

Data Mining Project 3

Ben Straub

April 18th, 2017

Introduction

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. $> 10^4\text{J}$) 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 ($>10^4\text{J}$) 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 $> 10^4\text{J}$ was listed. Table 1 in the Appendix has a listing of all 18 variables and their descriptions.

The purpose of this project is to find whether and how the other 18 variables can be used to determine the hazard status of the mine. In project 2, we utilized techniques such as the indicator matrix linear regression, logistic regression, linear discriminant analysis(LDA), quadratic discriminant analysis (QDA), and regularized discriminant analysis (RDA) to try and find a model that would accurately predict the hazardous state. Unfortunately, all of the five project two methods performed poorly. We felt that two major issues were at hand for this poor performance of the five methods. First, the low incidences of “1’s” in the response variable class, which indicates a hazardous state in the mine. Only 170 “1’s” for class out of 2584 were observed. A difficult problem for our analyses. The second issue was multicollinearity. 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 ($\text{VIF} > 10$) between some of the covariates. Table 2 in the Appendix contains VIF’s for the linear regression model.

Multicollinearity can be address by dimension reduction techniques such as PCA, step-wise regression, LASSO and ridge. In project 2, we utilized step-wise regression and LASSO to arrive at two candidate models.

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 ($\text{VIF} > 10$) between some of the covariates, as shown below.

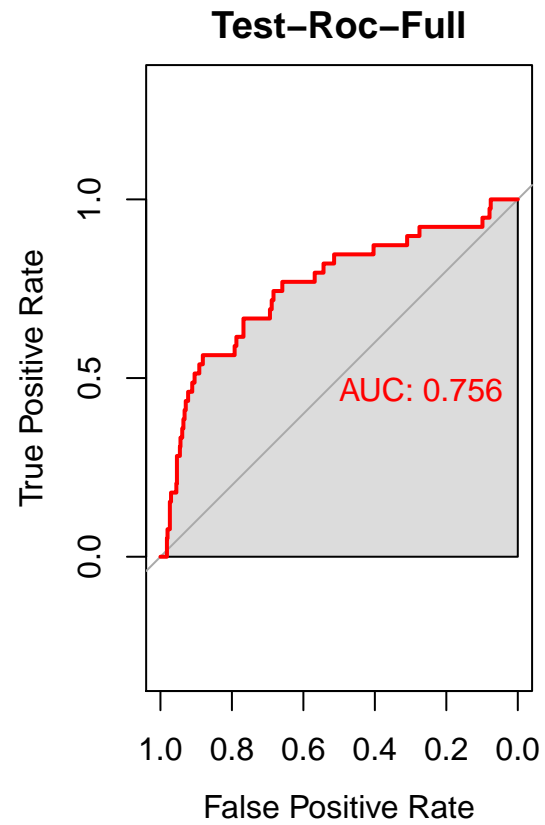
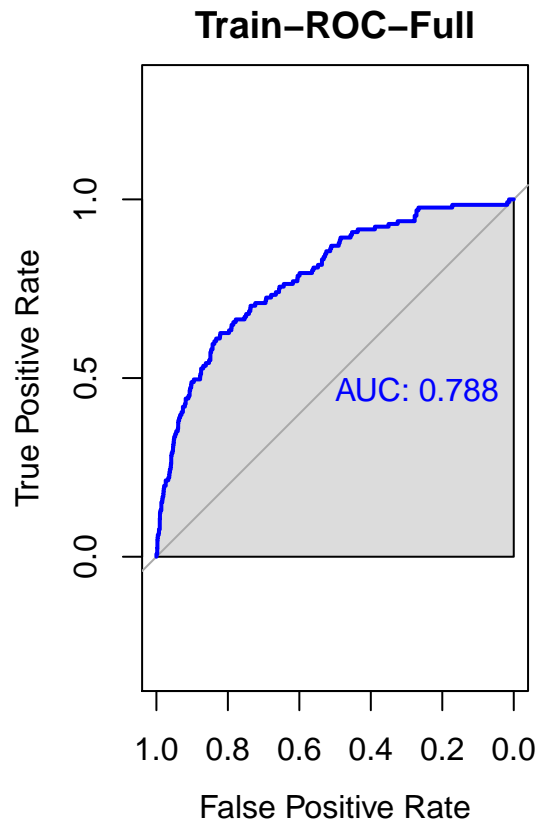
Classification before Variable Selection

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.

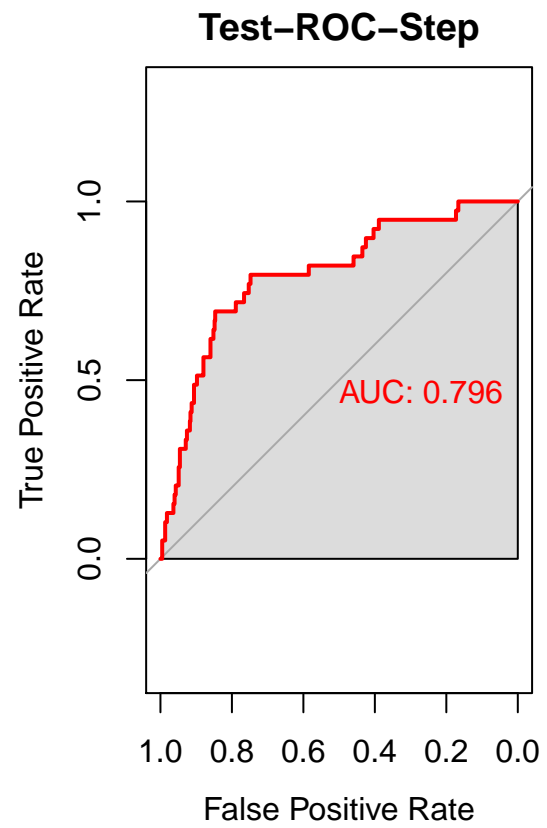
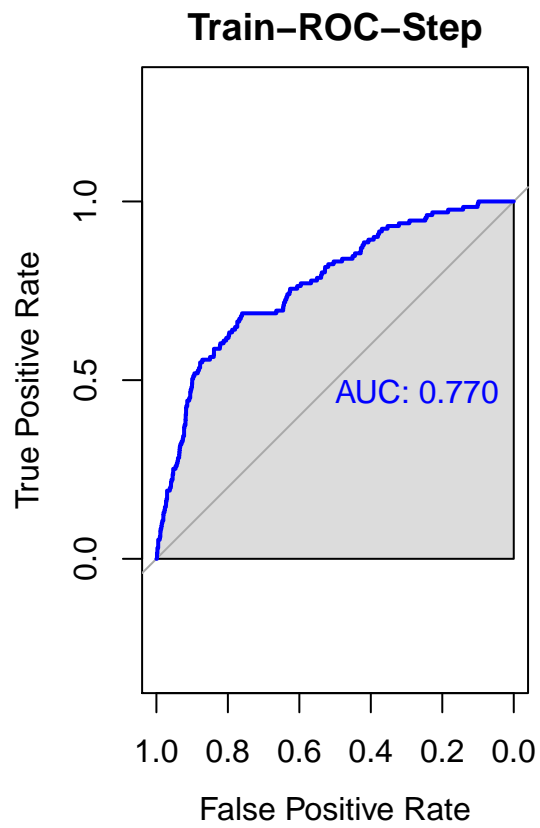
The goal is to examine which classification method outputs comparatively better prediction for seismic hazards based on available predictors.

Logistic Regression

Full Model

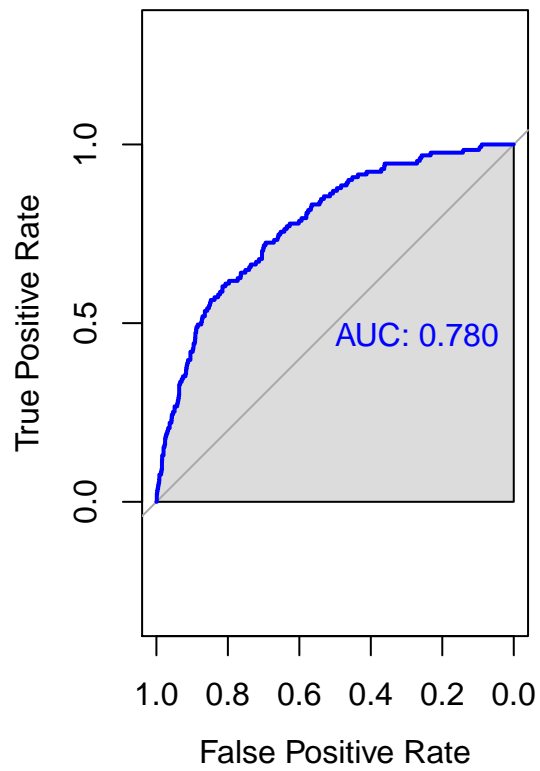


Logistic Regression - Step Model

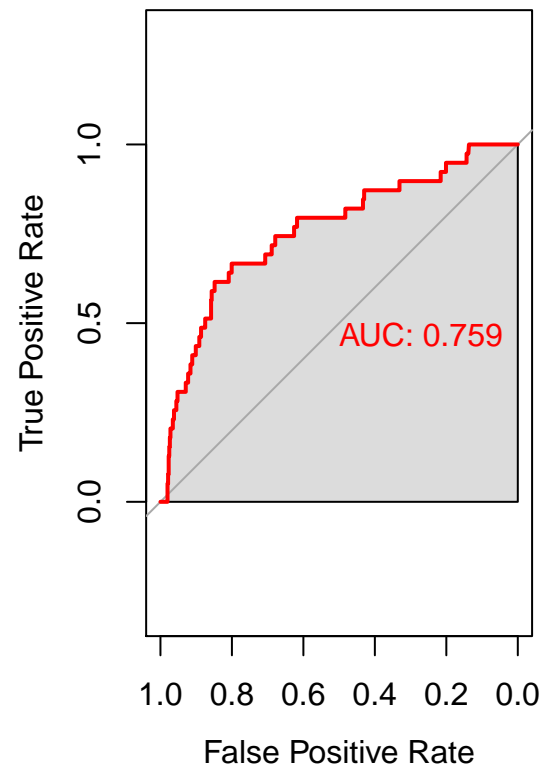


Logistic Regression - Lasso Model

Train-ROC-Lasso



Test-ROC-Lasso



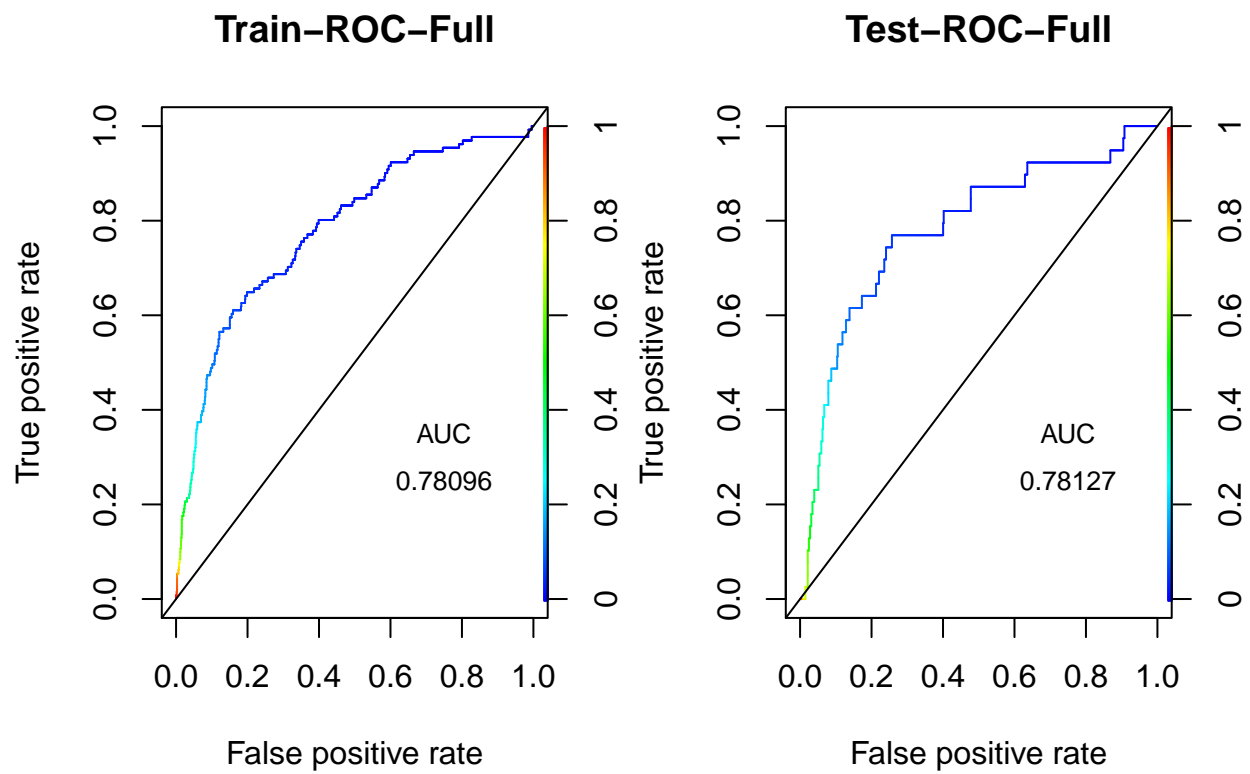
```
time1 time2 time3
elapsed 0.116 0.203 0.074
```

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

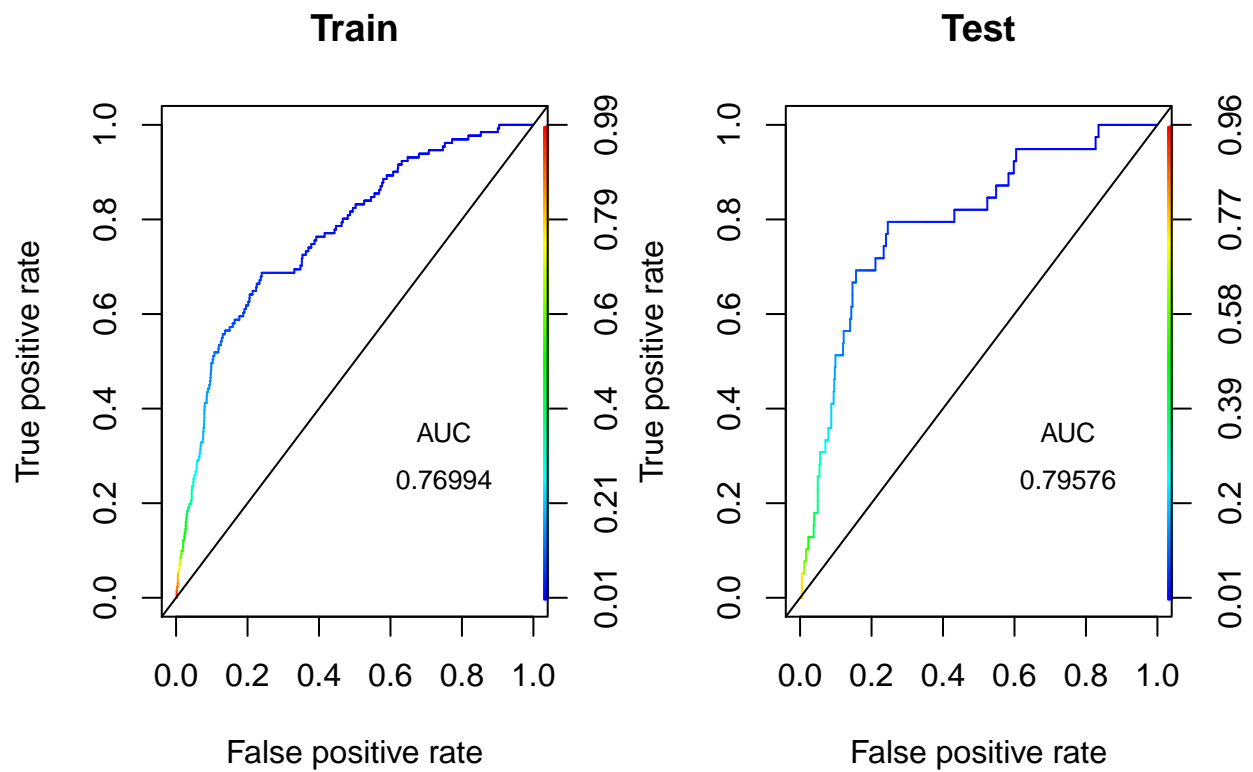
```
rate2.test rate4.test rate6.test
[1,]      0.065      0.062      0.062
```

Linear Discriminant Analysis

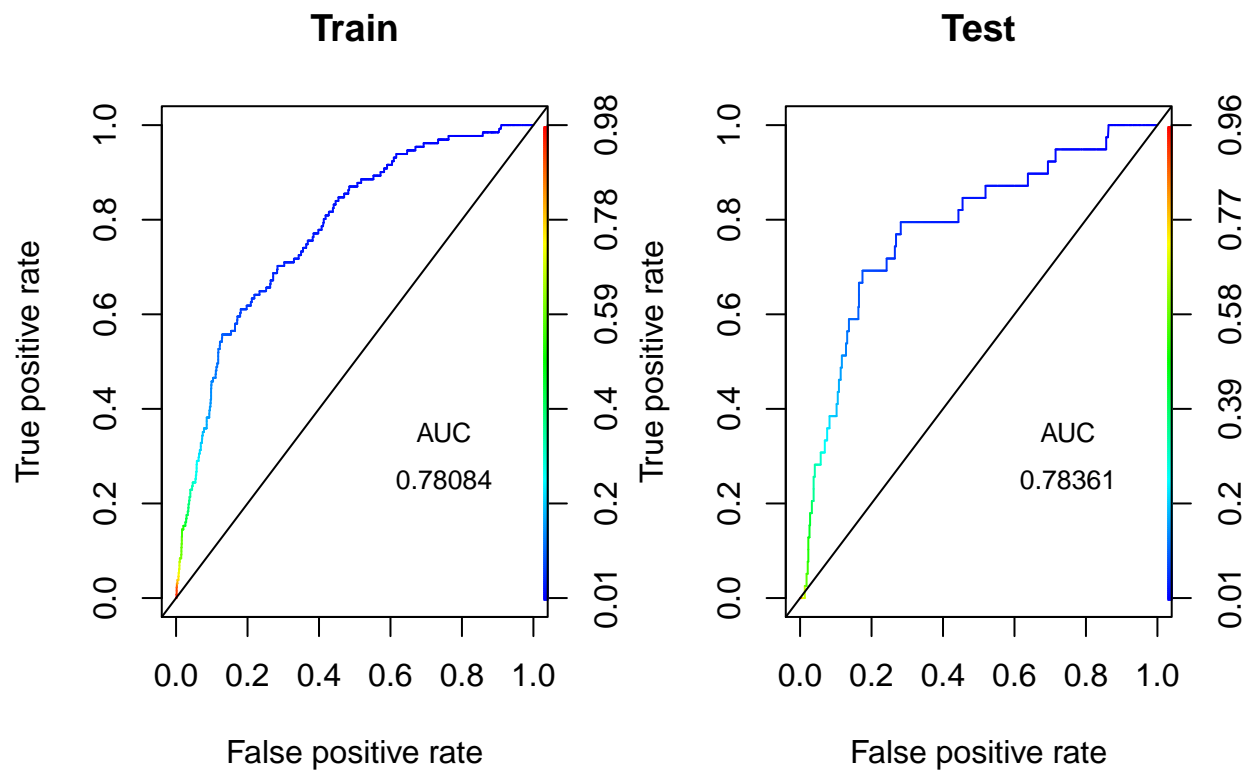
Full Model



Linear Discriminant Analysis - Step



Linear Discriminant Analysis - Lasso



```
time1 time2 time3
elapsed 0.873 0.828 0.887
```

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

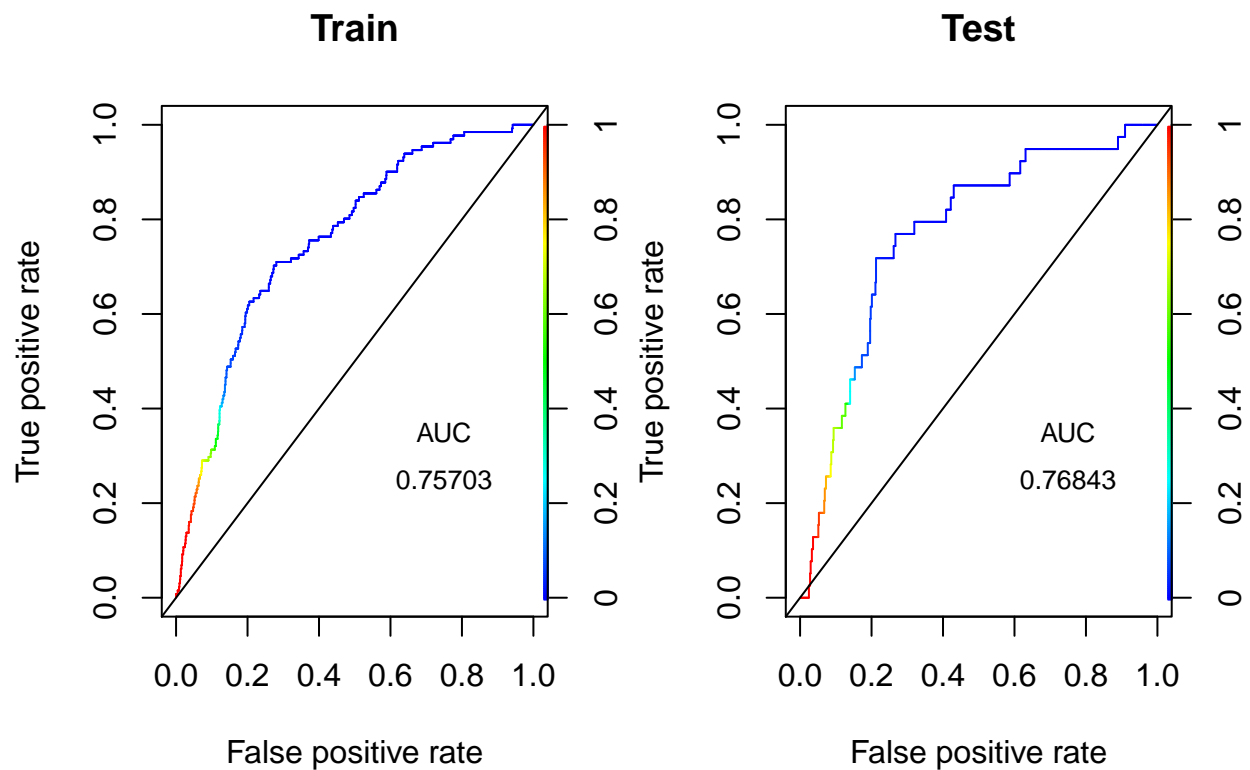
```
rate2.test rate4.test rate6.test
[1,]         0.077         0.076         0.076
```

Quadratic Discriminant Analysis

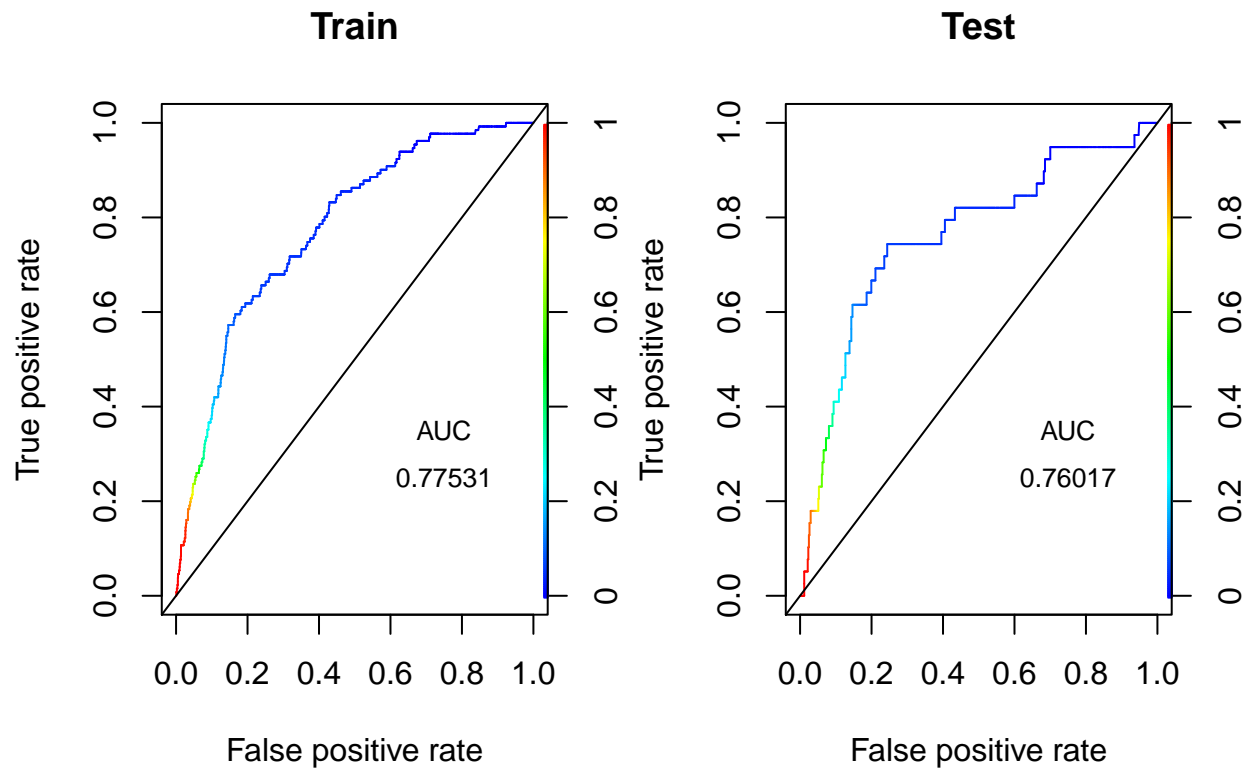
Full Model

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

Quadratic Discriminant Analysis - Step



Quadratic Discriminant Analysis - LASSO

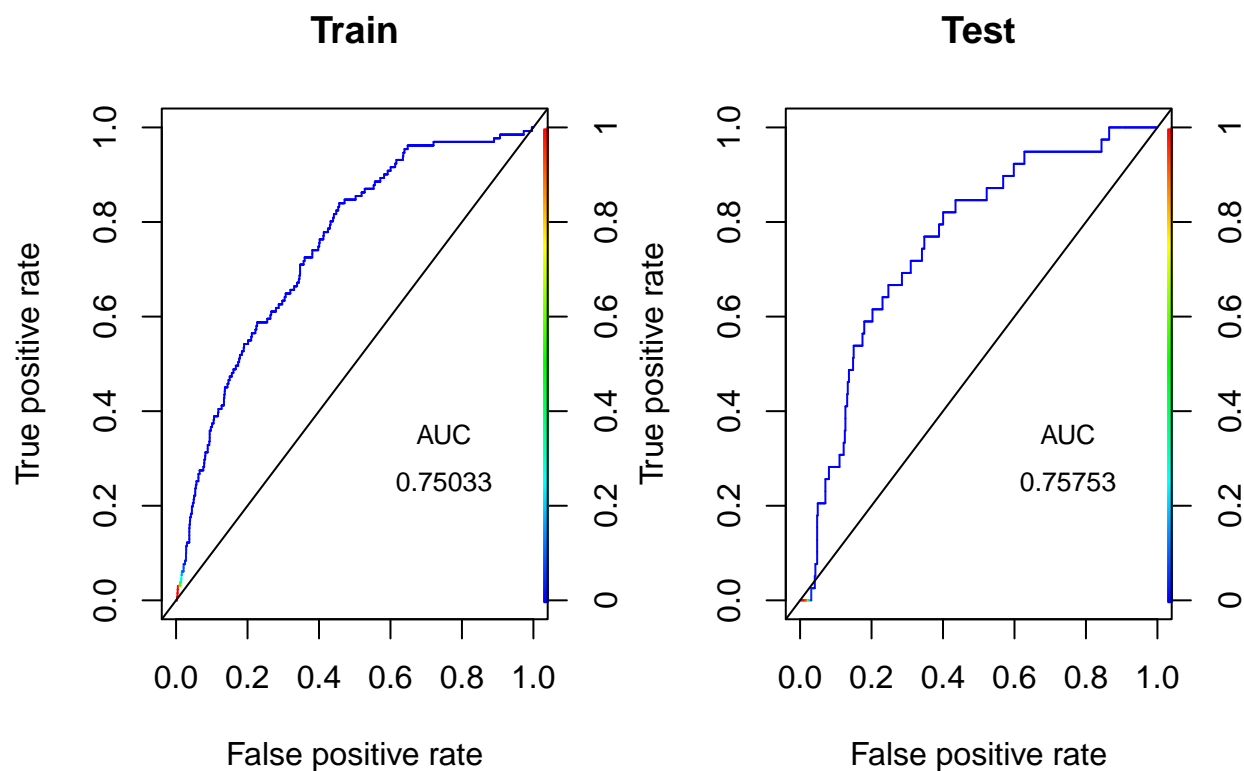


```
time1 time2  
elapsed 1.012 0.723
```

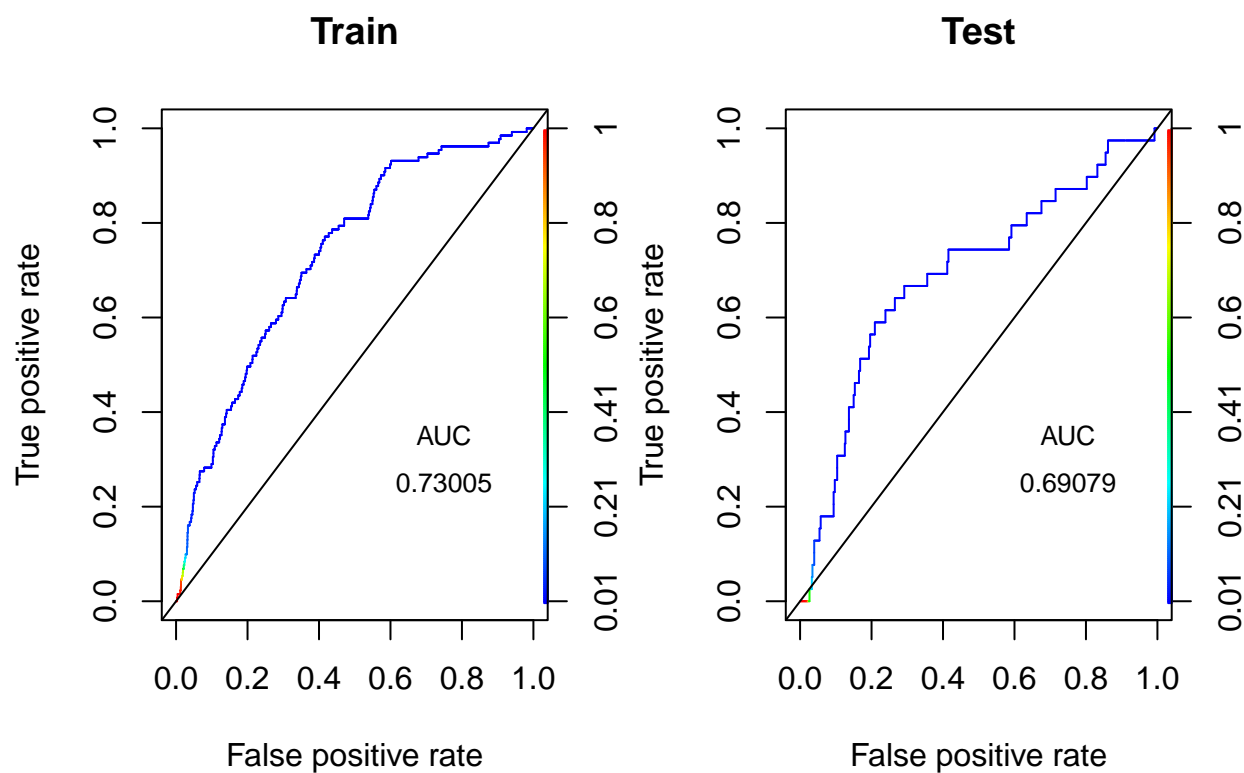
```
rate1.train rate3.train rate5.train  
[1,] 0.149 0.109 0.077
```

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

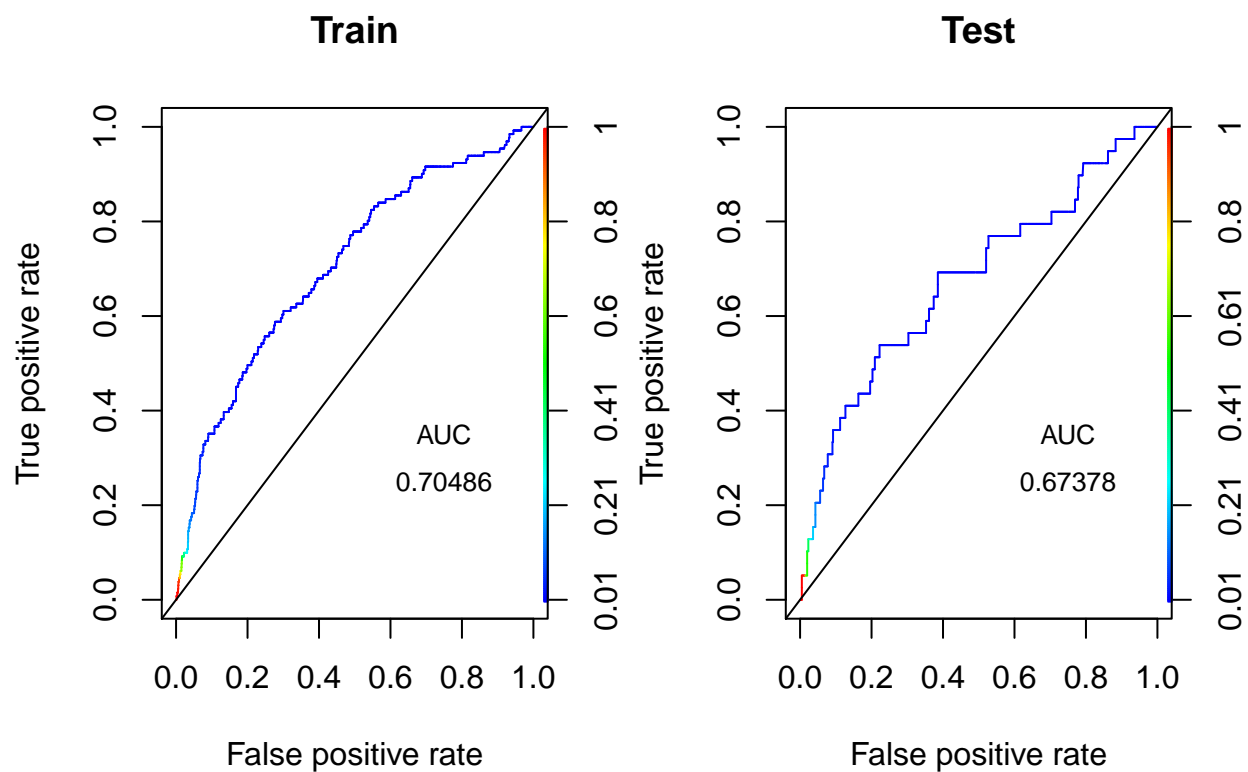

Regularized



Regularized Discriminant Analysis - Step



Regularized Discriminant Analysis - Lasso



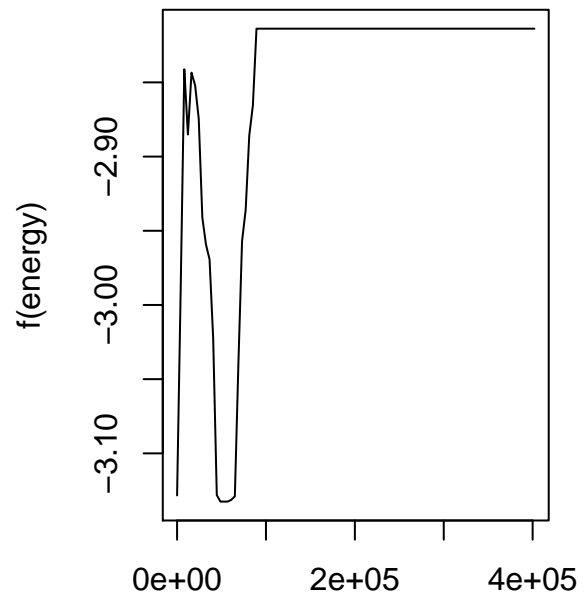
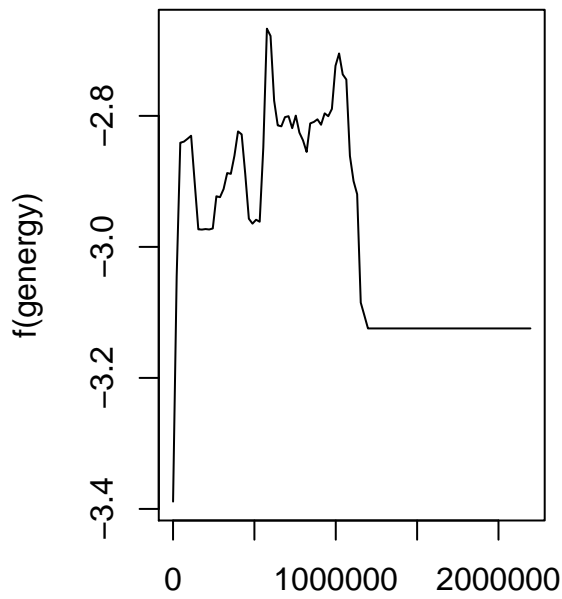
```
time1 time2
elapsed 3.653 1.605
```

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

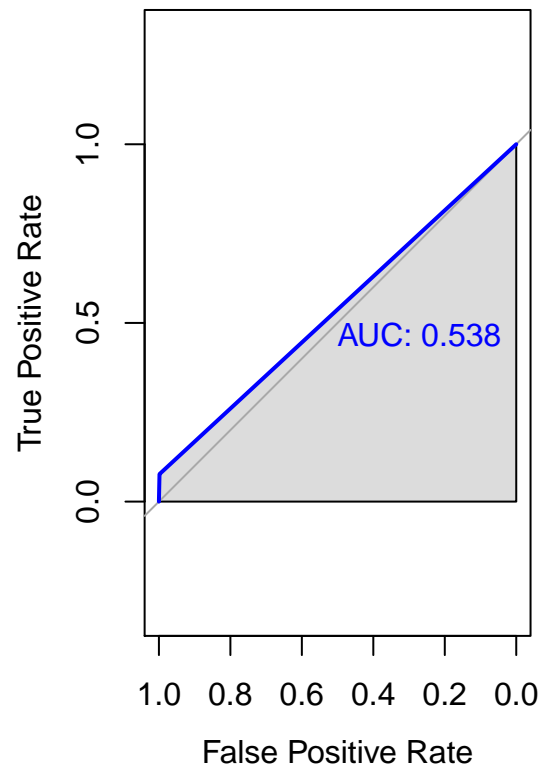
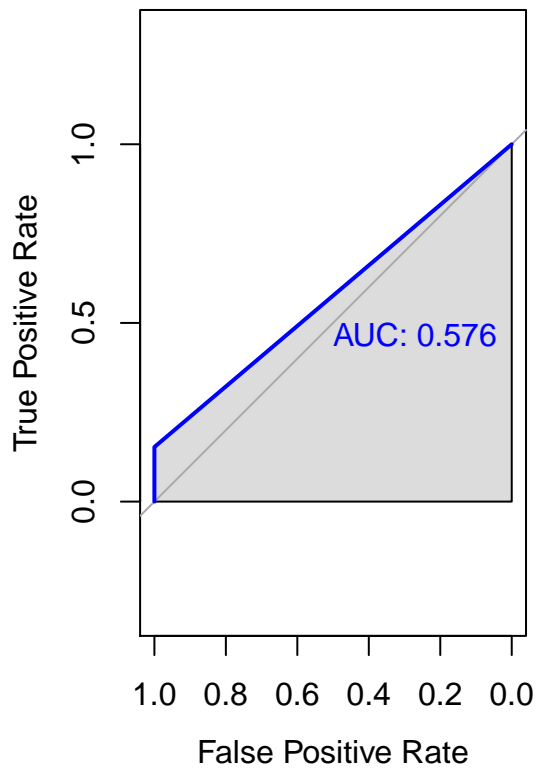
```
rate2.test rate4.test rate6.test
[1,]      0.082      0.085      0.074
```

<<<<<< HEAD # Boosting before variable selection

```
elapsed
7.836
```



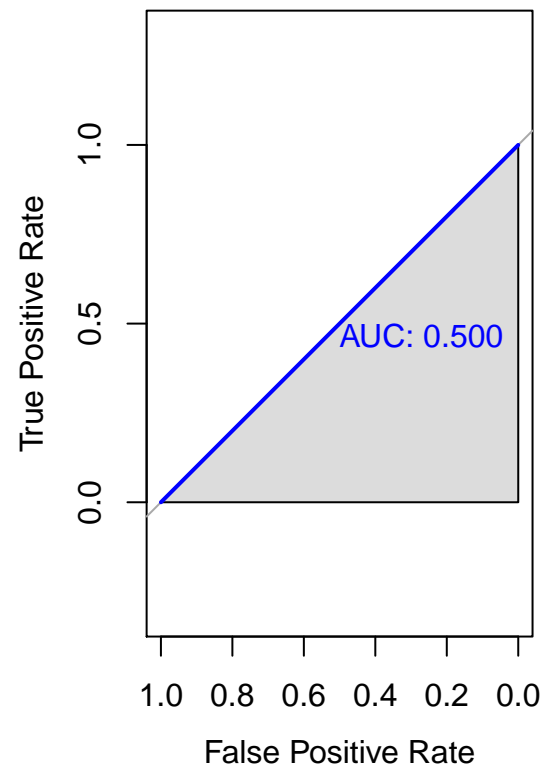
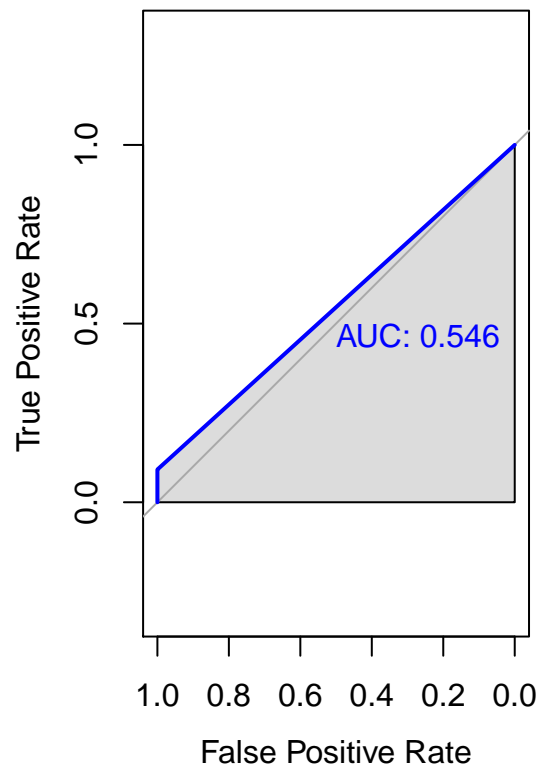
Test ROC for Boosting Classification



Boosting after variable selection

elapsed
3.467

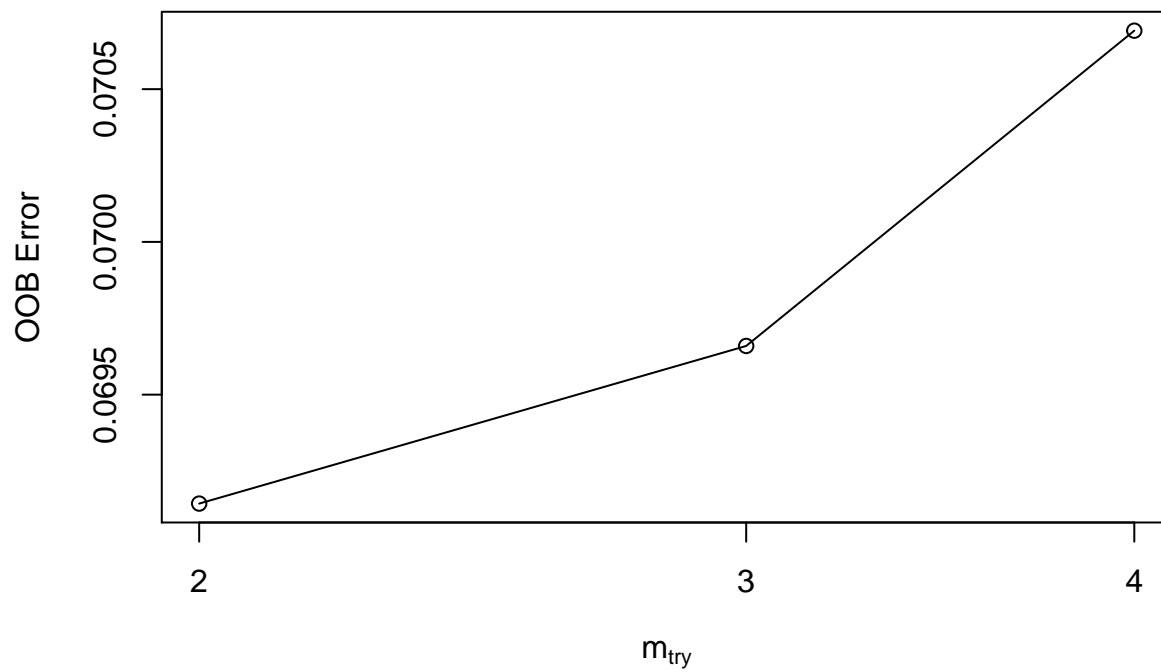
Test ROC for Boosting Classification



Random Forests Classification

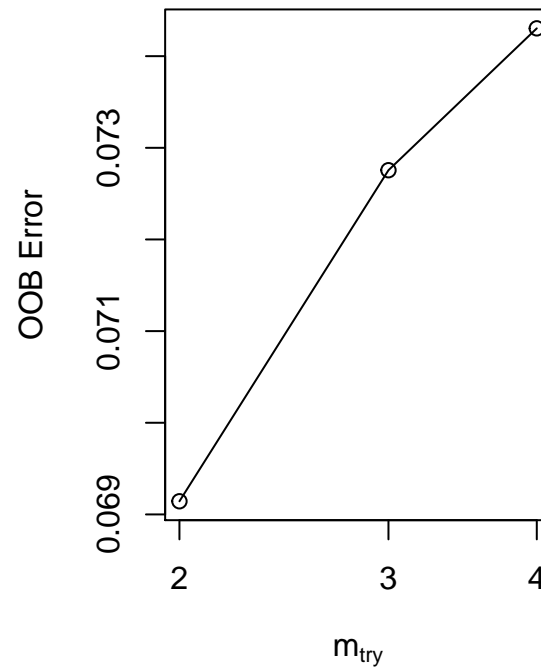
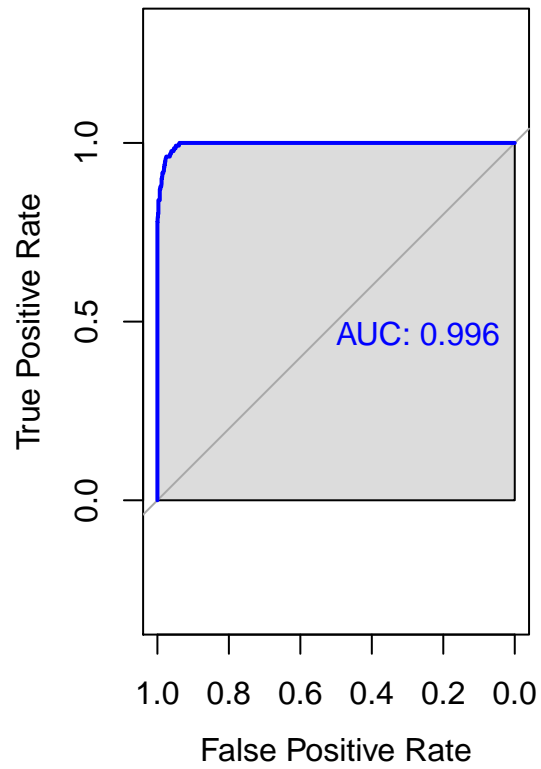
RF Classification BEFORE Variable Selection

```
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
```



```
mtry = 3  OOB error = 7.28%  
Searching left ...  
mtry = 2    OOB error = 6.91%  
0.04964539 0.01  
Searching right ...  
mtry = 4    OOB error = 7.43%  
-0.07462687 0.01
```

Train ROC for RF Classification

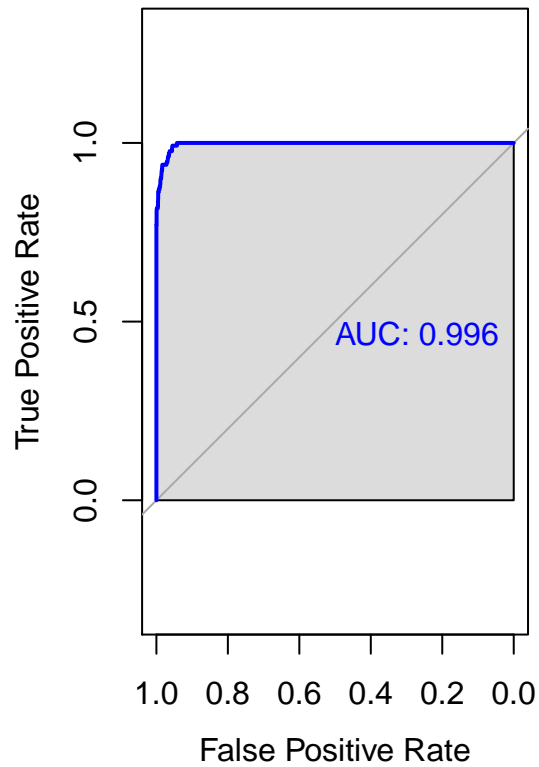


[1] 0

[1] 0.2363033

[1] 0.2203302

Train ROC for RF Classification



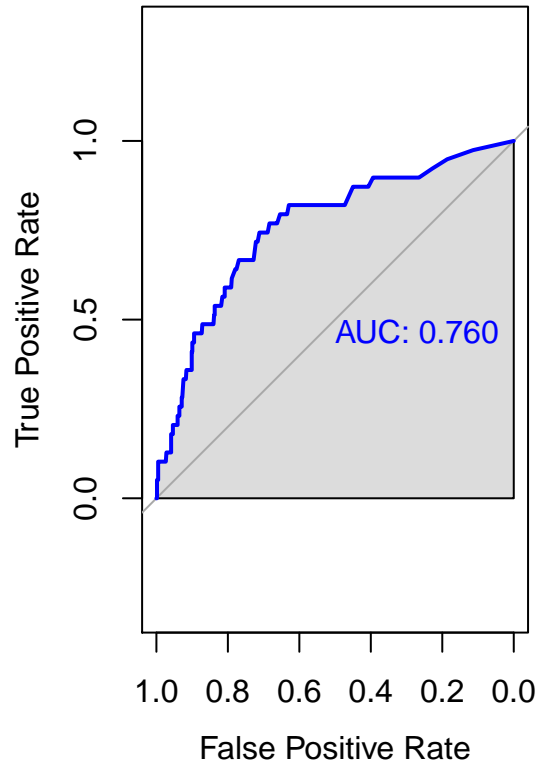
[1] 0.02564103

[1] 0.1136738

[1] 0.1083591

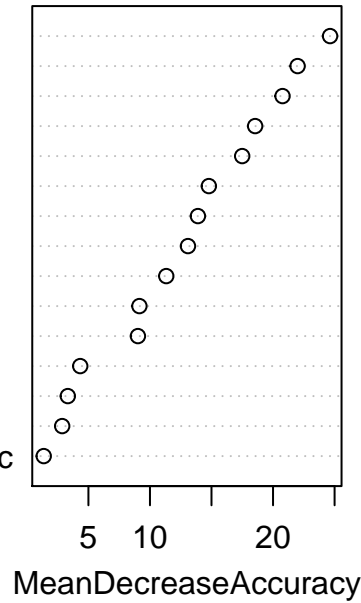
	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
seismic	7.639190	5.1409941	9.142981	4.2665367
seismoacoustic	1.581907	-0.2409790	1.368947	4.5015275
shift	2.486616	0.7266865	2.869017	2.4176743
genergy	12.086539	2.4284640	13.895572	25.1355203
gpuls	18.476810	13.6828191	21.994591	26.7512211
gdenergy	22.120246	-8.0739569	20.771536	20.7055737
gdpuls	25.688347	-7.5341551	24.634248	20.8810289
ghazard	4.587309	-2.7149240	3.327211	1.9414849
nbumps	13.977076	5.3373298	14.784089	11.5360046
nbumps2	6.668420	8.5245738	9.021708	8.5027181
nbumps3	9.531100	5.9025441	11.324696	7.3784317
nbumps4	14.878958	-10.0707066	13.088679	2.7869821
nbumps5	4.832149	-2.6126517	4.336337	0.3214691
energy	17.725076	-1.2777291	18.544822	18.4047305
maxenergy	17.086692	-5.1894493	17.493649	13.4764157

Test ROC for RF Classification



rf.seismic

gdpuls
gpuls
gdenergy
energy
maxenergy
nbumps
genergy
nbumps4
nbumps3
seismic
nbumps2
nbumps5
ghazard
shift
seismoacoustic

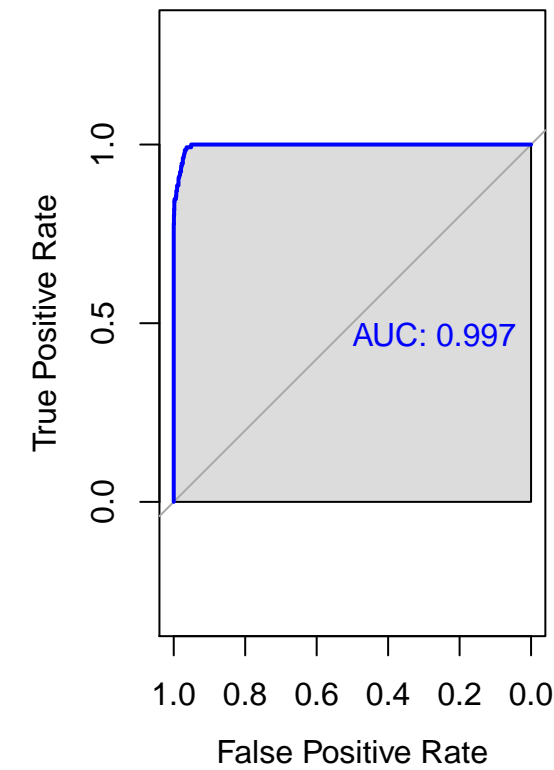


gpuls
genergy
gdpuls
gdenergy
energy
maxenergy
nbumps
nbumps2
nbumps3
seismoacoustic
seismic
nbumps4
shift
ghazard
nbumps5

RF Classification AFTER Variable Selection

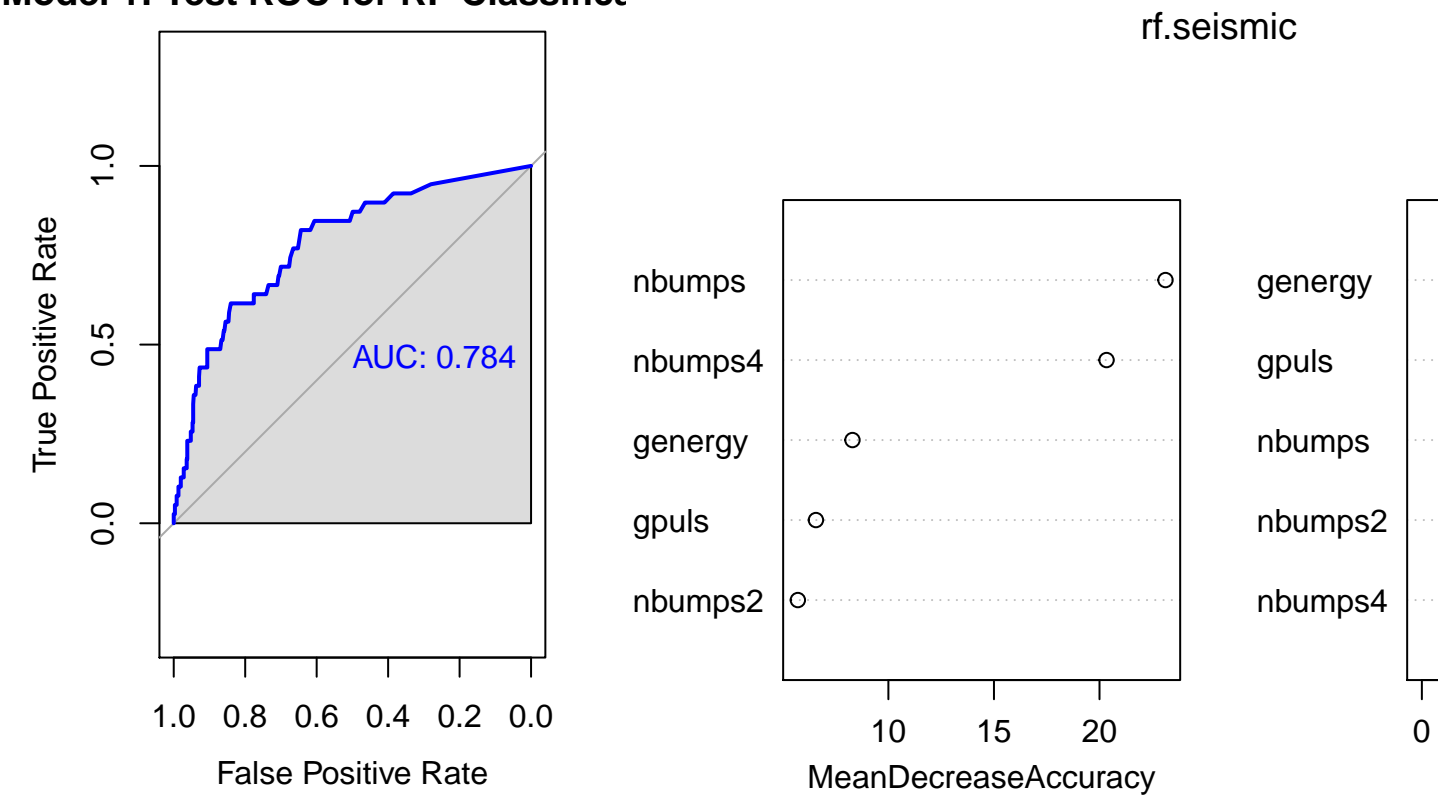
RF Classification AFTER Variable Selection

Model 1: Train ROC for RF Classific:

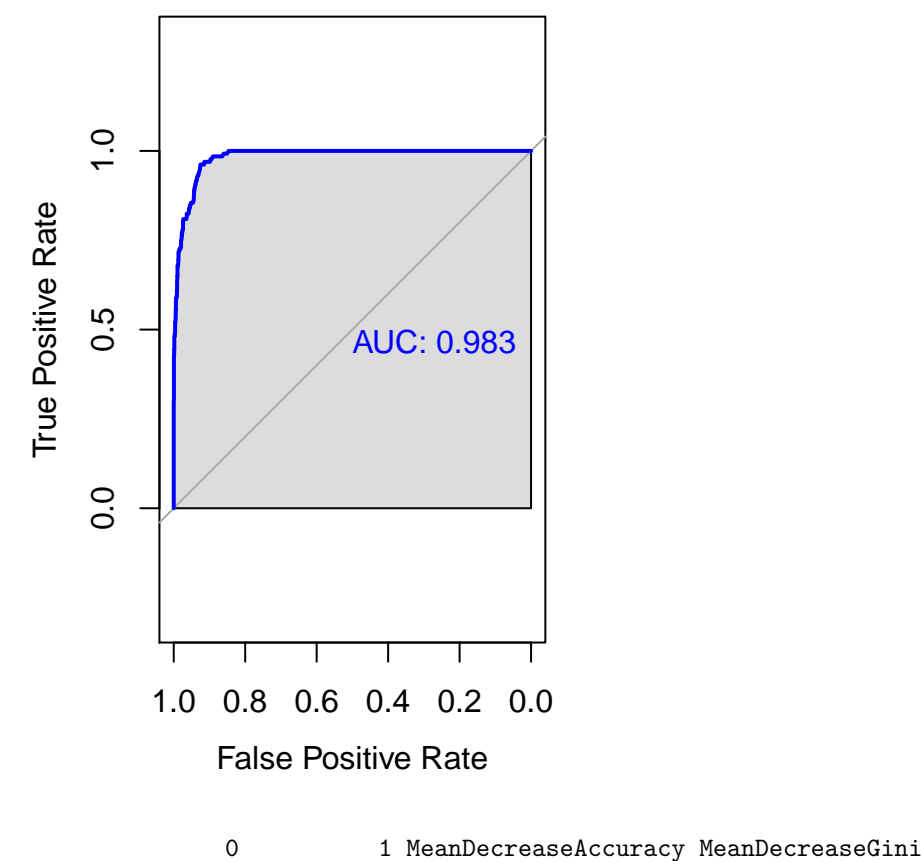


	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
genergy	6.564781	3.001198	8.298559	68.418908
gpuls	1.086896	18.312691	6.567174	66.430515
nbumps	15.305794	31.873589	23.124064	26.073064
nbumps2	2.285010	8.816717	5.711204	14.121004
nbumps4	23.265875	-8.649932	20.332543	7.037393

Model 1: Test ROC for RF Classification

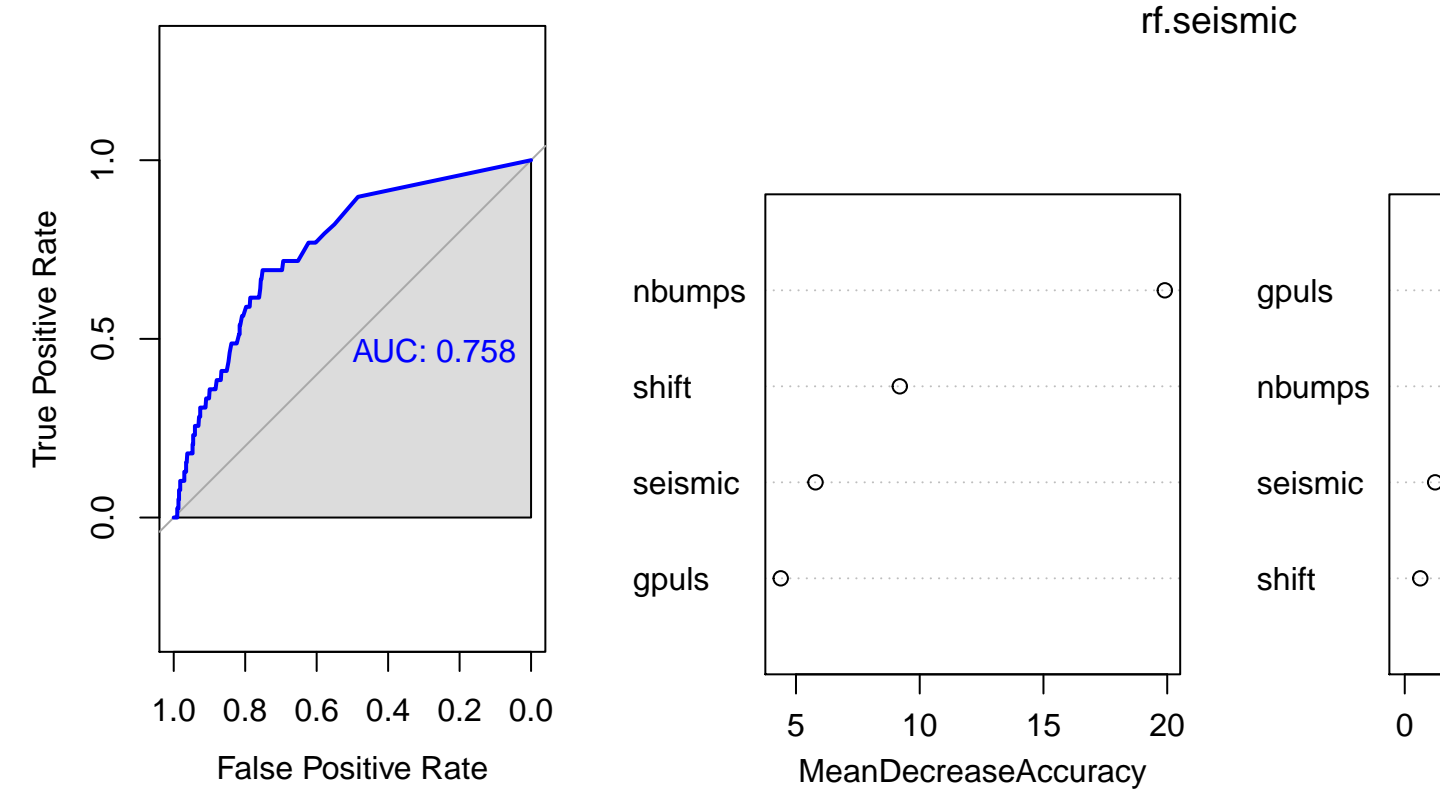


Model 2: Train ROC for RF Classification



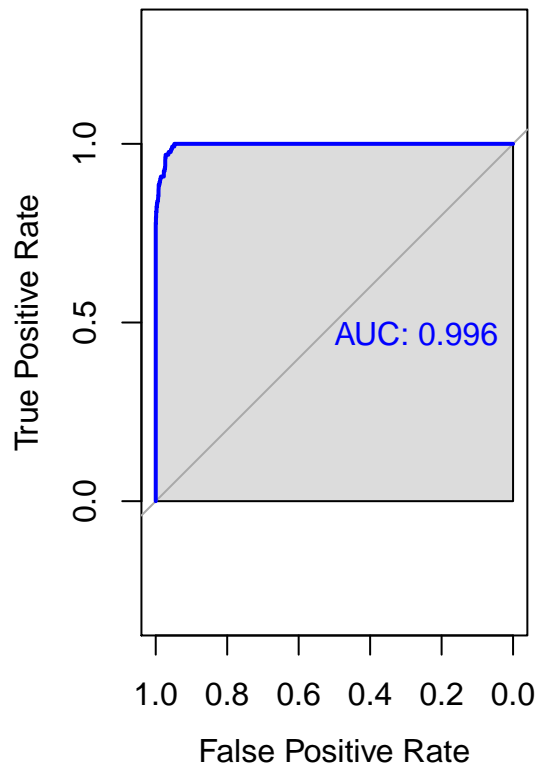
seismic	2.624006	11.019565	5.789011	5.976086
shift	11.051096	-11.510160	9.194394	3.050479
gpuls	1.478515	7.540176	4.383246	75.482383
nbumps	13.186473	22.919363	19.903187	27.608000

Model 2: Test ROC for RF Classification



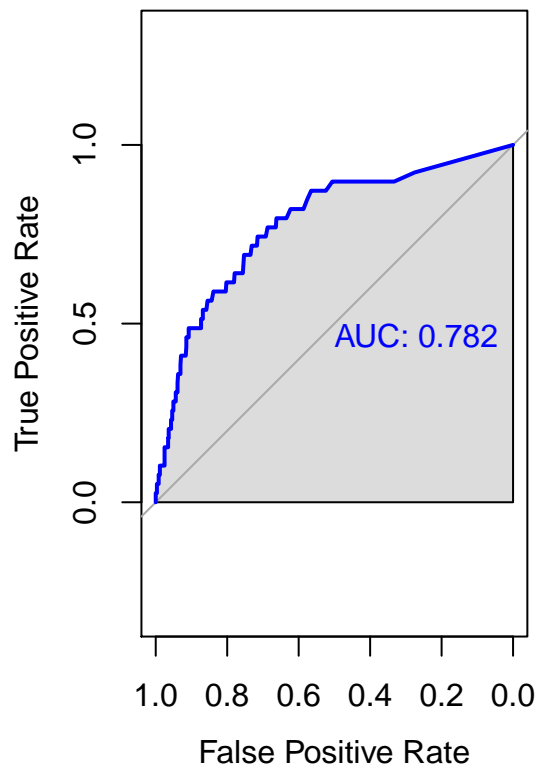
Boosting

Model 1: Train ROC for RF Classific:

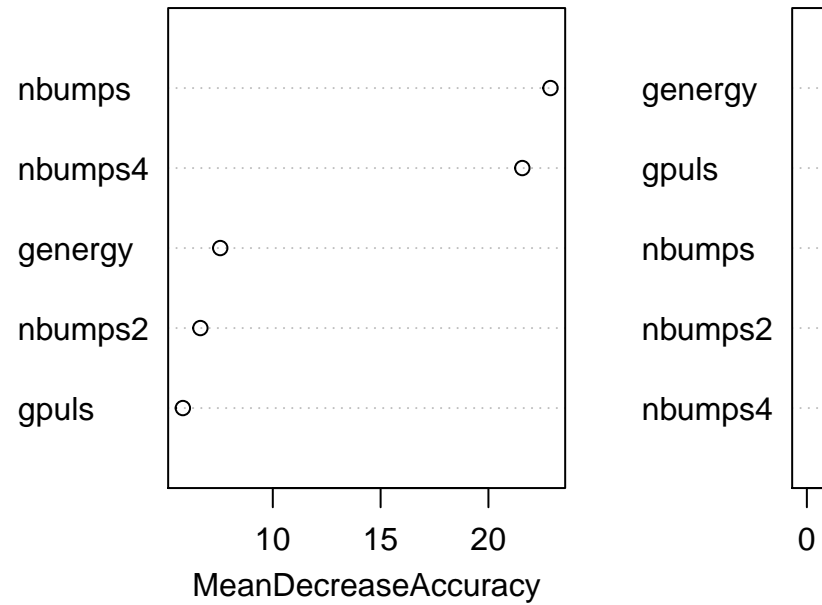


	0	1	MeanDecreaseAccuracy	MeanDecreaseGini
genergy	5.1300375	5.586820	7.563177	68.556399
gpuls	0.0700258	19.539698	5.829128	66.551265
nbumps	14.9000236	32.868947	22.877694	26.059296
nbumps2	3.0991937	9.011757	6.641027	14.385561
nbumps4	24.1165725	-8.627134	21.572093	7.176123

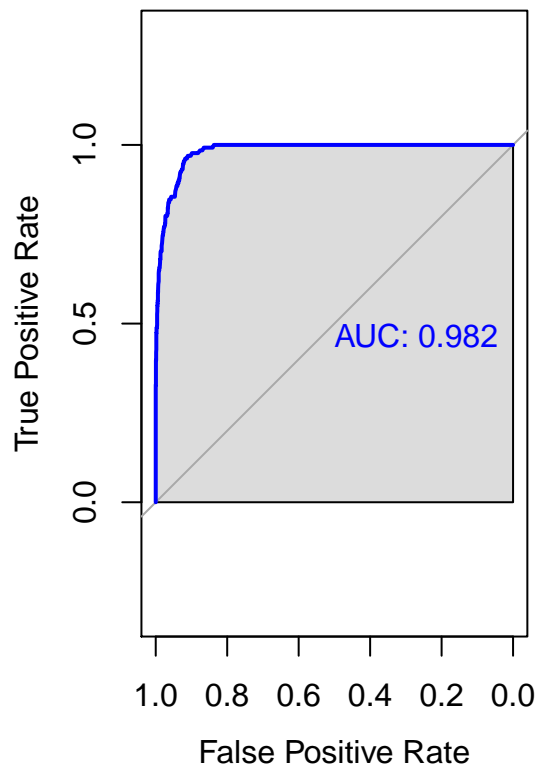
Model 1: Test ROC for RF Classification



rf.seismic



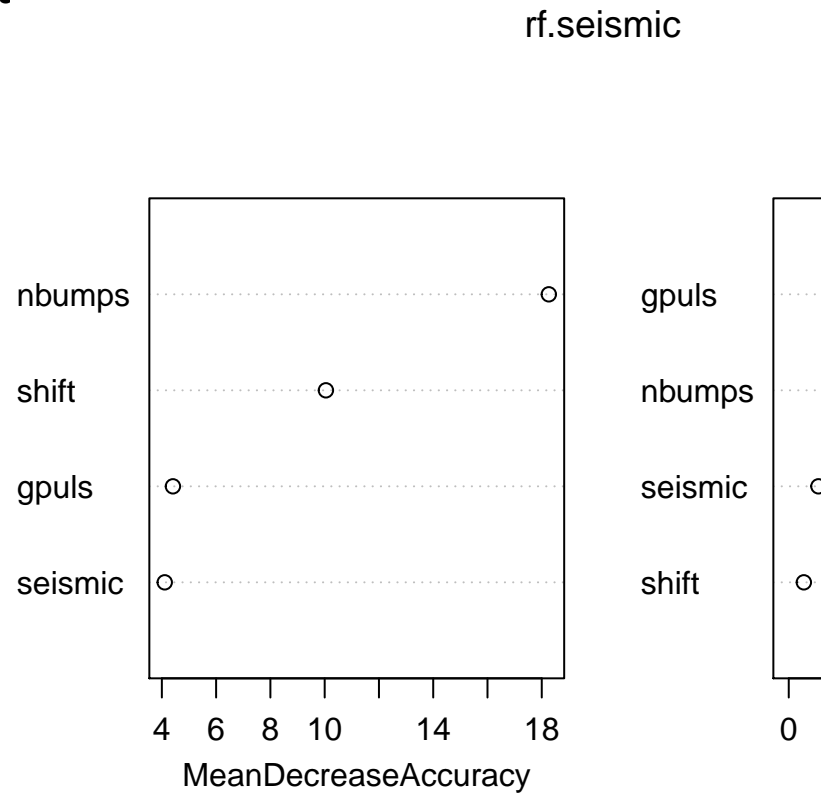
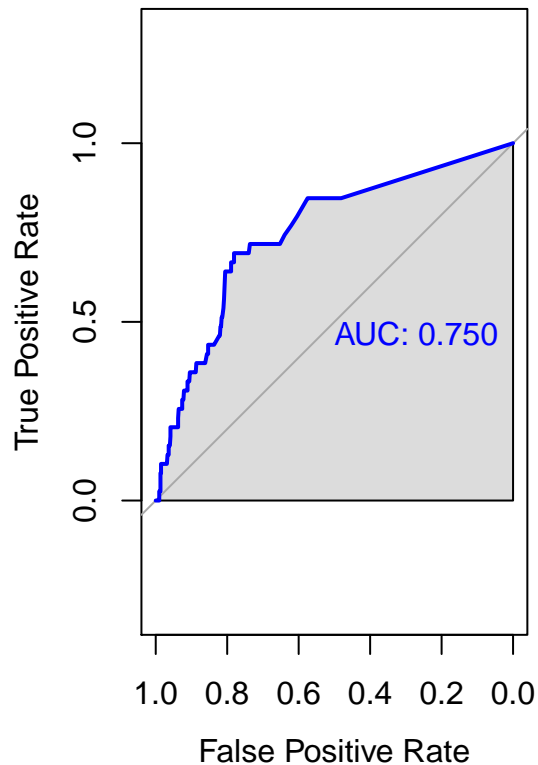
Model 2: Train ROC for RF Classification



0 1 MeanDecreaseAccuracy MeanDecreaseGini

seismic	0.7571748	9.478397	4.105877	5.821832
shift	11.4969471	-9.940458	10.045574	2.959784
gpuls	1.6136431	7.619120	4.407885	75.308097
nbumps	12.7174931	21.298900	18.259825	27.149236

Model 2: Test ROC for RF Classification



Support vector classifier and support vector machine

```
# Variable selection and refitting
#model1 = genenergy + 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")
```

```

    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: 0.06760857
##
## - Detailed performance results:
##   cost      error dispersion
## 1 0.001 0.06760857 0.01645615
## 2 0.010 0.06760857 0.01645615
## 3 0.100 0.06760857 0.01645615
## 4 1.000 0.06760857 0.01645615
## 5 5.000 0.06760857 0.01645615

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
```

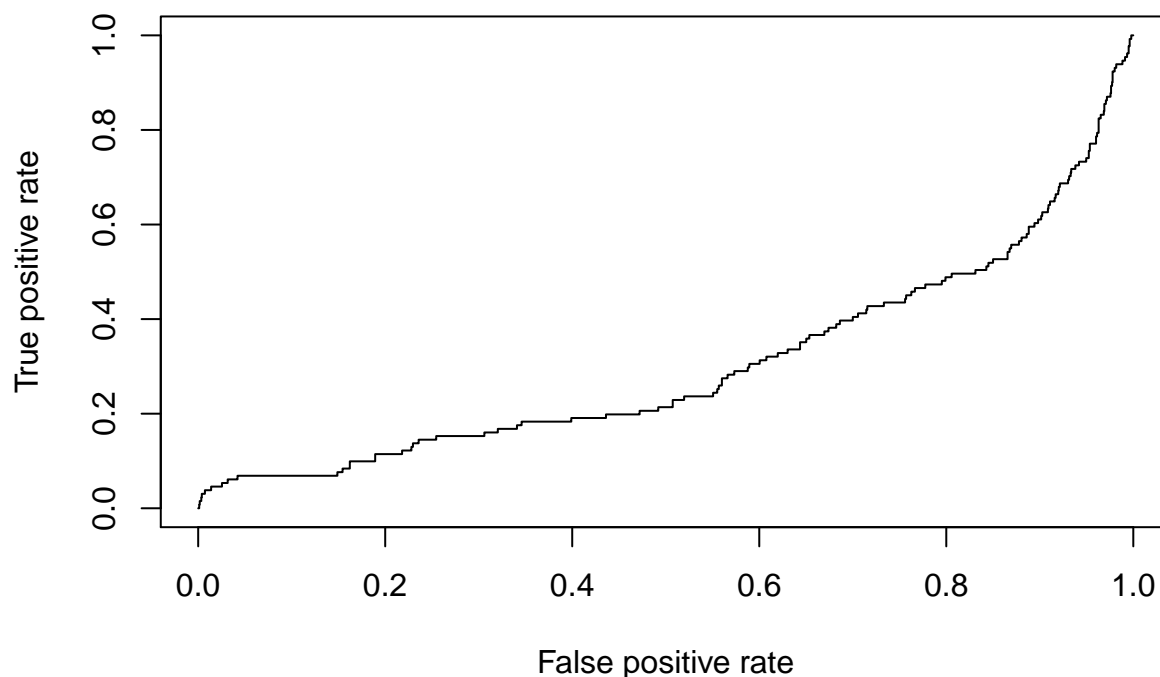
```
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

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

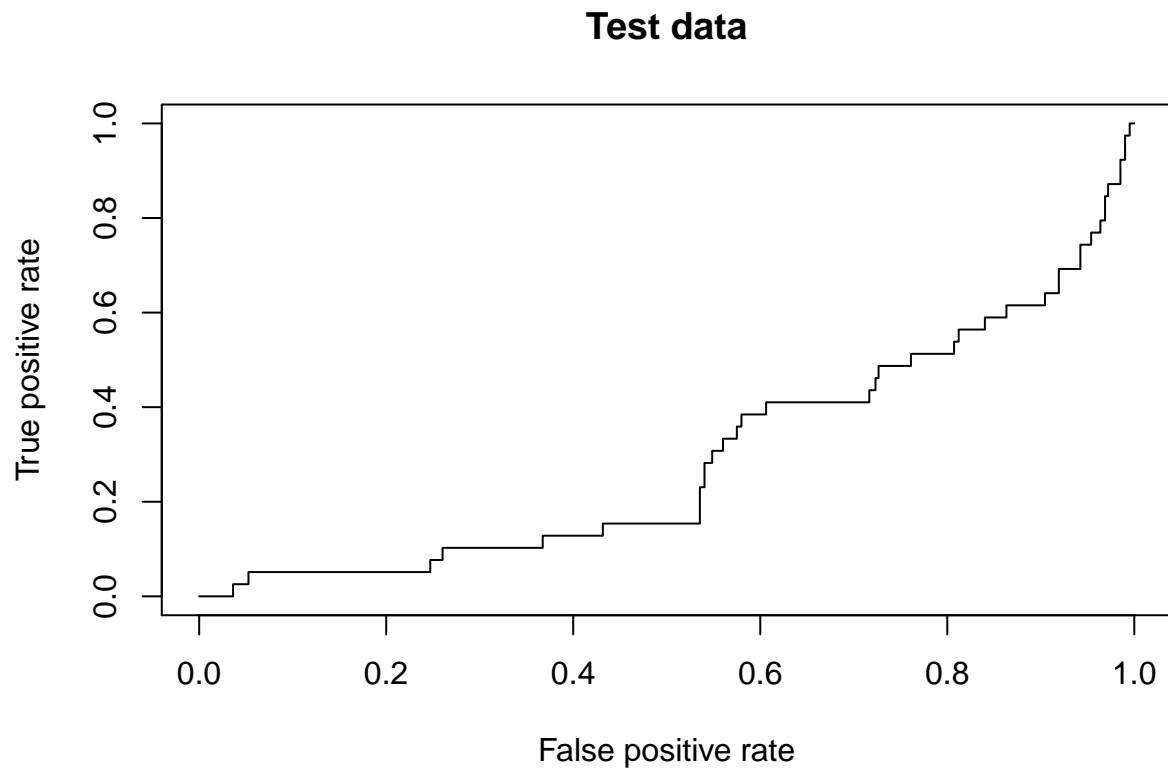
```
svmfit.best1 <- svm(factor(class)~genergy + gpuls + nbumps + nbumps2 + nbumps4, data = seismic[train,],
fitted1 <- attributes(predict(svmfit.best1, seismic[train,], decision.values = T))$decision.values
fitted.test1 <- attributes(predict(svmfit.best1, seismic[-train,], decision.values = T))$decision.values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

```
# 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")
```

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: 0.0676059
##
## - Detailed performance results:
```

```
##      cost      error dispersion
## 1 0.001 0.0676059 0.01815001
## 2 0.010 0.0676059 0.01815001
## 3 0.100 0.0676059 0.01815001
## 4 1.000 0.0676059 0.01815001
## 5 5.000 0.0676059 0.01815001
```

```
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
```

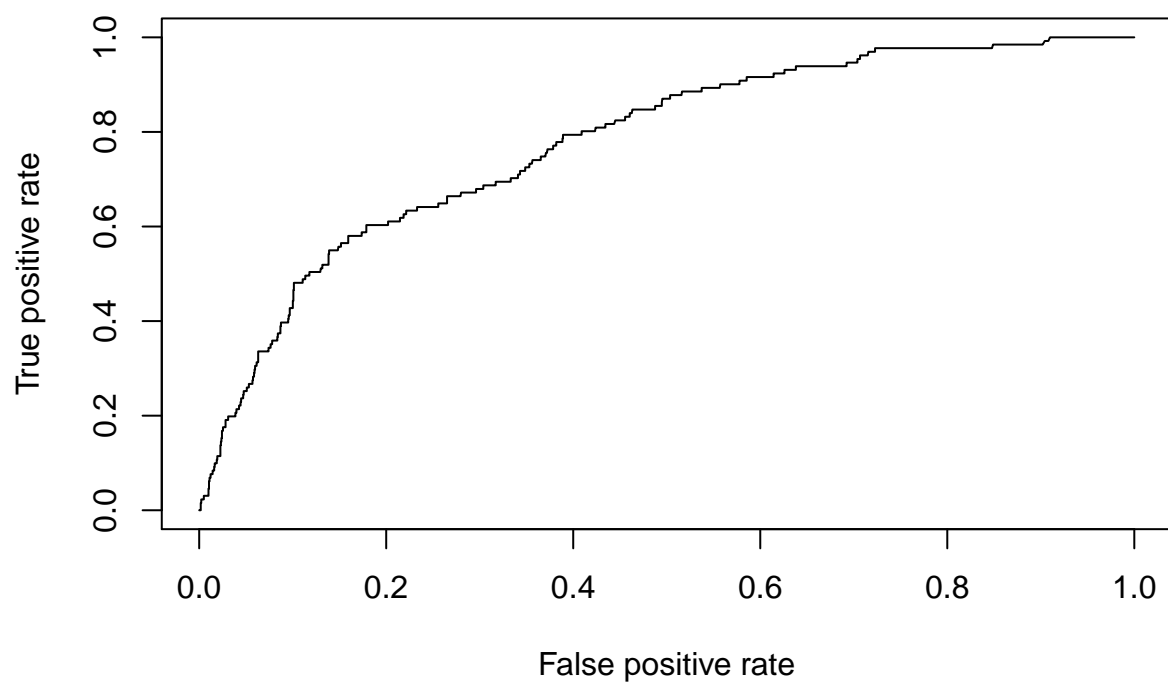
```
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

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

```
svmfit.best2 <- svm(factor(class)~seismic + shift + gpuls + nbumps, data = seismic[train,], kernel = "linear")
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

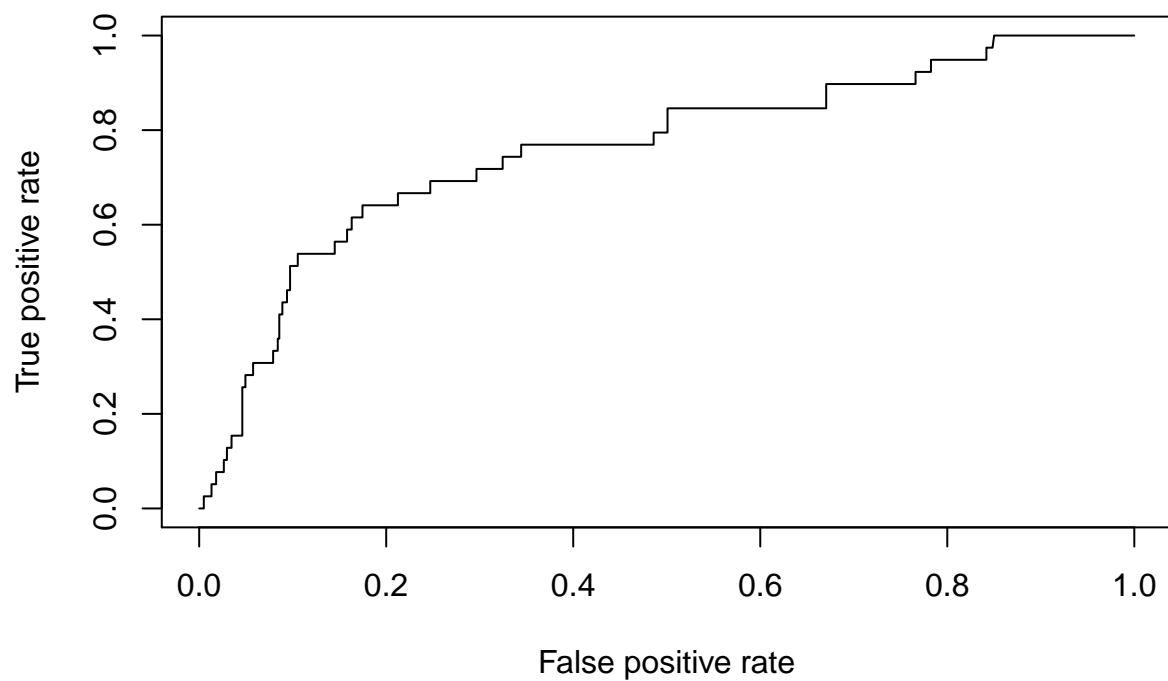
# This one shows a much better ROC curve
# But it still looks bad just from the original table produced
rocplot(fitted2, seismic[train,"class"], main = "Training data")
```

Training data



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

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,],
                 kernel = "radial", cost = 1, decision.values = T)

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

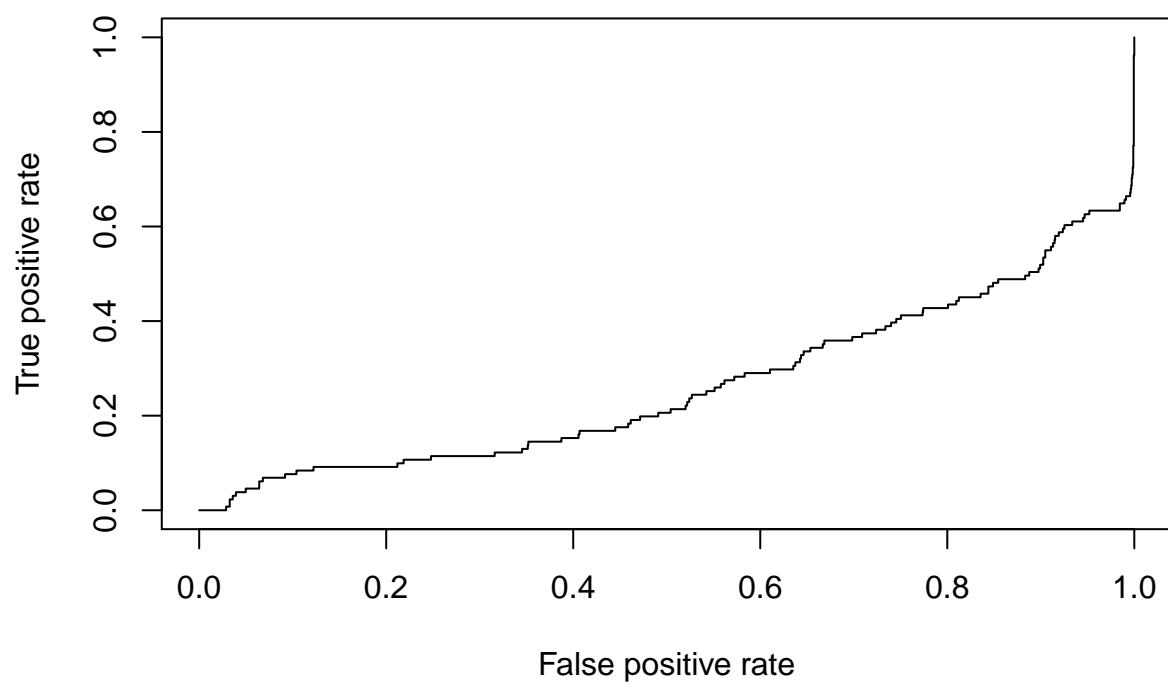
##          truth
## predict    0    1
##          0 607  39
##          1   0   0

svmrad2 <- svm(factor(class)~genergy + gpuls + nbumps + nbumps2 + nbumps4, data = seismic[train,], kernel = "radial",
               cost = 1, decision.values = T)
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")

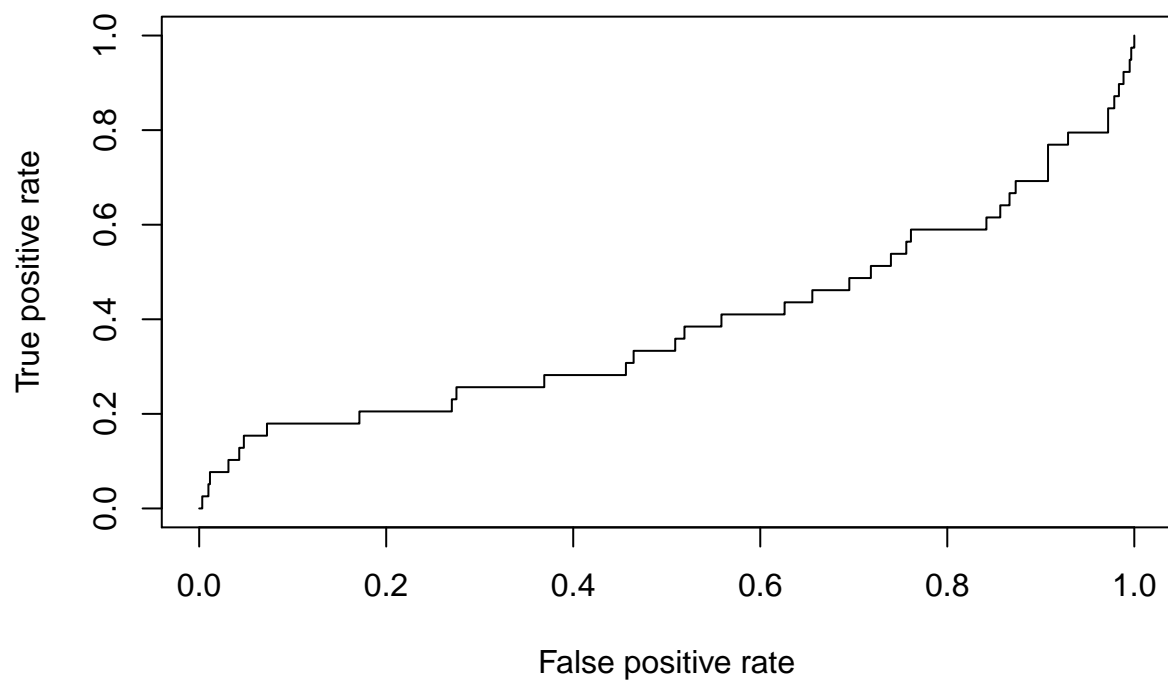
```

Training data



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

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

```

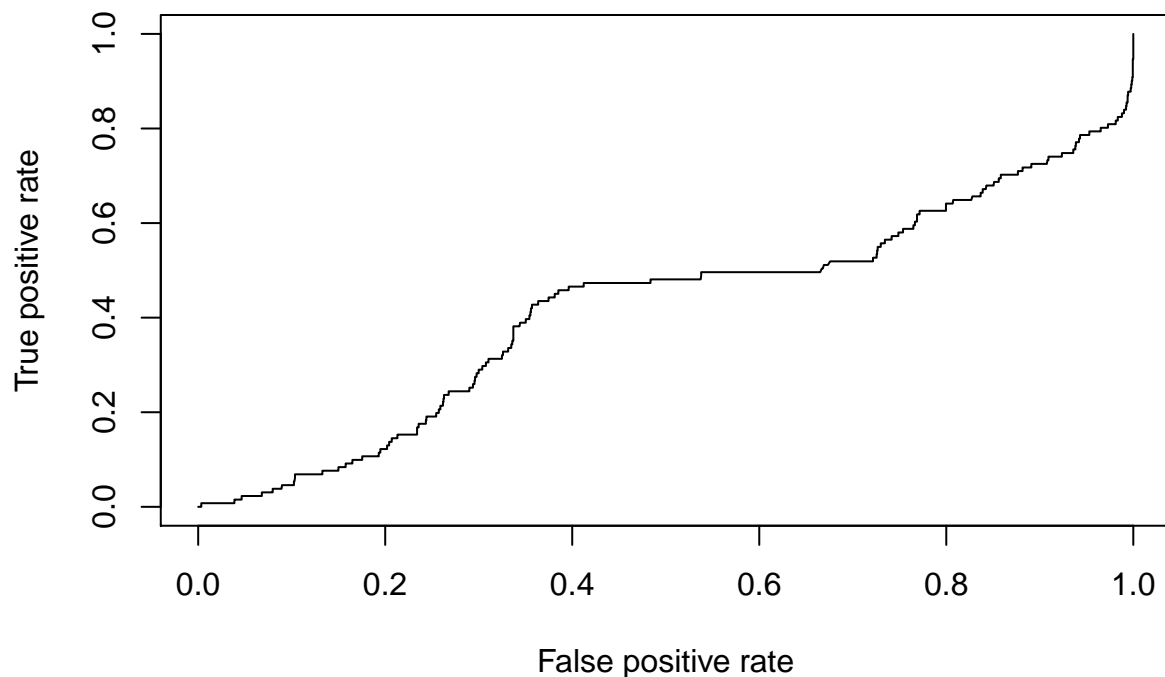
```

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

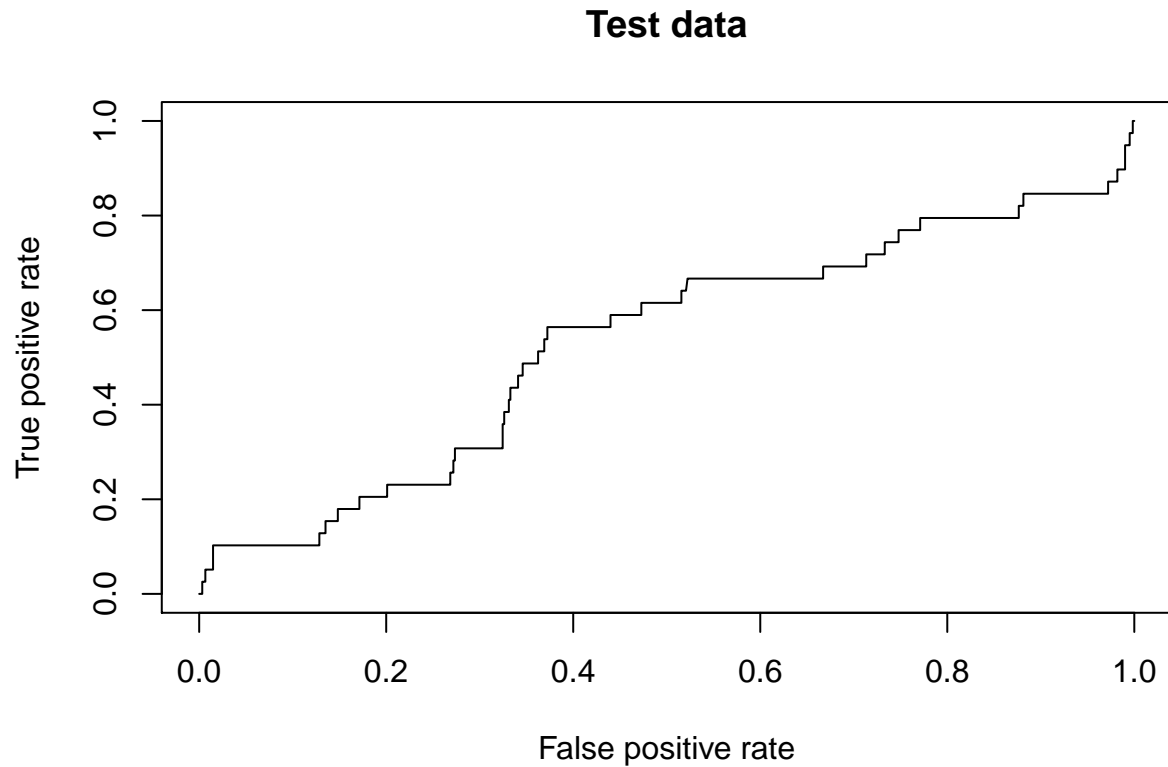
rocplot(fitted3, seismic[train,"class"], main = "Training data")

```

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

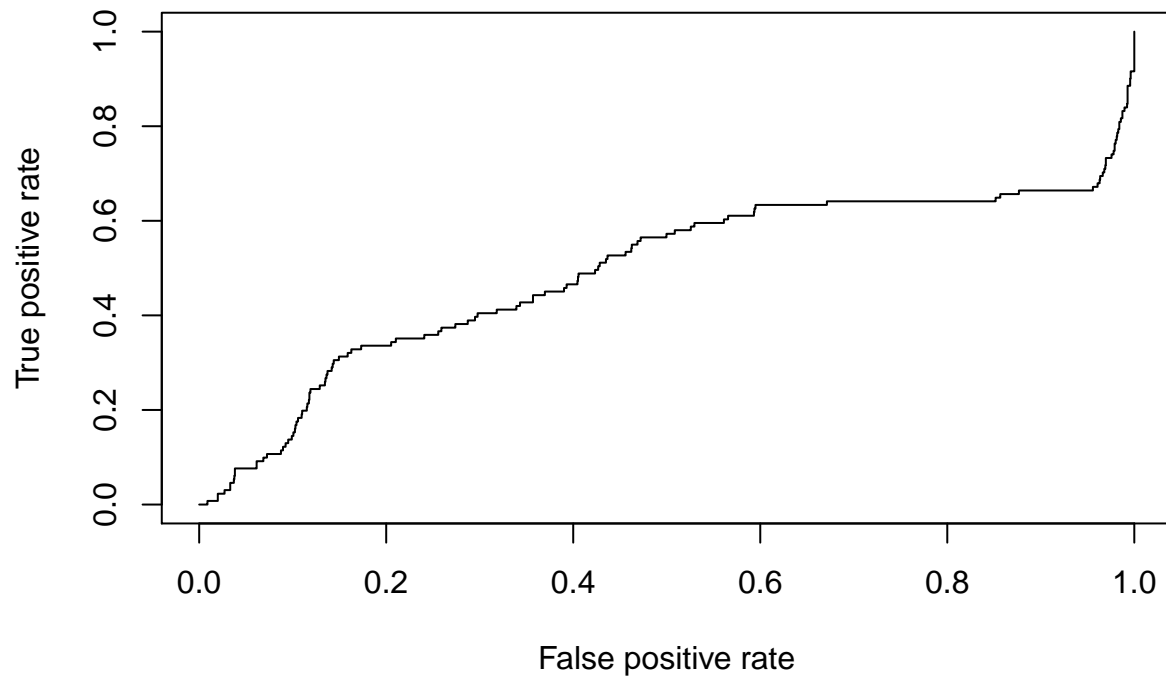
bestmod <- tune.out4$best.model
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

```
##      truth
## predict  0  1
##      0 606 39
##      1   1  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

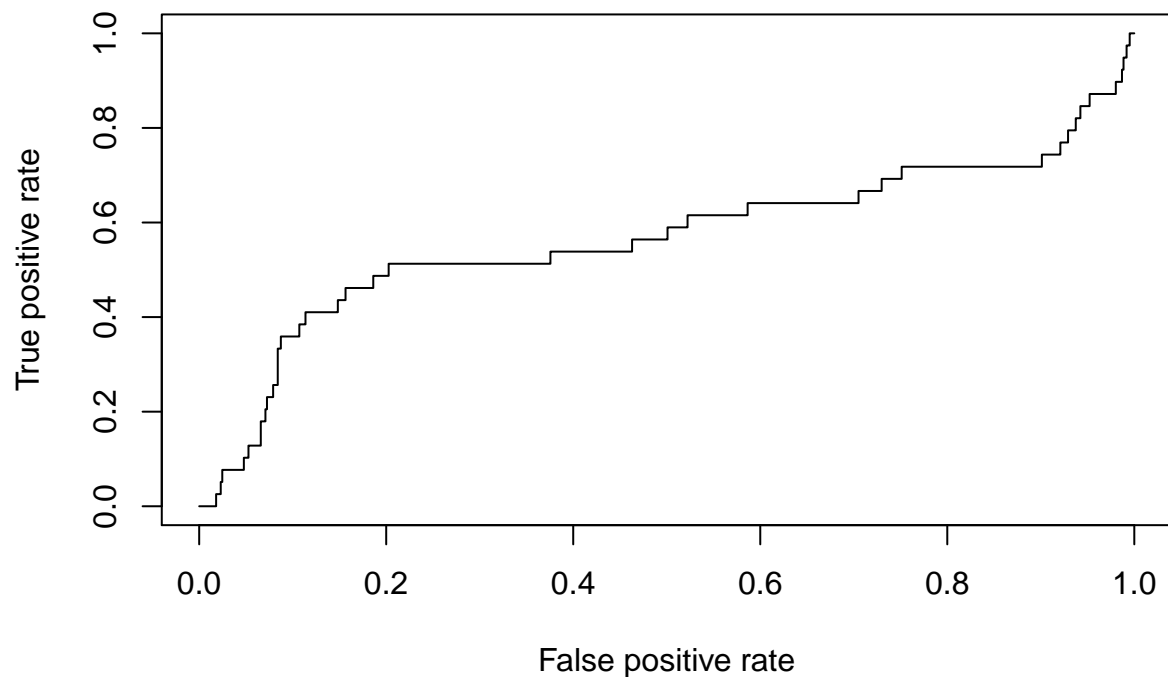
rocplot(fitted4, seismic[train,"class"], main = "Training data")
```

Training data



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


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 = "poly")

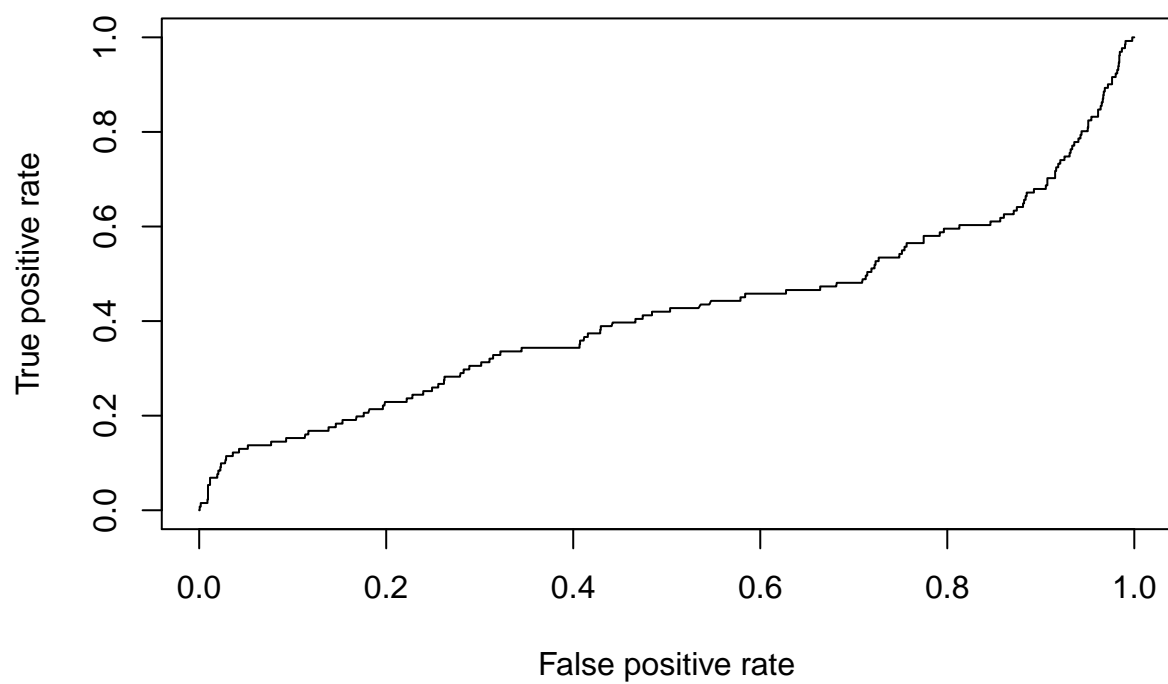
bestmod <- tune.out5$best.model
ypred <- predict(bestmod, seismic[-train,])
table(predict = ypred, truth = seismic$class[-train])
```

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

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

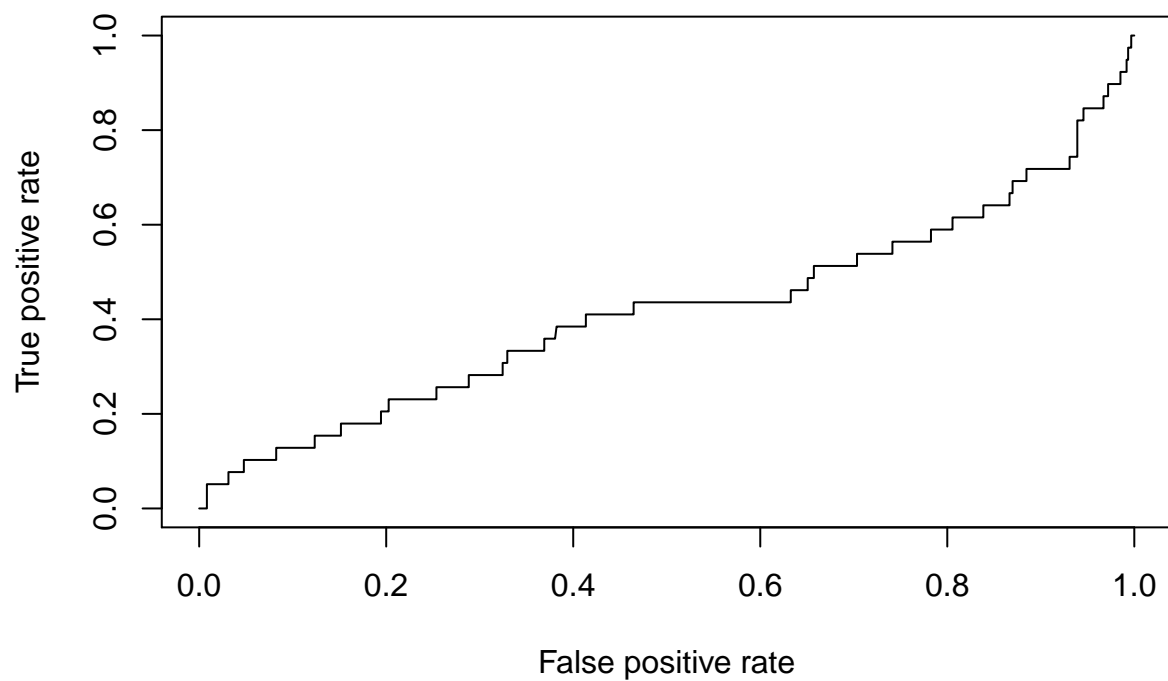
rocplot(fitted5, seismic[train,"class"], main = "Training data")
```

Training data



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

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
```

Appendix

Table I. 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, ‘d’ - very high hazard
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 longwall
gpuls	number of pulses recorded within previous shift by GMax
gdenenergy	deviation of recorded energy within previous shift from average energy recorded during eight previous shifts
gdpuls	deviation of recorded pulses within previous shift from average number of pulses recorded during eight previous shifts
ghazard	result of shift seismic hazard assessment by the seismoacoustic method based on registration coming from the previous shift
nbumps	the number of seismic bumps recorded within previous shift
nbumps i , $i \in \{1, \dots, 5\}$	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’), ‘0’ - no high energy seismic bump occurred in the next shift

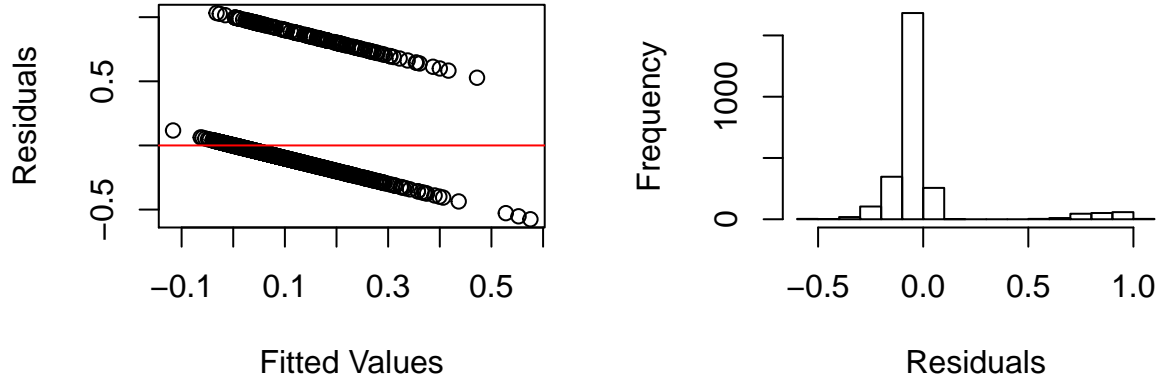


Table 2: Table II-VIFs of Linear Model

seismic	seismoacoustic	shift	genergy	gpuls	gdenergy	gdpuls
1.21	1.29	1.41	2.89	4.06	3	3.43

Table 3: Table II-VIFs of Linear Model

ghazard	nbumps	nbumps2	nbumps3	nbumps4	nbumps5	energy	maxenergy
1.4	2414.69	798.96	769.13	104.4	11.56	110.28	93.76