Lab 2 - Elias Alesand, elial148

Assignment 2

2.

Confusion matrices for Train and Test datasets using Deviance and Gini measurements:

Using **Deviance** for dataset **Train**:

	Yfit	
	bad	good
bad	61	86
good	20	333

Misclassification rate: 0.212

Using **Gini** for dataset **Train**:

	Yfit	
	bad	good
bad	66	81
good	38	315

Misclassification rate: 0.238

Using **Deviance** for dataset **Test**:

	Yfit	
	bad	good
bad	28	48
good	19	155

Misclassification rate: 0.268

Using **Gini** for dataset **Test**:

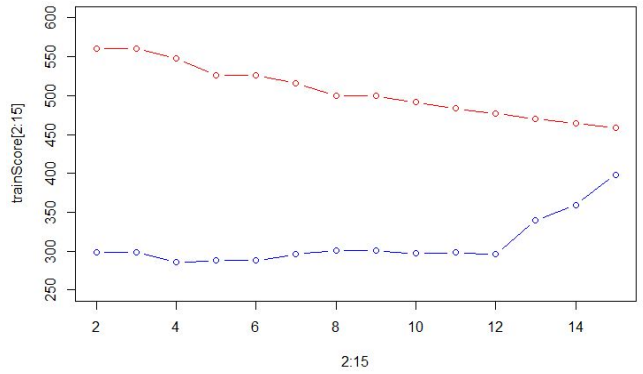
	Yfit	
	bad	good
bad	18	58
good	34	140

Misclassification rate: 0.368

I can see that using Deviance as a measure of impurity gives better results for both training and test dataset.

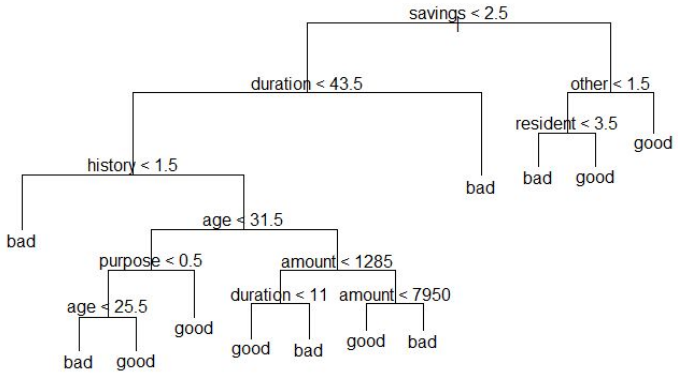
3.

Plot of the score of the training data and validation data as a function of the amount of terminal nodes. Training data is red and validation data is blue.



I see that the model is overfitted to the training data after 12 since the score of the testing data goes up. So i would say that the optimal number of leaves is 12.

The best tree with 12 leaves has a depth of 6 as shown below.



I see for example that if a customers savings is above 2.5 the decision tree is quite simple, otherwise it depends on a lot of different variables.
Predicting the customer behavior for dataset test gives the following results:

	Yfit2	
	bad	good
bad	32	44
good	23	151

With a misclassification rate of 0.268.

4.
Confusion matrices when using naive bayes classification:

Dataset **train**:

	Yfit	
	bad	good
bad	95	52
good	98	255

Misclassification rate: 0.3

Dataset **test**:

	Yfit	
	bad	good
bad	46	30
good	49	125

Misclassification rate: 0.316

The misclassification rate is worse than in the tree model in step 3 for both datasets.

5.

Confusion matrices when a loss function is used:

Dataset **train:**

	naiveLossP	
	bad	good
bad	137	10
good	263	90

Misclassification rate: 0.546

Dataset **test:**

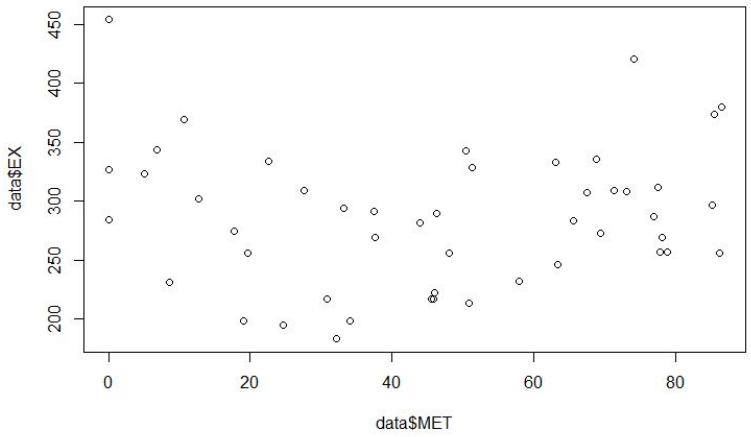
	naiveLossP	
	bad	good
bad	71	5
good	122	52

Misclassification rate: 0.508

The loss function penalize false positives so fewer cases will be classified as good. The misclassification rate is larger but this is expected since the classifier might class something as bad even though it is probably good.

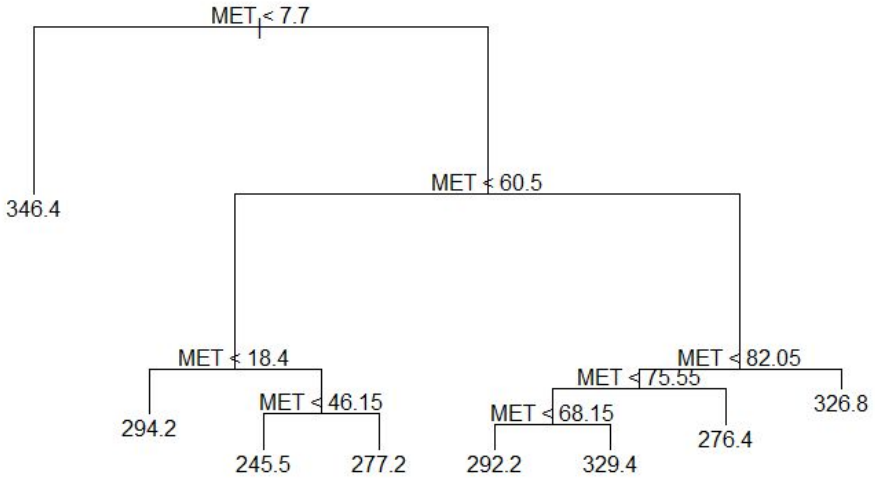
Assignment 3

1. Plot of EX vs MET

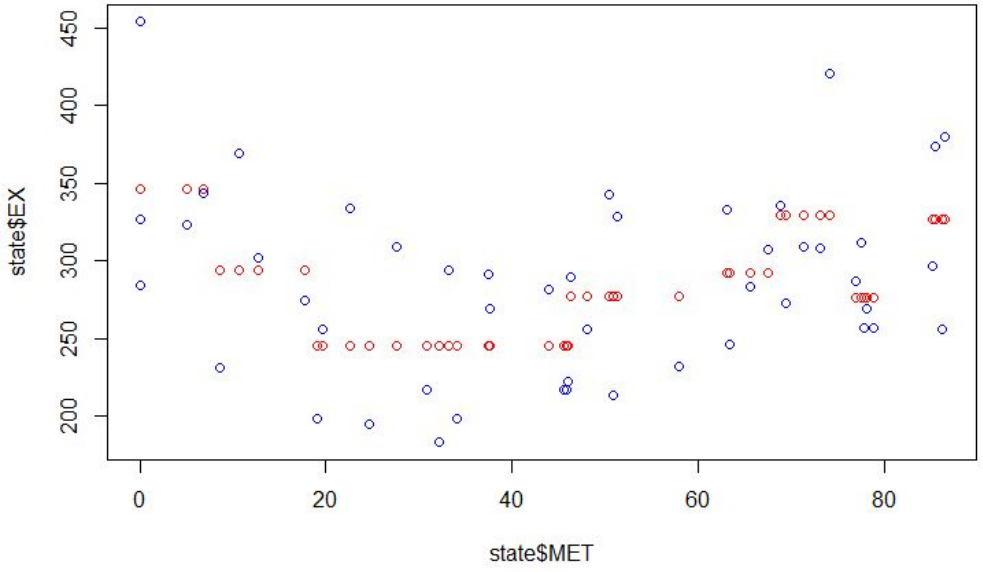


A model that i think can describe this data is a quadratic equation with some distribution. However I think there are too few data points to conclude exactly how the points are distributed.

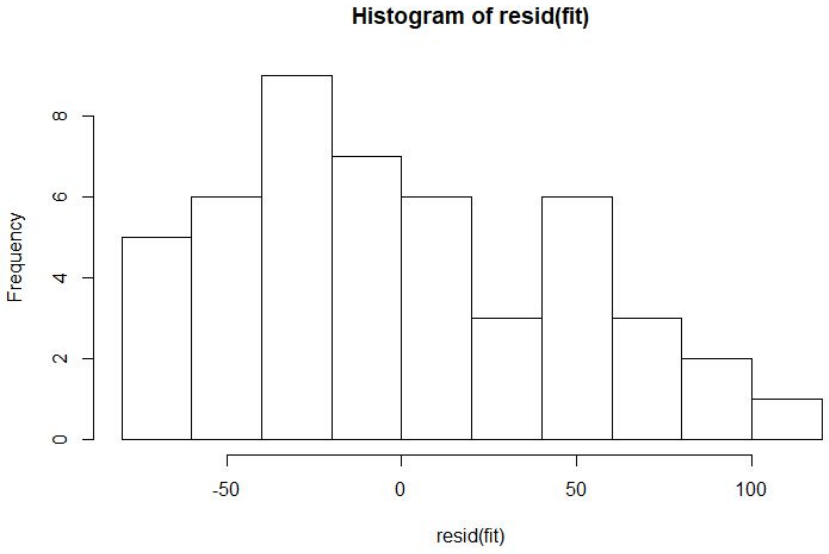
2. When fitting a regression tree model using package **tree** i get the following selected tree:



By plotting the original data and the fitted data in the same plot i can see the quality of the fit:

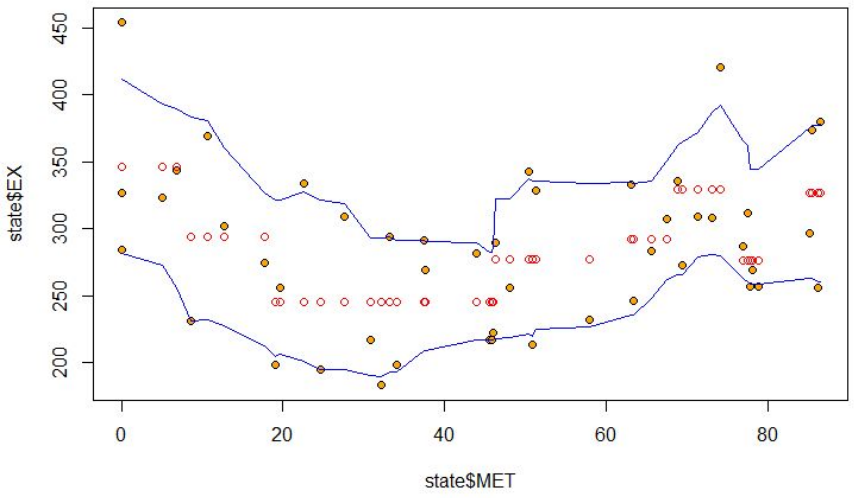


It does a decent job at fitting but the actual data points are distributed over large intervals so a lot of data points are quite far from the predicted data points.
Plotting histogram of residuals:



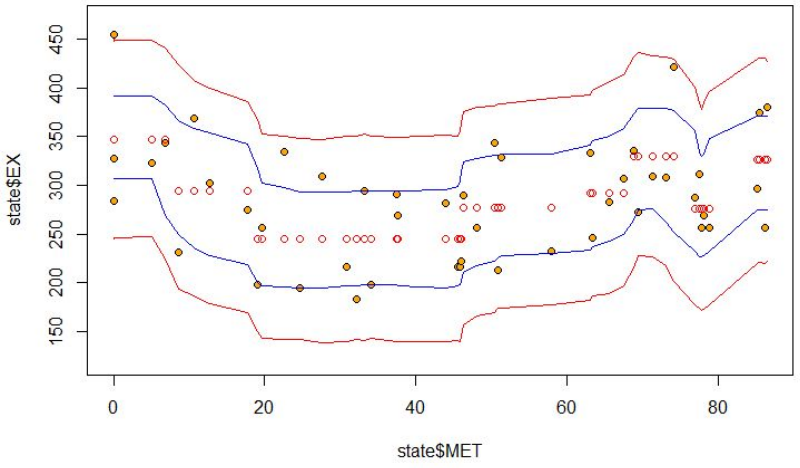
It does not look like the residuals are normally distributed. It is closer to a shape of log-normal, poisson or gamma distribution.

3. Plotting the confidence bands and the original data and the fitted tree model using non-parametric bootstrapping:



The confidence bands are not very smooth. I would guess that this is because there are not a lot of data points so any outlier will have a large effect on the bands. Most of the data points are inside the confidence bands, so i would say that the model is reliable.

4. Plotting the confidence bands and the original data and the fitted tree model using parametric bootstrapping. The prediction band is shown in red and the confidence band in blue:

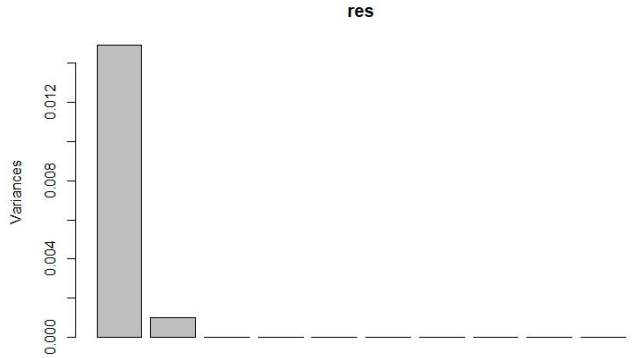


Less data points are inside the confidence bands this time so it might be less reliable than the non-parametric bootstrap, however the majority of data points are still inside. Only one data point is outside the prediction band which would be around 2% ($1/48 \approx 0.02$). It should not necessarily be the case that 5% of data points are outside the prediction band since it has more to do with where the normal distribution is cut off.

5. In the histogram it does not look like the data is normally distributed so i would suggest to use parametric bootstrap with some other distribution. Maybe log-normal, poisson or gamma distribution.

Assignment 4

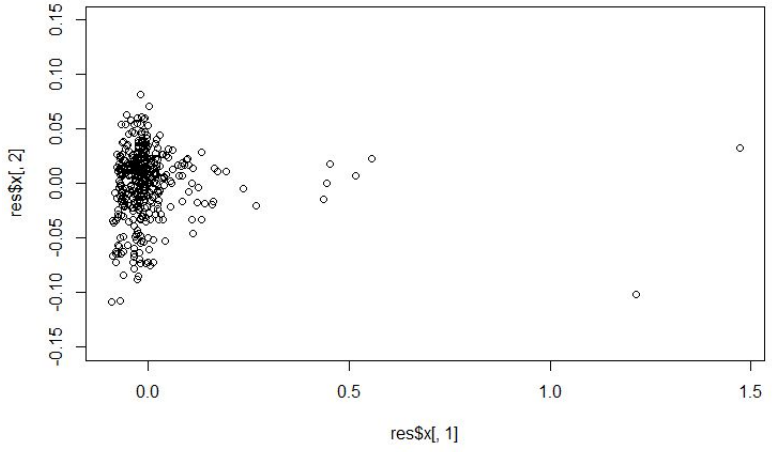
1. A standard PCA of the data gives the following results:




```
 [1] "93.332" "6.263"  "0.185"  "0.101"  "0.068"  "0.025"  "0.009"  "0.003"  "0.003"
[10] "0.002"  "0.001"  "0.001"  "0.001"  "0.001"  "0.000"  "0.000"  "0.000"  "0.000"
[19] "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"  "0.000"
```

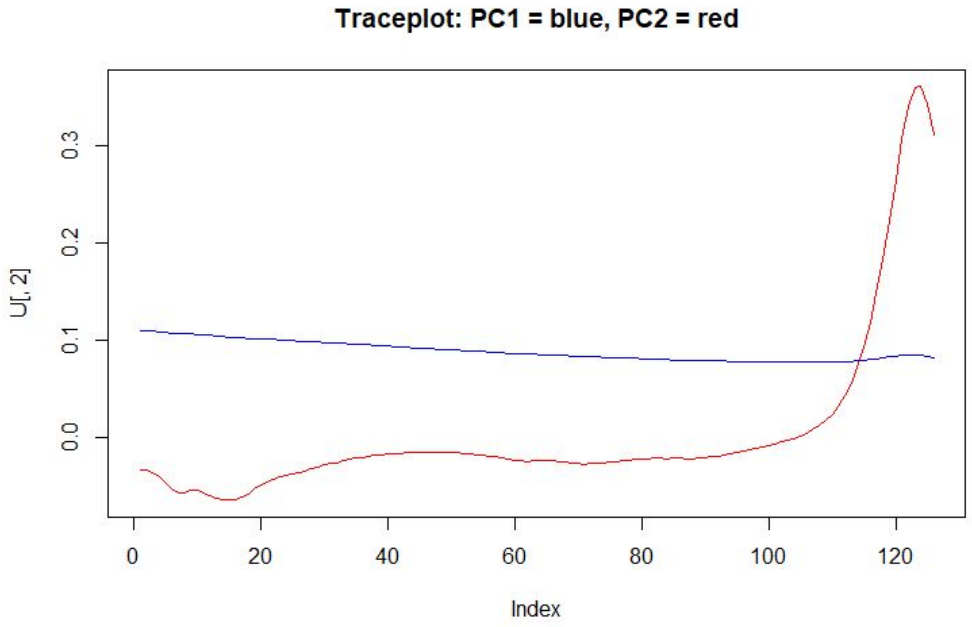
The plot is not detailed enough to see how many PC should be extracted but i see that two components stand out. By printing the variances I see that two components explain more than 99% of the total variance and I see that 14 components have variances larger than 0.

Plot of the scores in coordinates (PC1, PC2):



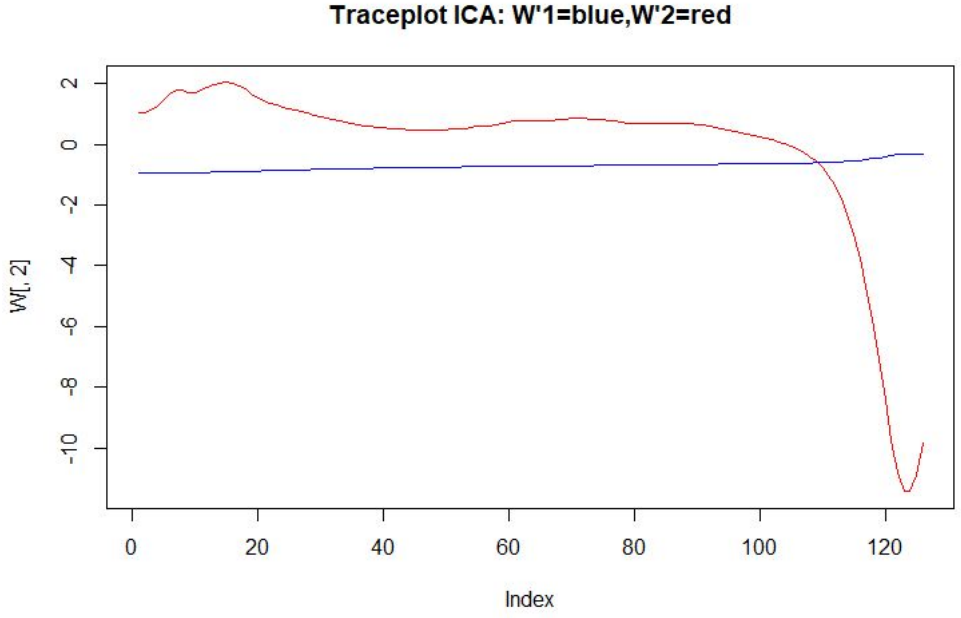
A couple of diesel fuel samples are unusual since some data points are far away from the rest. This could also be explained by errors in measurement.

2. Traceplot showing how PC1 and PC2 depend on the different features:



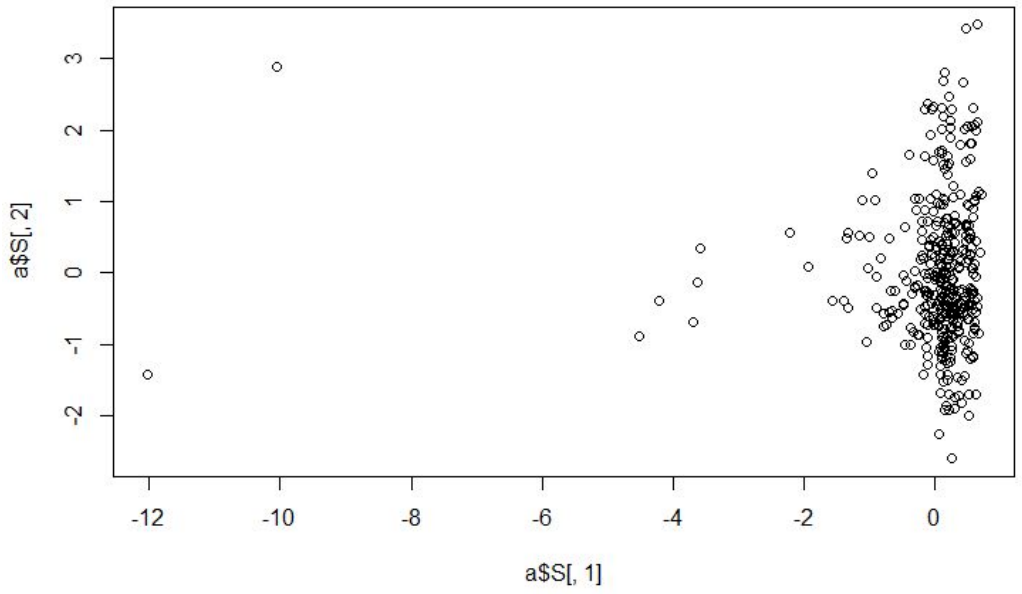
PC1 is fairly constant for all features but PC2 seems to depend more on features above ~110.

3. a:



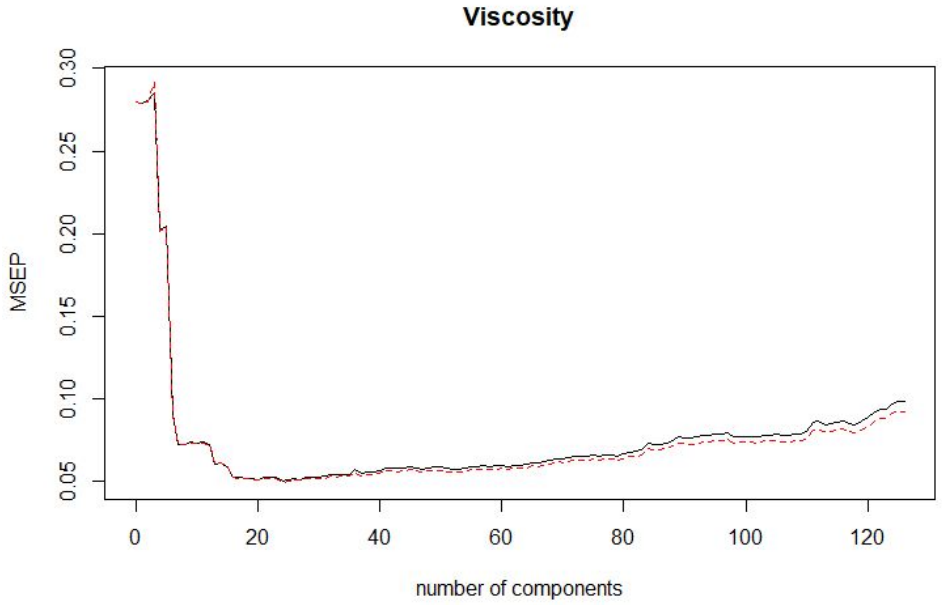
As in step two one of the lines is fairly constant while the other looks differently above around 110. This plot looks like an inversion of the one in step 2 with a different scale on the Y-axis. W' measures the components dependency on the different spectrum features much like the traceplots in step 2.

b: Plot of the scores of the first two latent features.



It looks very similar to the plot in step 1 but it is mirrored and the scales on the Y- and X axis are different.

4. Plot showing the dependence of the mean-square predicted error of the PCR model as a function of the number of components chosen by cross validation.



The error drops off a lot around 10 chosen components and stabilizes around 20 and then slowly goes up again as the components increase. I think that 20 components is a reasonable amount to choose to keep the complexity low while still having a low error.

Code appendix:

Assignment 2:

```
library(readxl)
library(tree)
library(MASS)
library(e1071)
creditscores = read_excel("creditscoring.xlsx")
creditscores$good_bad=as.factor(creditscores$good_bad)
#_____Dividing into training/validation/test data____
n=dim(creditscores)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.5))
train=creditscores[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=creditscores[id2,]
id3=setdiff(id1,id2)
test=creditscores[id3,]

#____Fitting decision tree to training data____
n=dim(train)[1]
fit=tree(good_bad~.,data=train,split = "deviance")
#fit=tree(good_bad~.,data=train,split = "gini")
plot(fit)
text(fit, pretty=0)
fit
summary(fit)

#____Creating confusion matrix to figure out the misclassification rate
Yfit=predict(fit, newdata=test, type="class")
table(test$good_bad,Yfit)

#____Chosing optimal tree

trainScore=rep(0,15)
testScore=rep(0,15)
for(i in 2:15) {
  prunedTree=prune.tree(fit,best=i)
  pred=predict(prunedTree, newdata=valid,
               type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
}
plot(2:15, trainScore[2:15], type="b", col="red",
     ylim=c(250,600))
points(2:15, testScore[2:15], type="b", col="blue")

finalTree=prune.tree(fit, best=12)
```

```
plot(finalTree)
summary(finalTree)
text(finalTree, pretty=0)
Yfit2=predict(finalTree, newdata=test, type="class")
table(test$good_bad,Yfit2)

#____Naive bayes classification
fit=naiveBayes(good_bad~., data=train)
fit
Yfit=predict(fit, newdata=train)
table(train$good_bad,Yfit)

lossMatrix = matrix(c(0,1,10,0),nrow=2,ncol=2)

naiveLoss = predict(fit, newdata=train, type='raw')
naiveLoss = naiveLoss %*% lossMatrix
naiveLossP = factor(ifelse(naiveLoss[,1] > naiveLoss[,2], 2, 1), labels=c("bad",
"good"))
print(table(train$good_bad,naiveLossP))
```

Assignment 3:

```
library(tree)
library(boot)
library(readxl)
library(CIplot)
state = read_excel("state.xlsx")
#___Reordering data according to MET and plotting EX vs MET
state = state[order(state$MET),]
plot(state$MET,state$EX,col="blue")
set.seed(12345)

#___Fitting regression tree model
fit = tree(EX~MET,state,control = tree.control(48,minsize=8))
plot(fit)
text(fit,pretty = 0)
Yfit=predict(fit, newdata=state)
plot(state$MET,state$EX,col="blue")
points(state$MET,Yfit,col="red")
hist(resid(fit))
#___Non-parametric bootstrap
f=function(data, ind){
  set.seed(12345)
  data1=data[ind,]# extract bootstrap sample
  bootFit = tree(EX~MET,data1,control = tree.control(48,minsize=8)) #fit model
  prediction=predict(bootFit,newdata=state)
  return(prediction)
}
res=boot(state, f, R=1000) #make bootstrap
plot(res)
e=envelope(res)

plot(state$MET,state$EX,pch=21,bg="orange")
points(state$MET,Yfit,col="red")
points(state$MET,e$point[2,], type="l", col="blue")
points(state$MET,e$point[1,], type="l", col="blue")
#___Parametric bootstrap

mle=fit
rng=function(data, mle) {
  data1=data.frame(EX=data$EX,
                   MET=data$MET)
  n=length(data$EX)
  data1$EX=rnorm(n,predict(mle,newdata=data1),sd(resid(mle)))
  return(data1)
}
f1=function(data1){
  res=tree(EX~MET,data1,control = tree.control(48,minsize=8))
  predicted=predict(res,newdata=state)
  return(predicted)
}
res=boot(state, statistic=f1, R=1000,
```

```
         mle=mle,ran.gen=rng, sim="parametric")

e=envelope(res)

plot(state$MET,state$EX,pch=21,bg="orange", ylim = c(120,470))
points(state$MET,Yfit,col="red")
points(state$MET,e$point[2,], type="l", col="blue")
points(state$MET,e$point[1,], type="l", col="blue")
#__Prediction band
f2=function(data1){
  res=tree(EX~MET,data1,control = tree.control(48,minsize=8))
  priceP=predict(res,newdata=state)
  n=length(state$EX)
  predictedP=rnorm(n,priceP, sd(resid(mle)))
  return(predictedP)
}
res2=boot(state, statistic=f2, R=10000,
         mle=mle,ran.gen=rng, sim="parametric")
e2=envelope(res2)
points(state$MET,e2$point[2,], type="l", col="red")
points(state$MET,e2$point[1,], type="l", col="red")
```

Assignment 4:

```
library(readxl)
library(fastICA)
library(pls)
NIRSpectra = read_excel("NIRSpectra.xlsx")
data1=NIRSpectra
#__Standard PCA
data1$Viscosity=c()
res=prcomp(data1)
lambda=res$sdev^2
sprintf("%2.3f",lambda/sum(lambda)*100)
screeplot(res)

plot(res$x[,1], res$x[,2], ylim=c(-0.15,0.15))

#__Traceplot of loadings
U=res$rotation
plot(U[,2], main="Traceplot: PC1 = blue, PC2 = red",type="l",col="red")
lines(U[,1],col="blue")

#__ICA
a = fastICA(data1, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
            method = "R", row.norm = FALSE, maxit = 200, tol = 0.0001, verbose = TRUE)
W=a$K%*%a$W
plot(W[,2], main="Traceplot ICA: W'1=blue,W'2=red",type ="l",col="red")
lines(W[,1],col="blue")
plot(a$S[,1],a$S[,2])
#__Fitting PCR model

set.seed(12345)
pcr.fit=pcr(Viscosity~., data=data1,validation="CV")
summary(pcr.fit)
validationplot(pcr.fit,val.type="MSEP")
```