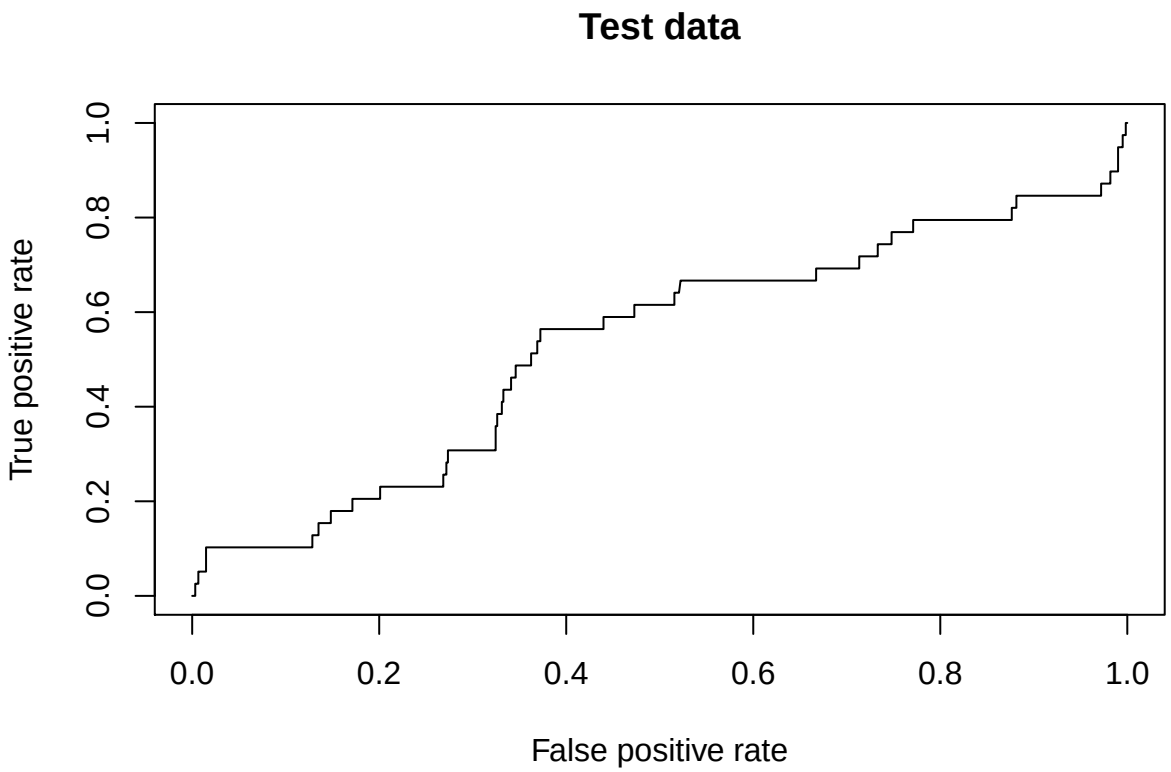
```
total.time <- proc.time() - start.time
time3 <- total.time[3]

##
## 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")
```



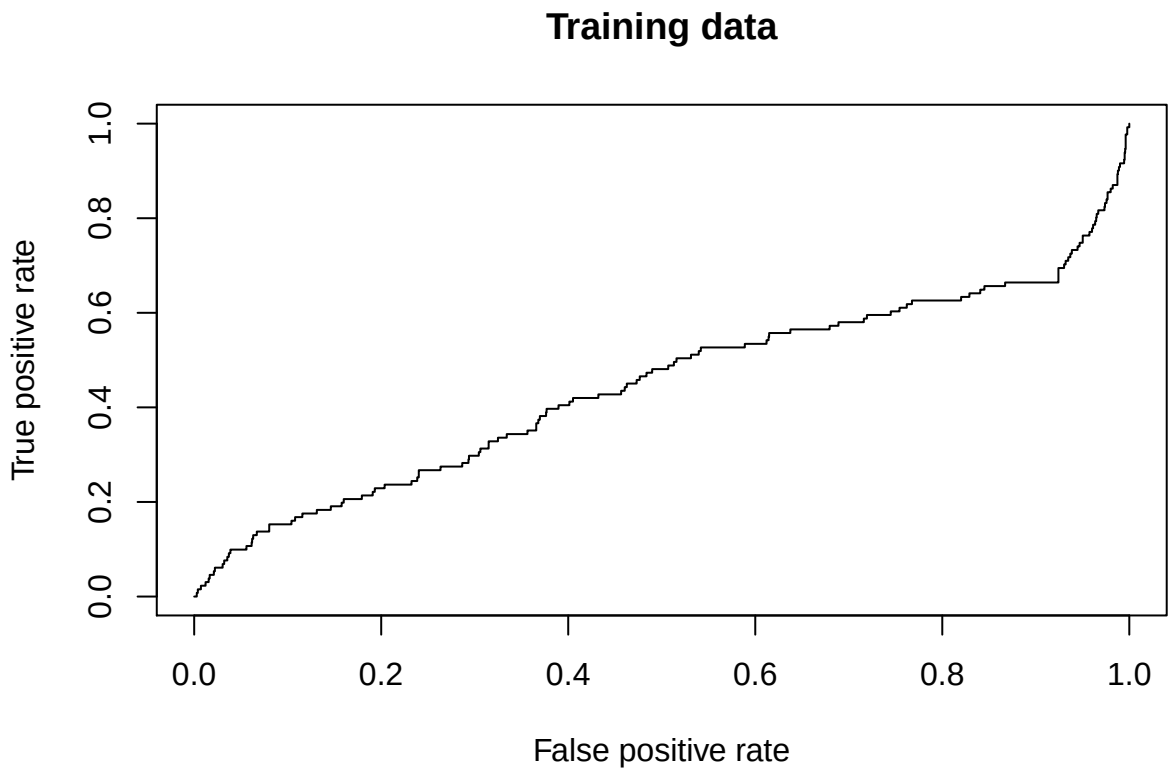
```
rocplot(fitted.test3, seismic[-train,"class"], main = "Test data")
```



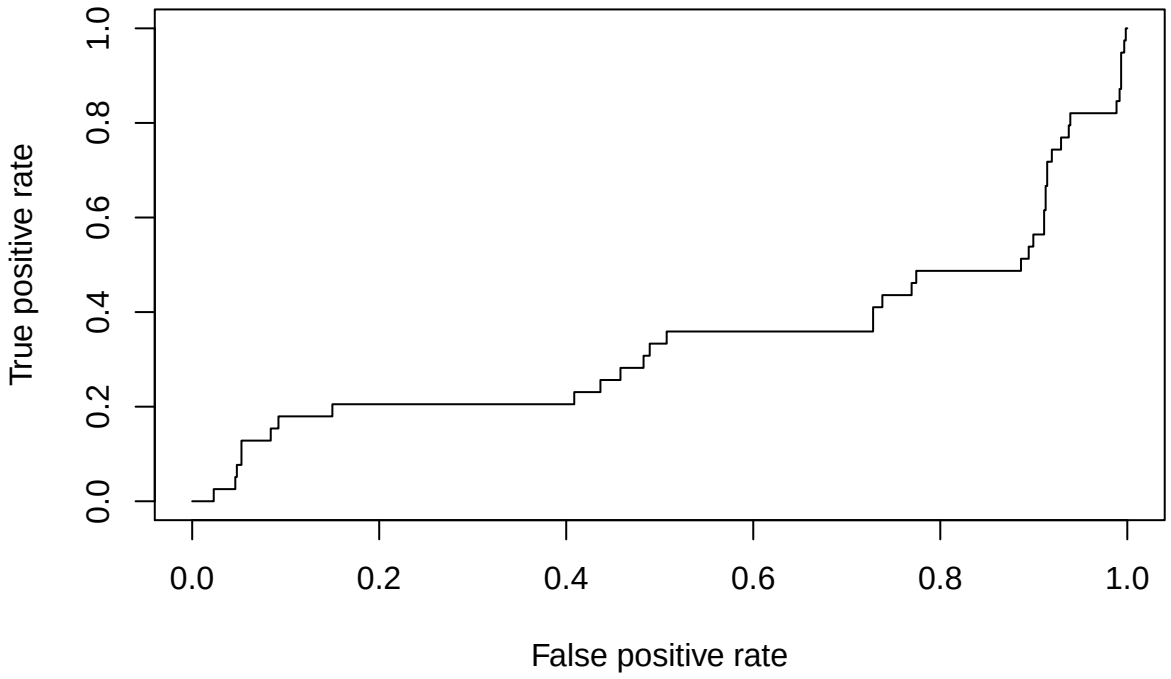

```
total.time <- proc.time() - start.time
time4 <- total.time[3]

#-------------------------------------------------
# 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,], ker
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")
```



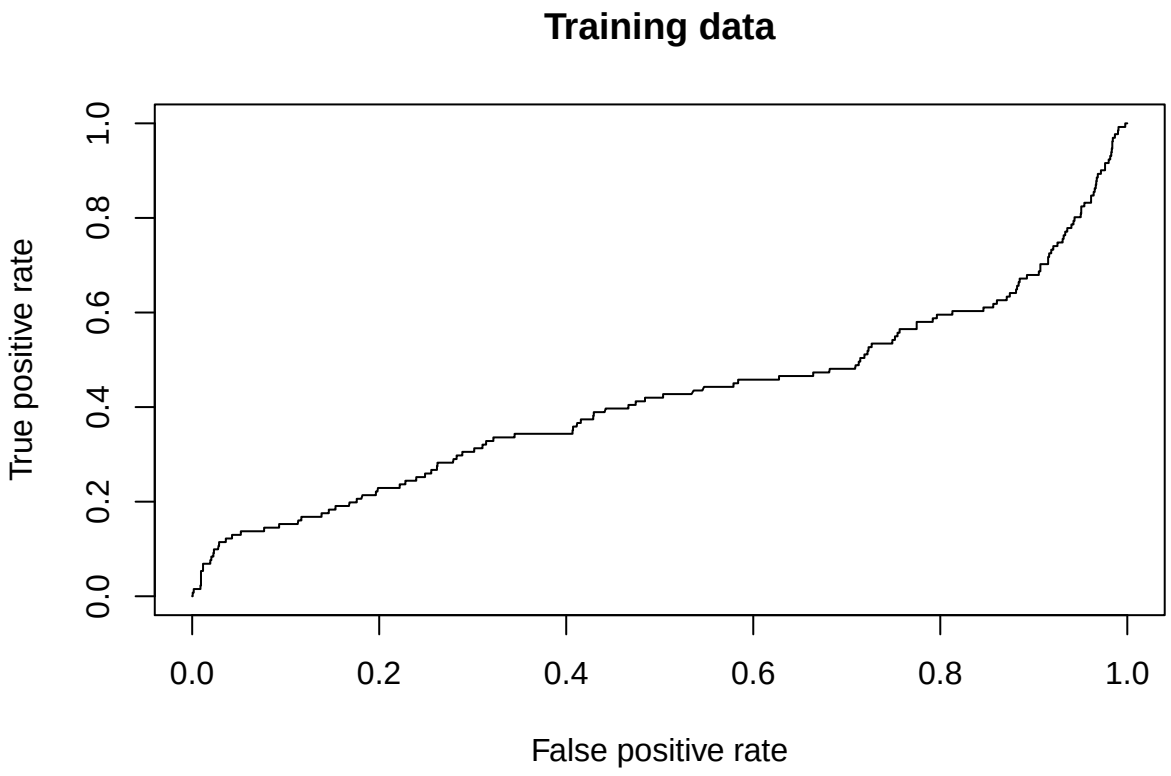

```
rocplot(fitted.test4, seismic[-train,"class"], main = "Test data")
```



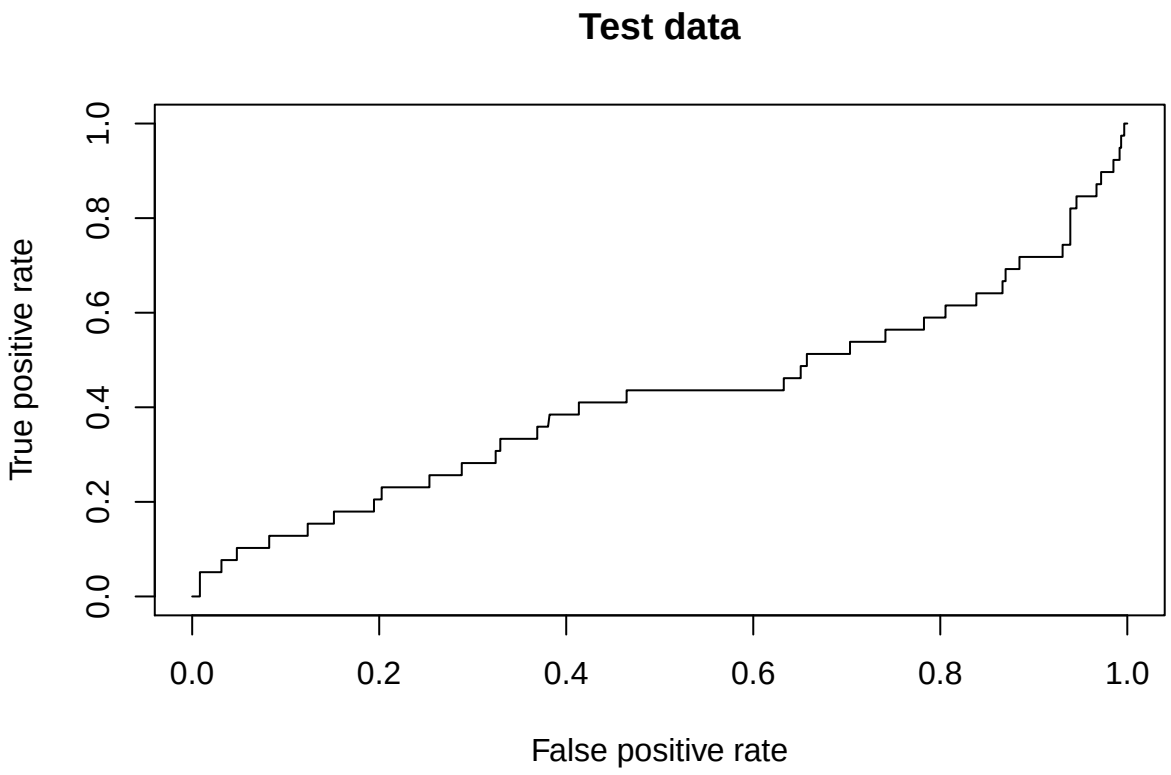

```
total.time <- proc.time() - start.time
time5 <- total.time[3]

##
## 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 = "polyn
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")
```




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



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

How to time your code!!!

```
#-------------------------------------
# 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
```