Statement of Contribution

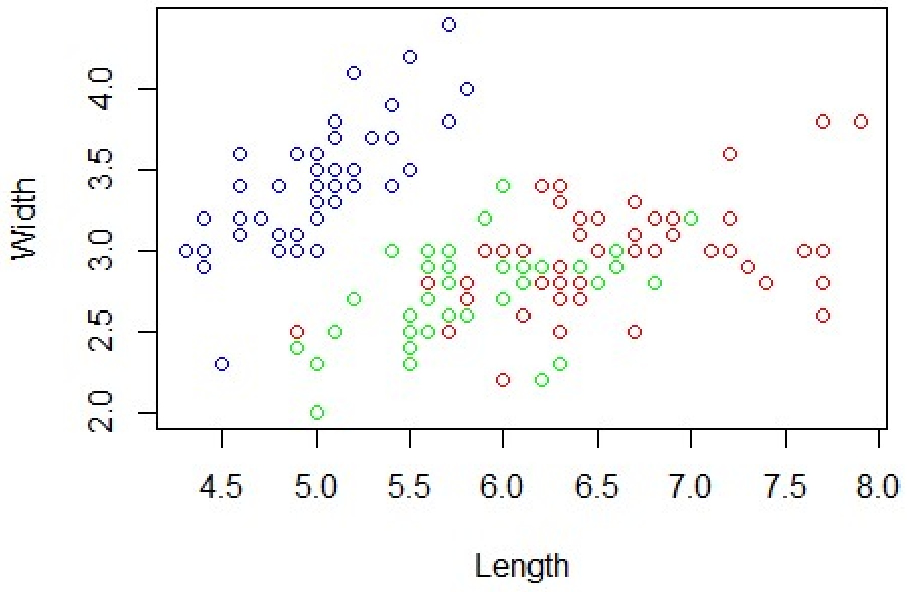
Alexander Bois is responsible for assignment 1, Daniel Bissessar for assignment 2 and Filip Brunander for assignment 3. Moreover, several programming sessions was held where we discussed problems and helped each other. However, the help and discussions where limited where the one responsible for the assignment also wrote all the code and answered all questions related to that assignment. Sessions were held to explain the different assignments to each group member.

Lab 2 assignment 1

Responsible: Alexander Bois

Task 1

```r
plot(iris$Sepal.Length, iris$Sepal.Width, xlab = "Length", ylab = "Width")
points(setosa$Sepal.Length, setosa$Sepal.Width, col = "blue")
points(versicolor$Sepal.Length, versicolor$Sepal.Width,  col = "green")
points( virginica$Sepal.Length, virginica$Sepal.Width, col = "red")
```



It does seem like the Setosa(blue seems to be able to classify clearly using LDA. The Versicolor(green) and Virginica(red) is going to be harder to more clearly seperate, it seems from just inspecting the data, since they are ,uch more entangled along all dimensions in the plot.

Task 2

a)

First we define the functions to calculate the means, Covariance matrix and prior probabilities for the classes.

```r
ClassMean <- function(x){
  return(c(mean(x$Sepal.Length), mean(x$Sepal.Width)))
}

cov_matrix <- function(x){
  mat <- cbind(x$Sepal.Length, x$Sepal.Width)
  return(cov(mat, method = "pearson"))
}

prior_prob <- function(x){
  return(nrow(x)/nrow(iris))
}
```

With these we calculate the means, covariance matrices and prior probabilities. Means are reported: Length, Width

Setosa mean: 5.006, 3.428

Versicolor mean: 5.936, 2.770

Virginica mean: 6.588 2.974

Setosa covariance matrix:

```r
setosa_cov

##            [,1]       [,2]
## [1,] 0.12424898 0.09921633
## [2,] 0.09921633 0.14368980
```

Versicolor covariance matrix:

```r
versicolor_cov

##            [,1]       [,2]
## [1,] 0.26643265 0.08518367
## [2,] 0.08518367 0.09846939
```

Virginica covariance matrix:

```
virginica_cov

##             [,1]       [,2]
## [1,] 0.40434286 0.09376327
## [2,] 0.09376327 0.10400408
```

Prior probabilities are the same for all classes: 1/3

b)

Then we compute the pooled(combined) covariance matrix for the classes.

```
pooled_cov <- (setosa_cov+versicolor_cov+virginica_cov)/3
pooled_cov

##             [,1]       [,2]
## [1,] 0.26500816 0.09272109
## [2,] 0.09272109 0.11538776
```

c)

The probablistic model for LDA is:

$$x|y = C_i, \mu_i, \Sigma \sim N(\mu_i, \Sigma)$$
$$y|\pi \sim Multinomial(\pi_1, .., \pi_k)$$

d)

The discriminant functions were computed for each class. Please see the code for refrence.

e)

The equations of decision boundaries between classes are as follows.The equation is 0 when the point x is on the line. Between Setosa and versicolor:

```
set_ver_eq <- function(x){
  set_w <- weights(setosa_meanLW, setosa_prior_prob)
  ver_w <- weights(versicolor_meanLW, versicolor_prior_prob)
  res <- t(set_w[1:2]-ver_w[1:2])%*%x + (set_w[3]-ver_w[3])
  return(res)
}
```

Between Vesicolor and Virginica :

```
ver_vir_eq <- function(x){
  ver_w <- weights(versicolor_meanLW, versicolor_prior_prob)
  vir_w <- weights(virginica_meanLW, virginica_prior_prob)
  res <- t(ver_w[1:2]-vir_w[1:2])%*%x + (ver_w[3]-vir_w[3])
  return(res)
}
```

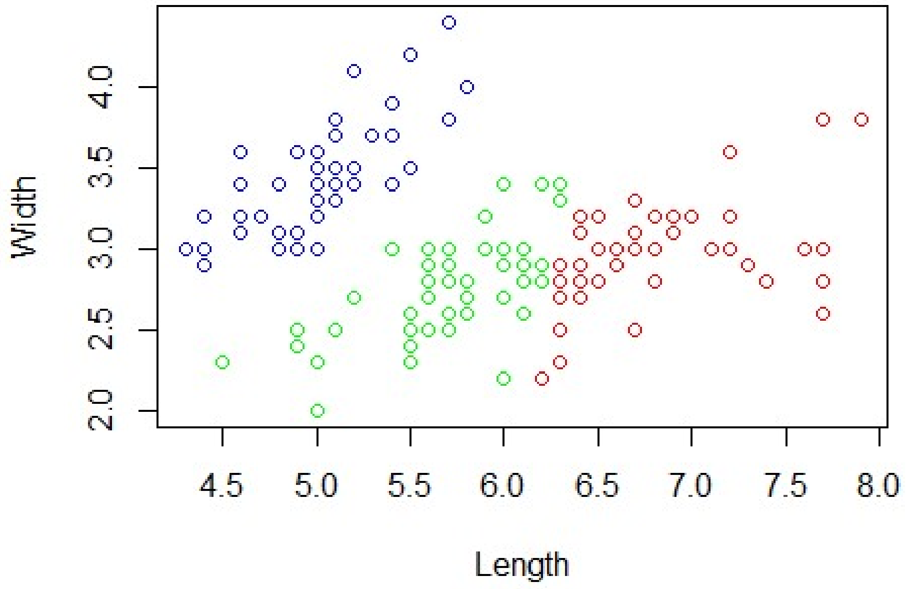
Between Setosa and Virginica:

```
set_vir_eq <- function(x){
  set_w <- weights(setosa_meanLW, setosa_prior_prob)
  vir_w <- weights(virginica_meanLW, virginica_prior_prob)
  res <- t(set_w[1:2]-vir_w[1:2])%*%x + (set_w[3]-vir_w[3])
  return(res)
}
```

The LDA assumption of the same covariance matrix for all classes does seem a little missplaced since the covariance matrices does seem to vary some.

Task 3

```
plot(iris$Sepal.Length, iris$Sepal.Width, xlab="Length", ylab="Width")
for (i in 1:length(y_hat)){
  color <- y_hat[i]
  points(iris$Sepal.Length[i], iris$Sepal.Width[i], col = ifelse(color=="seto
sa", "blue", ifelse(color=="versicolor", "green", ifelse(color=="virginica",
"red", "black"))))
}
```



The missclassification rate of the prediction and then the confusion matrix:

```
missclass_rate
```

```
## [1] 0.2

table(iris$Species, y_hat)

##             y_hat
##              setosa versicolor virginica
##   setosa         49          1         0
##   versicolor      0         36        14
##   virginica       0         15        35
```

I does seem high with 20% misscassification, but considering the distribution of data along the dimensions for Versicolor and Vitginica, it seems reasonable.

We then calculate a lda model and do predicition with lad() function from MASS package, on the same data as we used for our lda.

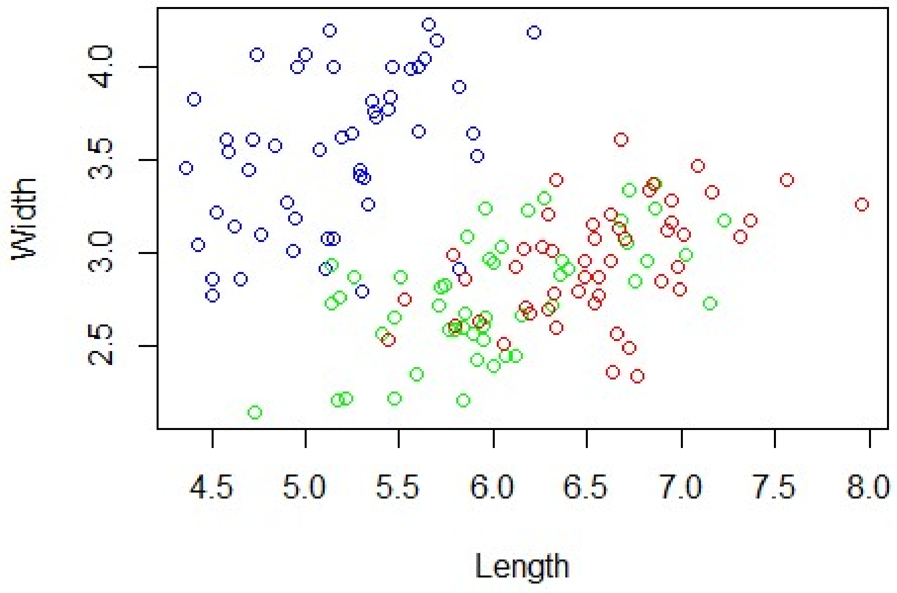
```
missclass_rate_true

## [1] 0.2

table(iris$Species, pred$class)

##
##              setosa versicolor virginica
##   setosa         49          1         0
##   versicolor      0         36        14
##   virginica       0         15        35
```

We get the same missclassification rate and the same confusion matrix for our prediction and the lda() from MASS package. Yes, it should be the same since we have made our lda presumably from the same assumtions. Which can also be seen when further investigateing the object obtained from lda() function from MASS package.

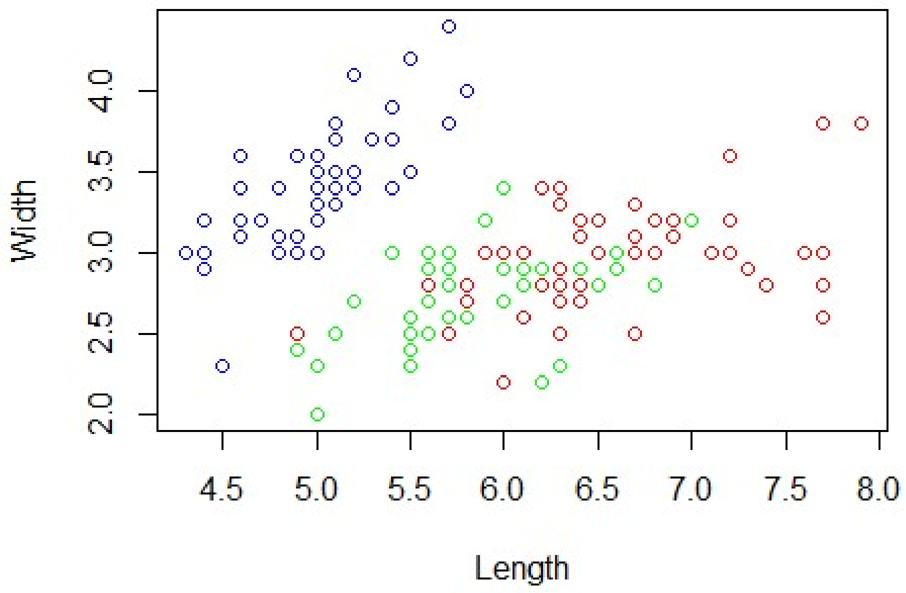
Task 4

We first generate new data based on the means for each class and with the pooled covariance matrix since that is the assumption of lda.

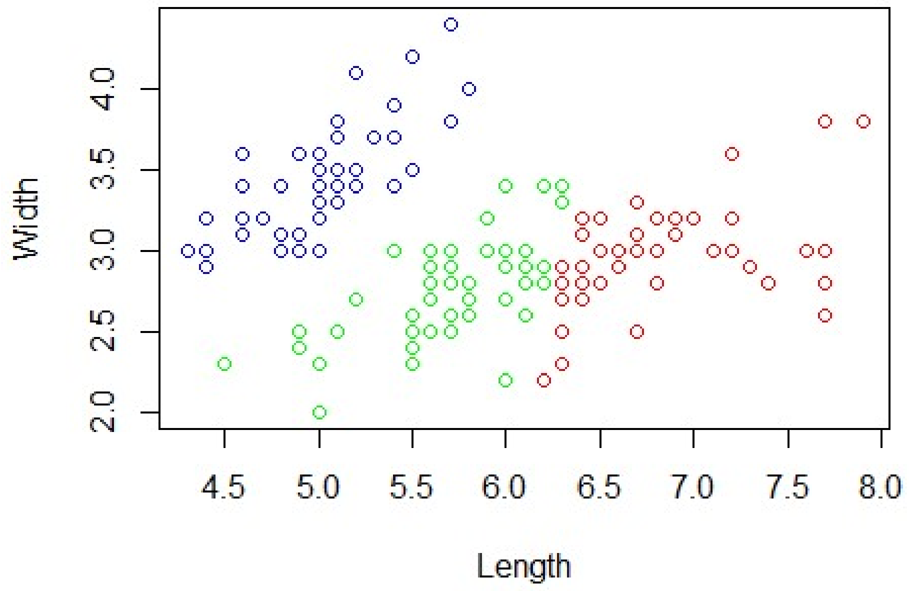
The plot with new data:



The plot with original data:



The plot with predictions on the original data:



The conclusions that can be made when looking at these plots is:

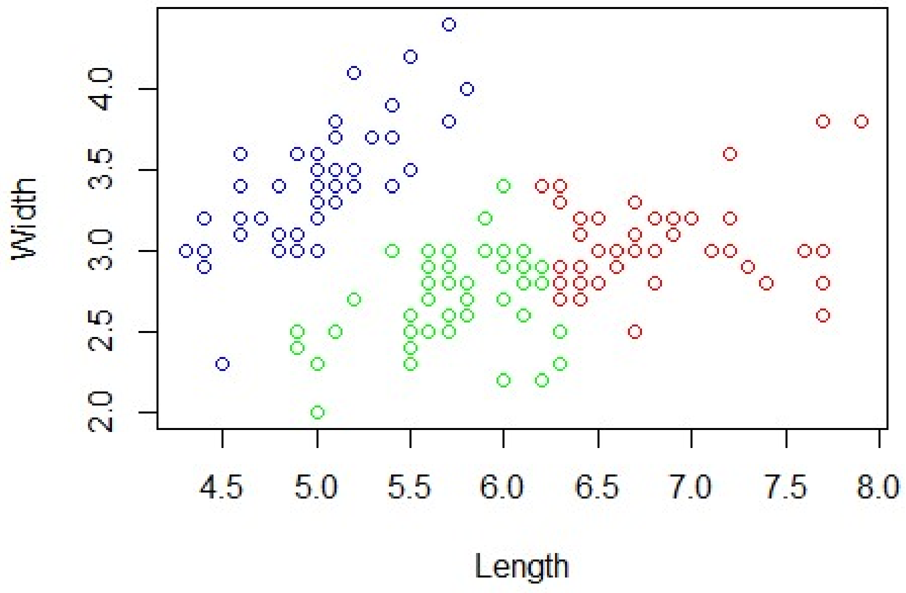
- The assumption of the classes having the same covariance matrix does seem to generate data that is not very similar to the true data. This in turn leads us to the conclusion of that that assumption is a bit heavy handed in this case.
- The predictions made seems to be doing a good job regarding Setosa(blue), as expected, but is having trouble keeping apart Versicolor(green) and Virginica(red).

Task 5

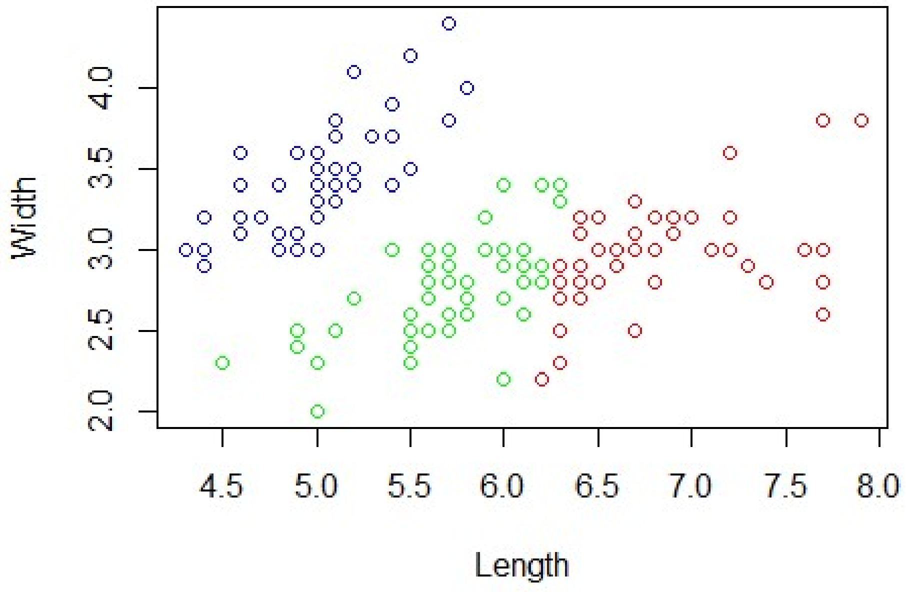
We make a logistic regression to compare the results of that and the lda.

First we plot the prediction for logistic regression:

```
plot(iris$Sepal.Length, iris$Sepal.Width, xlab = "Length", ylab = "Width")
for (i in 1:length(y_hat_log_reg)){
  color <- y_hat_log_reg[i]
  points(iris$Sepal.Length[i], iris$Sepal.Width[i], col = ifelse(color=="seto
sa", "blue", ifelse(color=="versicolor", "green", ifelse(color=="virginica",
"red", "black"))))
}
```



The the plot of our lda prediction.



The missclassification rate for logistic regression and the confusion matrix:

```
## [1] 0.1666667

table(iris$Species, y_hat_log_reg)

##              y_hat_log_reg
##               setosa versicolor virginica
##   setosa          50          0         0
##   versicolor       0         38        12
##   virginica        0         13        37
```

The missclassification rate for lda:

```
## [1] 0.2
```

As can be seen from both the plot, the missclassification rate and the confusion matrices it does seem that the logistic regression does a bett er job of classifying this data.

Lab 2 Assignment 2

Responsible: Daniel Bissesar

Task 1: Data needed to be converted to factors instead of chrs, otherwise standard stuff.

```
data <- read.csv2("C:/Users/Daniel/Desktop/bank-full.csv")
data[,2] = as.factor(data[,2])
data[,3] = as.factor(data[,3])
data[,4] = as.factor(data[,4])
data[,5] = as.factor(data[,5])
data[,7] = as.factor(data[,7])
data[,8] = as.factor(data[,8])
data[,9] = as.factor(data[,9])
data[,11] = as.factor(data[,11])
data[,16] = as.factor(data[,16])
data[,17] = as.factor(data[,17])

data = as.data.frame(data)
data = data[,-12]
library(tree)
library(naivebayes)
library(rpart)
library(rpart.plot)
library(caret)
library(e1071)

n=dim(data)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.4))
```

```
train=data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
valid=data[id2,]
id3=setdiff(id1,id2)
test=data[id3,]
```

Task 2: The missclassification rates were 0.1048, 0.1048 and 0.09362 respectively for the training data. For the validation data the missclassification rates were 0.1093, 1093 and 0.1112. This indicates that the best model is one from 2a or 2b since they return the same model.

```
fit=tree(y~., data=train)
plot(fit)
text(fit, pretty=0)
#fit
summary(fit)

#2.b
fit7000=tree(y~., data=train, minsize=7000)
plot(fit7000)
text(fit7000, pretty=0)
#fit7000
summary(fit7000)

#2.c
fitmindev=tree(y~., data=train, mindev=0.0005)
plot(fitmindev)
text(fitmindev, pretty=0)
#fitmindev
summary(fitmindev)
```

Task 3: The optimal model is with 21 leaves and gives a missclassification rate of about 11% which is still not great.

```
trainScore=rep(0,50)
valScore=rep(0,50)
for(i in 2:50) {
  prunedTree=prune.tree(fitmindev,best=i)
  pred=predict(prunedTree, newdata=valid,
               type="tree")
  trainScore[i]=deviance(prunedTree)
  valScore[i]=deviance(pred)
}
plot(2:50, trainScore[2:50], type="b", col="red",
     ylim=c(0,20000))
points(2:50, valScore[2:50], type="b", col="blue")

#validation score is best at 21 leaves, which variables are most important?
finalTree=prune.tree(fitmindev, best=21)
```

```
Yfit=predict(finalTree, newdata=valid,
             type="class")
confusionmatrix = data.matrix(table(valid$y,Yfit))
missclass = 1-sum(diag(confusionmatrix))/sum(confusionmatrix)
# missclass is about 11%, not a very good model
```

Task 4: This new model has missclassification rate of 11,7% which is worse than the previous model, the model now also always says no because of the more penalizing loss factor.

```
#2.4
l = c(0,1,5,0)
L = matrix(l, nrow = 2, ncol = 2)
treeclass = rpart(y~., data=train, method="class", parms=list(loss=L))
treeclassfit=predict(treeclass, newdata=test,
             type="class")
confusiontreeclass = data.matrix(table(test$y,treeclassfit))
missclasstreeclass = 1-sum(diag(confusiontreeclass))/sum(confusiontreeclass)
#lower missclassification rate but model now just always says no lol
```

Task 5: When plottin the ROC-curves we can see that in general the final fit has higher TPR values for lower FPR values which indicates a better model.

```
#2.5

fitNaive <- naiveBayes(formula = y~., data = train)
finalPredTest <- predict(finalTree, newdata = test, type = "vector")
naivePredTest <- predict(fitNaive, newdata = test, type = "raw")

sequence <- seq(from = 0,to = 0.95,by = 0.05)
TPRFinal <- rep(0,length(sequence))
TPRNaive <- rep(0,length(sequence))
FPRFinal <- rep(0,length(sequence))
FPRNaive <- rep(0,length(sequence))

for(i in 1:length(sequence)){
  TP1 = 0
  TP2 = 0
  FP1 = 0
  FP2 = 0
  for(j in 1:length(test$y)){
    if(finalPredTest[j,2] > sequence[i]) {
      if(test$y[j] == "yes"){
        TP1 <- TP1 + 1
      }
      else{
        FP1 <- FP1 + 1
      }
    }
    if(naivePredTest[j,2] > sequence[i]){
      if(test$y[j] == "yes"){
        TP2 <- TP2 + 1
      }
      else{
        FP2 <- FP2 + 1
      }
    }
  }

  TPRFinal[i] = TP1/sum(test$y == "yes")
  TPRNaive[i] = TP2/sum(test$y == "yes")
  FPRFinal[i] = FP1/sum(test$y == "no")
  FPRNaive[i] = FP2/sum(test$y == "no")
}
plot(FPRFinal, TPRFinal)
plot(FPRNaive, TPRNaive)
```

Lab 2 Assignment 3

Responsible: Filip Brunander

Task 1

```
xData <- csvData
xData$ViolentCrimesPerPop <- c()

centered.xData <- scale(xData, scale = FALSE)
centered.scaled.xData <- scale(xData)
covData <- cov(centered.scaled.xData)

eigenX <- eigen(covData)

print(sum(eigenX$values[1:34]))

## [1] 94.98273

print(sum(eigenX$values[1:35]))

## [1] 95.27018

#PC1:
print(eigenX$values[1])

## [1] 25.01699

#PC2:
print(eigenX$values[2])
```

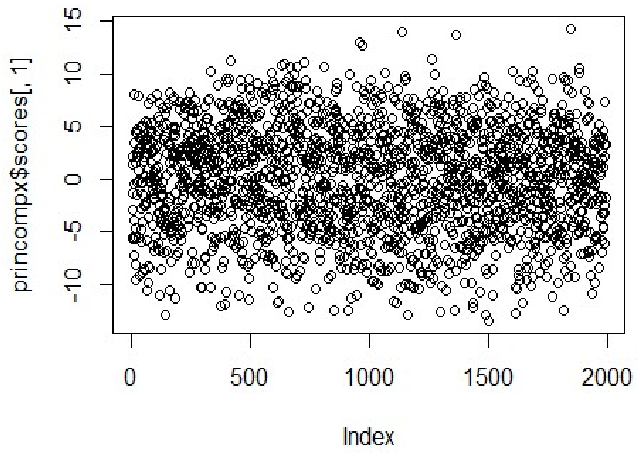
```
## [1] 16.93597
```

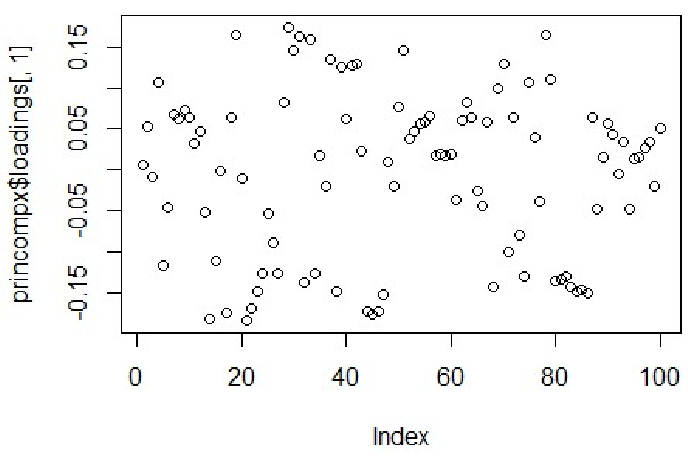
By printing the first 34 respective 35 feature we can see that 35 feature is where a variance of 95% is obtainable. Therefore, 35 or more features are needed to obtain at least 95% of variance.

We also print the first two principle components. Here we can see that the first principle component contributes with a variance of around 25% and the second around 17%.

Task 2

```
princompx <- princomp(xData, cor = TRUE, scores = TRUE)
plot(princompx$scores[,1])

plot(princompx$loadings[,1])
```





For the score plot the data is spread out quite a lot with no larger groups forming. The distribution could be normal. From the observations it seems like quite a lot of features have notable contribution to the first principle component. We also plotted the loadings to easier see how features affected the first principle component. There seems to be a few, on both the negative and positive part of the loadings that have notable contriobution to this component.

```
absScores <- abs(princompx$loadings[,1])
top5Scores <- tail(sort(unlist(absScores)), 5)

top5Scores

## PctPopUnderPov     pctWInvInc    PctKids2Par      medIncome      medFamInc
##      0.1737978      0.1748683      0.1755423      0.1819830      0.1833080
```

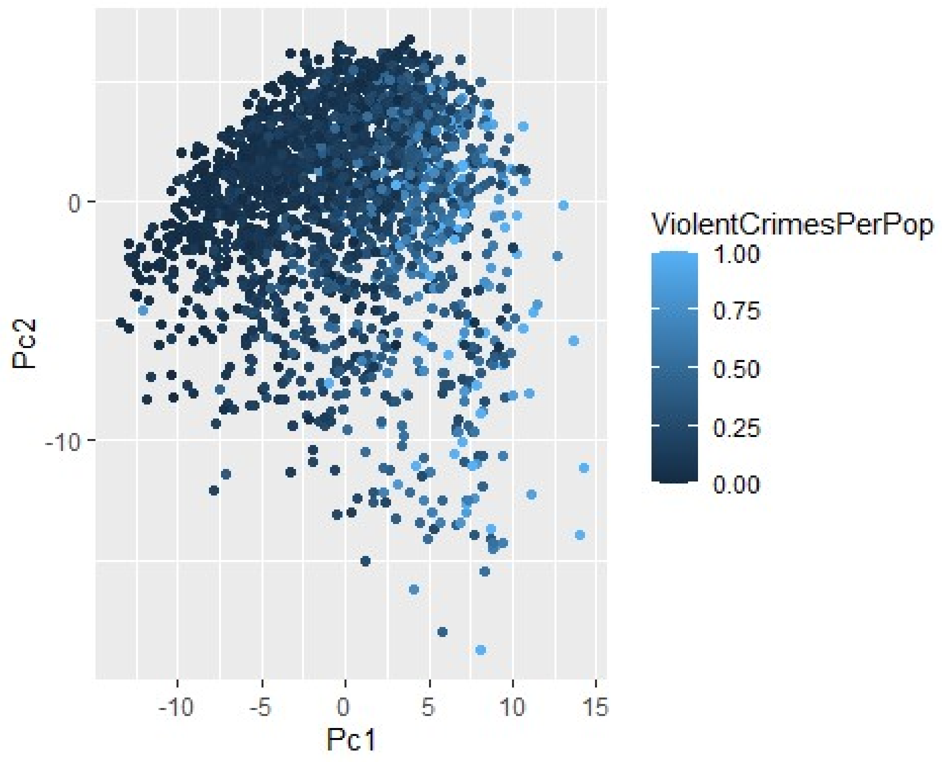
The five features that contribute the most is:

1. PctPopUnderPov (positive) - percentage of people under the poverty level.
2. pctWInvInc (negative) - percentage of households with investment / rent income in 1989.
3. PctKids2Par (negative) - percentage of kids in family housing with two parents.

4. medIncome (negative) - median household income.
5. medFamInc (negative) - median family income (differs from household income for non-family households).

We believe feature 1 has a logical relationship to increased crime levels. The same regarding feature 2 which instead is negative with more investments -> less crimes. Feature 3 is a bit less logical. Higher percentage of homes with two parents -> less crimes. We could see how this feature could affect other features and have an indirect affection to the crime level, but we do not see a direct logical connection between this feature and the target. Feature 4 and Feature 5 we do also see as rather logical with a higher median income/higher family income contribute to a lower crime level. We also believe that especially feature 1 has a logical connection to both 4 and 5. One could also argue that 2 and 3 seems to be connected as well.

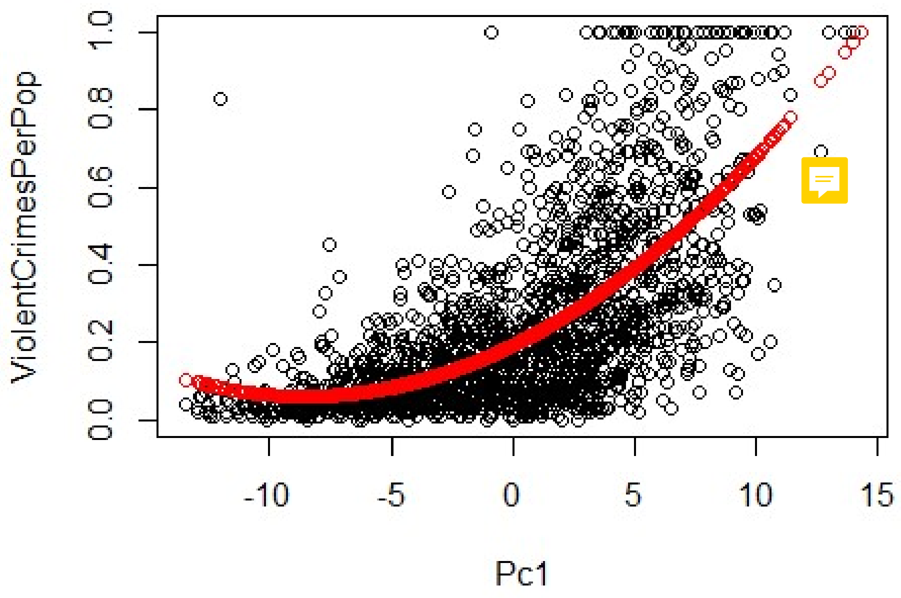
```
pcs <- data.frame(Pc1=princompx$scores[,1],Pc2=princompx$scores[,2])
pcsbind <- cbind(pcs, ViolentCrimesPerPop=csvData$ViolentCrimesPerPop)
ggplot(pcsbind, mapping = aes(x=Pc1,y=Pc2,colour=ViolentCrimesPerPop))+geom_p
oint()
```



Analysis of the ViolentCrimesPerPop plot. We can see how violent crimes increase when mainly the first principle component increases. The second principle component does not have the same level of influence it seems like. However, I do still observe some increase in violent crimes when the second principle decreases.

Task 3

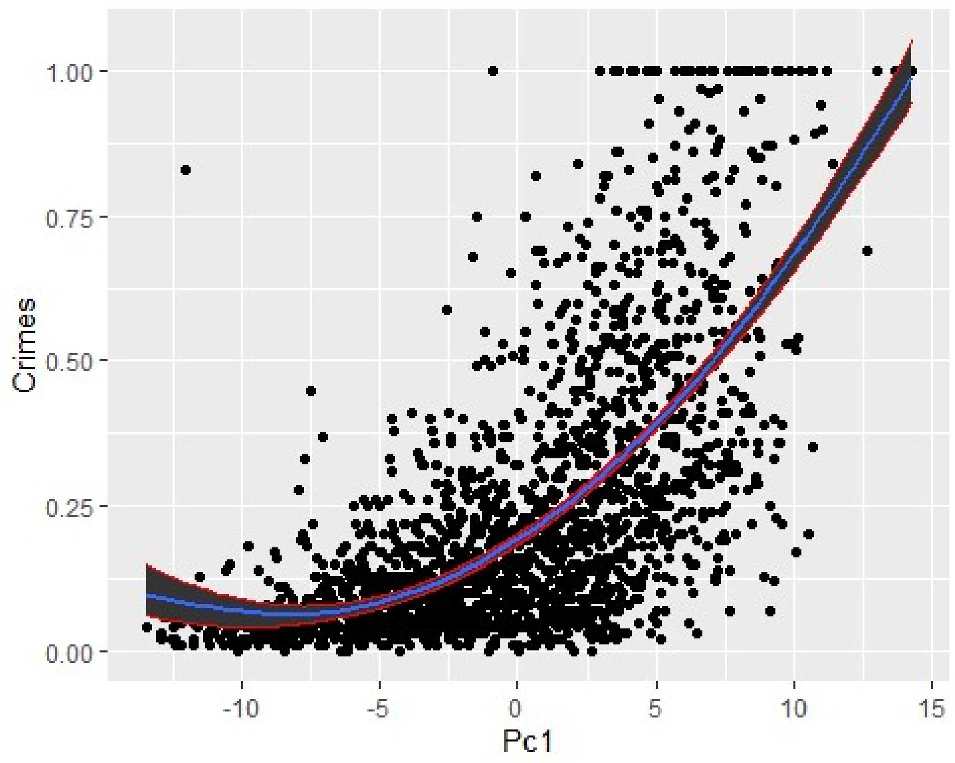
```
pcsbind3 <- pcsbind
pcsbind3[,2] <- c()
fit <- lm(formula = pcsbind3[,2]~poly(pcsbind3[,1], degree = 2), data=pcsbind
3)
fitPred <- predict(fit)

plot(pcsbind3, xvar=pcsbind3[,1],yvar=pcsbind3[,2])

points(x=pcsbind3[,1], y=fitPred, col="red")
```



Analysis of the crimes against Pc1 plot. To a beginning, the model seem to predict a bit high due to degree = 2 of the polynomial model. The model prediction on the target increases while the actual target remains low. From around Pc1 = -10/-5 to 5 the model seem to predict rather close to the center of the distribution which is a good sign. However, when increasing Pc1>5 we see how the prediction curve is not steep enough to follow the increase in the target. This is a sign that the polynomial model of degree = 2 is not able to properly follow the target data on both ends of the Pc1 scale. Partly this model seems to capture the connection between target and feature but it is lacking a bit towards the ends of the pc1 scale especially. But since most of data is centered around -10 to +5 on the Pc1 axis the target seems to atleast be decently explained by the feature.

Task 4

```
mle <- fit
data2 <- data.frame(Crimes=pcsbind3[,2], Pc1=pcsbind3[,1])

rng = function(data, mle) {
  data1=data.frame(Crimes=data$Crimes, Pc1=data$Pc1)
  n=length(data$Crimes)
  data1$Crimes=rnorm(n,predict(mle, newdata = data1), sd(mle$residuals))
  #data1$Crimes = runif(n, min = 0, max = 1)
  return(data1)
}

#Confidence band.
f1=function(data1){
  res <- lm(formula = Crimes~poly(Pc1, degree = 2), data = data1)
  crimePred <- predict(res,newdata=data2)
  return(crimePred)
}

set.seed(12345)
res <- boot(data=data2, statistic=f1, R=1000, mle=mle, ran.gen = rng, sim="pa
rametric")
e <- envelope(res, level = 0.95)

fitT4=lm(Crimes~poly(Pc1, degree = 2), data=data2)
crimesP = predict(fitT4)

ggplot(data2, aes(x=Pc1,y=Crimes))+
  geom_point()+
  geom_ribbon(aes(ymin=e$point[2,], ymax=e$point[1,]), colour="red")+
  geom_smooth( mapping = aes(x=Pc1, y=crimesP))

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

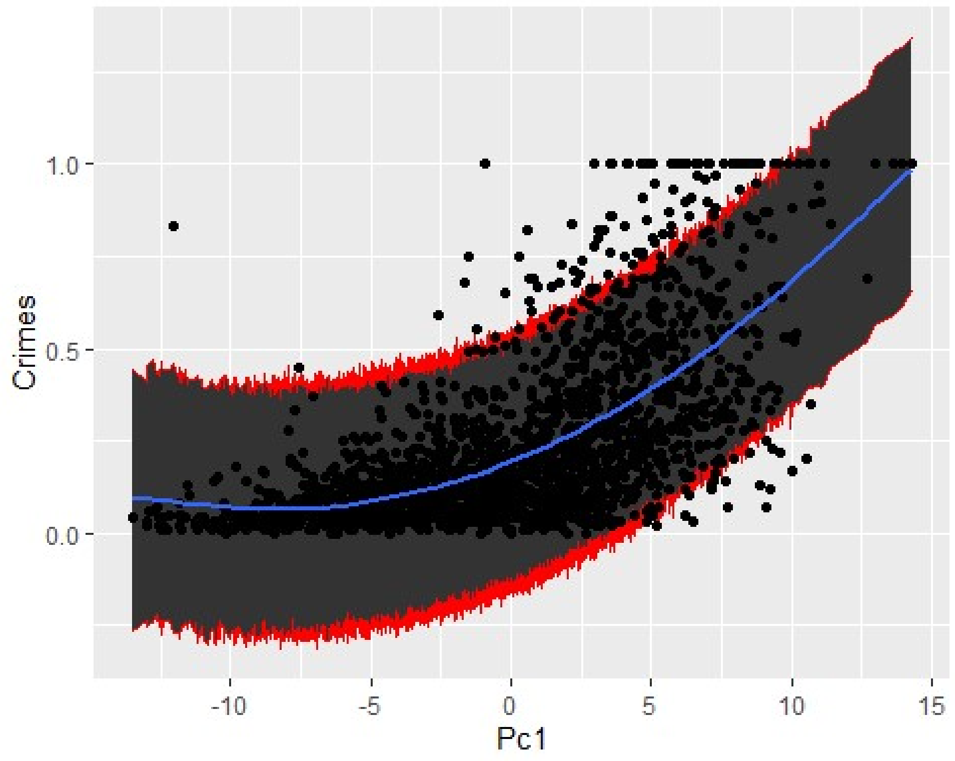


Task 4.a

We can in the graph with the confidence interval see that the majority of points is outside of it. This is not uncommon with large datasets. Where we already have many datapoints that we could use in our fitting of the model, we can see that the confidence interval is more narrow for those compared to the points at the ends of the Pc1-axis. This shows us that the prediction uncertainty is larger at those parts of the graph/model.

```
#Prediction band.
f2=function(data1) {
  res <- lm(formula = Crimes~poly(Pc1, degree = 2), data = data1)
  crimePred <- predict(res,newdata=data2)
  n=length(data1$Crimes)
  predictedC=rnorm(n, crimePred,sd(mle$residuals))
  return(predictedC)
}

res2=boot(data2, statistic = f2, R=1000, mle=mle, ran.gen = rng, sim="paramet
ric")
e2 <- envelope(res2, level = 0.95)

## Warning in envelope(res2, level = 0.95): unable to achieve requested overa
ll
## error rate
```

```r
ggplot(data2, aes(x=Pc1,y=Crimes))+
  geom_ribbon(aes(ymin=e2$point[2,], ymax=e2$point[1,]), colour="red")+
  geom_point()+
  geom_smooth( mapping = aes(x=Pc1, y=crimesP))

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Task 4.b

We can in this graph see that the prediction interval is much larger than the confidence interval, which is expected due to predicting a single point is more uncertain than the mean response. The prediction interval shall represent the uncertainty of the new datapoint (predicted value), so we are 95% sure that the next prediction will fall in our prediction interval. In our case, the uncertainty is large throughout all values for Pc1, even though the data given is much less concentrated parts of the Pc1 axis.

Appendix

Assignment 1 code:

```
#Assignment 1

library(datasets)

library(dplyr)

library(MASS)

library("nnet")

library("mvtnorm")

library("OceanView")


data(iris)

#summary(iris)


setosa <- filter(iris, Species == 'setosa')

versicolor <- filter(iris, Species == 'versicolor')

virginica <- filter(iris, Species == 'virginica')


plot(iris$Sepal.Length, iris$Sepal.Width)

points(setosa$Sepal.Length, setosa$Sepal.Width, col = "blue")

points(versicolor$Sepal.Length, versicolor$Sepal.Width,  col = "green")

points( virginica$Sepal.Length, virginica$Sepal.Width, col = "red")


prior_prob <- function(x){

  return(nrow(x)/nrow(iris))

}


ClassMean <- function(x){
  return(c(mean(x$Sepal.Length), mean(x$Sepal.Width)))
}


cov_matrix <- function(x){
  mat <- cbind(x$Sepal.Length, x$Sepal.Width)
  return(cov(mat, method = "pearson"))
}
#prior probabilities
setosa_prior_prob <- prior_prob(setosa)
versicolor_prior_prob <- prior_prob(versicolor)
virginica_prior_prob <- prior_prob(virginica)

#means per class and varibale
setosa_meanLW <- ClassMean(setosa)
versicolor_meanLW <- ClassMean(versicolor)
virginica_meanLW <- ClassMean(virginica)

#covariance matrix
setosa_cov <- cov_matrix(setosa)
versicolor_cov <- cov_matrix(versicolor)
virginica_cov <- cov_matrix(virginica)

#pooled cov_matrix
pooled_cov <- (setosa_cov+versicolor_cov+virginica_cov)/3

#discriminative functions:

weights <- function(meanLW,prob){
  wix <- solve(pooled_cov)%*%meanLW
  w0 <-  (-0.5)*t(meanLW)%*%solve(pooled_cov)%*%meanLW + log(prob)
  res <- c(wix, w0)
  return(res)
}

setosa_disc_fun <- function(x){
  w <- weights(setosa_meanLW, setosa_prior_prob)
  res <- x%*%c(w[1],w[2]) + w[3]
  return(res)
}

versicolor_disc_fun <- function(x){
  w <- weights(versicolor_meanLW, versicolor_prior_prob)
  res <- x%*%c(w[1], w[2]) + w[3]
  return(res)
}

virginica_disc_fun <- function(x){
  w <- weights(virginica_meanLW,virginica_prior_prob)
  res <- x%*%c(w[1], w[2]) + w[3]
  return(res)
}


set_ver_eq <- function(x){
  set_w <- weights(setosa_meanLW, setosa_prior_prob)
  ver_w <- weights(versicolor_meanLW, versicolor_prior_prob)
  res <- t(set_w[1:2]-ver_w[1:2])%*%x + (set_w[3]-ver_w[3])
  return(res)
```

```
}

set_vir_eq <- function(x){

  set_w <- weights(setosa_meanLW, setosa_prior_prob)

  vir_w <- weights(virginica_meanLW, virginica_prior_prob)

  res <- t(set_w[1:2]-vir_w[1:2])%*%x + (set_w[3]-vir_w[3])

  return(res)

}

ver_vir_eq <- function(x){

  ver_w <- weights(versicolor_meanLW, versicolor_prior_prob)

  vir_w <- weights(virginica_meanLW, virginica_prior_prob)

  res <- t(ver_w[1:2]-vir_w[1:2])%*%x + (ver_w[3]-vir_w[3])

  return(res)

}


#3


my_predict <- function(x){

  se <- setosa_disc_fun(x)

  ve <- versicolor_disc_fun(x)

  vi <- virginica_disc_fun(x)

  disc_mat <- cbind(se,ve,vi)

  y_hat <- rep("???", dim(disc_mat)[1])

  for (i in 1:dim(disc_mat)[1]){

    pred <- which.max(disc_mat[i,])

    if (pred == 1){

      y_hat[i] <- "setosa"

    } else if (pred == 2){

      y_hat[i] <- "versicolor"

    } else if (pred == 3){
      y_hat[i] <- "virginica"
    }
  }
  return(y_hat)
}

y_hat <- my_predict(cbind(iris$Sepal.Length, iris$Sepal.Width))

plot(iris$Sepal.Length, iris$Sepal.Width)

for (i in 1:length(y_hat)){
  color <- y_hat[i]
  points(iris$Sepal.Length[i], iris$Sepal.Width[i], col = ifelse(color=="setosa", "blue",
ifelse(color=="versicolor", "green", ifelse(color=="virginica", "red", "black"))))
}

missclass_rate_fun <- function(y, y_hat){
  wrong <- 0
  for (i in 1:length(y)){
    if (y[i]!=y_hat[i]){
      wrong <- wrong + 1
    }
  }
  res <- wrong/length(y)
  return(res)
}

missclass_rate <- missclass_rate_fun(iris$Species, y_hat)
table(iris$Species, y_hat)
```

```
lda_fit <- lda(Species~Sepal.Length + Sepal.Width, iris)
pred <- predict(lda_fit, iris)
table(iris$Species, pred$class)

#4
library("mvtnorm")




#the pooled covariance
set.seed(12345)
setosa_new <- rmvnorm(50, mean = setosa_meanLW, sigma = pooled_cov)
#set.seed(12345)
versicolor_new <- rmvnorm(50, mean = versicolor_meanLW, sigma = pooled_cov)
#set.seed(12345)
virginica_new <- rmvnorm(50, mean = virginica_meanLW, sigma = pooled_cov)
iris_new <- rbind(setosa_new, versicolor_new, virginica_new)

plot(iris_new)
points(setosa_new[,1], setosa_new[,2], col = "blue")
points(versicolor_new[,1], versicolor_new[,2], col = "green")
points(virginica_new[,1], virginica_new[,2], col = "red")

#5
library("nnet")
log_reg_model <- multinom(Species~Sepal.Length + Sepal.Width, iris, model = TRUE)

y_hat_log_reg <- rep("???", dim(log_reg_model$fitted.values)[1])
```

```
for (i in 1:dim(log_reg_model$fitted.values)[1]){
  pred <- which.max(log_reg_model$fitted.values[i,])
  if (pred == 1){
    y_hat_log_reg[i] <- "setosa"
  } else if (pred == 2){
    y_hat_log_reg[i] <- "versicolor"
  } else if (pred == 3){
    y_hat_log_reg[i] <- "virginica"
  }
}


#plot log regression
plot(iris$Sepal.Length, iris$Sepal.Width)
for (i in 1:length(y_hat_log_reg)){
  color <- y_hat_log_reg[i]
  points(iris$Sepal.Length[i], iris$Sepal.Width[i], col = ifelse(color=="setosa", "blue",
ifelse(color=="versicolor", "green", ifelse(color=="virginica", "red", "black"))))
}


#plot lda
plot(iris$Sepal.Length, iris$Sepal.Width)
for (i in 1:length(y_hat)){
  color <- y_hat[i]
  points(iris$Sepal.Length[i], iris$Sepal.Width[i], col = ifelse(color=="setosa", "blue",
ifelse(color=="versicolor", "green", ifelse(color=="virginica", "red", "black"))))
}


log_reg_missclass_rate <- missclass_rate_fun(iris$Species, y_hat_log_reg)
```

Assignment 2 code:

```
data <- read.csv2("C:/Users/Daniel/Desktop/bank-full.csv")

data[,2] = as.factor(data[,2])

data[,3] = as.factor(data[,3])

data[,4] = as.factor(data[,4])

data[,5] = as.factor(data[,5])

data[,7] = as.factor(data[,7])

data[,8] = as.factor(data[,8])

data[,9] = as.factor(data[,9])

data[,11] = as.factor(data[,11])

data[,16] = as.factor(data[,16])

data[,17] = as.factor(data[,17])







data = as.data.frame(data)

data = data[,-12]

library(tree)

library(naivebayes)

library(rpart)

library(rpart.plot)

library(caret)

library(e1071)


n=dim(data)[1]
```

```
set.seed(12345)
id=sample(1:n, floor(n*0.4))
train=data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.3))
valid=data[id2,]
id3=setdiff(id1,id2)
test=data[id3,]

#2.a
fit=tree(y~., data=train)
plot(fit)
text(fit, pretty=0)
summary(fit)
Yfitoriginal=predict(fit, newdata=valid,
        type="class")
confusionmatrixoriginal = data.matrix(table(valid$y,Yfitoriginal))
missclassoriginal = 1-sum(diag(confusionmatrixoriginal))/sum(confusionmatrixoriginal)

#2.b
fit7000=tree(y~., data=train, minsize=7000)
plot(fit7000)
text(fit7000, pretty=0)
summary(fit7000)
Yfit7000=predict(fit7000, newdata=valid,
        type="class")
confusionmatrix7000 = data.matrix(table(valid$y,Yfit7000))
missclass7000 = 1-sum(diag(confusionmatrix7000))/sum(confusionmatrix7000)
```

```
#2.c

fitmindev=tree(y~., data=train, mindev=0.0005)

plot(fitmindev)

text(fitmindev, pretty=0)

summary(fitmindev)

Yfitmindev=predict(fitmindev, newdata=valid,

            type="class")

confusionmatrixmindev = data.matrix(table(valid$y,Yfitmindev))

missclassmindev = 1-sum(diag(confusionmatrixmindev))/sum(confusionmatrixmindev)



trainScore=rep(0,50)

valScore=rep(0,50)

for(i in 2:50) {

  prunedTree=prune.tree(fitmindev,best=i)

  pred=predict(prunedTree, newdata=valid,

             type="tree")

  trainScore[i]=deviance(prunedTree)

  valScore[i]=deviance(pred)

}

plot(2:50, trainScore[2:50], type="b", col="red",

     ylim=c(0,20000))

points(2:50, valScore[2:50], type="b", col="blue")


#validation score is best at 21 leaves, which variables are most important?

finalTree=prune.tree(fitmindev, best=21)

Yfit=predict(finalTree, newdata=valid,

            type="class")
```

```
confusionmatrix = data.matrix(table(valid$y,Yfit))

missclass = 1-sum(diag(confusionmatrix))/sum(confusionmatrix)

# missclass is about 11%, not a very good model


#2.4

l = c(0,1,5,0)

L = matrix(l, nrow = 2, ncol = 2)

treeclass = rpart(y~., data=train, method="class", parms=list(loss=L))

treeclassfit=predict(treeclass, newdata=test,

        type="class")

confusiontreeclass = data.matrix(table(test$y,treeclassfit))

missclasstreeclass = 1-sum(diag(confusiontreeclass))/sum(confusiontreeclass)

#lower missclassification rate but model now just always says no lol


#2.5


fitNaive <- naiveBayes(formula = y~., data = train)

finalPredTest <- predict(finalTree, newdata = test, type = "vector")

naivePredTest <- predict(fitNaive, newdata = test, type = "raw")


sequence <- seq(from = 0,to = 0.95,by = 0.05)

TPRFinal <- rep(0,length(sequence))

TPRNaive <- rep(0,length(sequence))

FPRFinal <- rep(0,length(sequence))

FPRNaive <- rep(0,length(sequence))


for(i in 1:length(sequence)){

  TP1 = 0

  TP2 = 0
  FP1 = 0
  FP2 = 0
  for(j in 1:length(test$y)){
    if(finalPredTest[j,2] > sequence[i]) {
      if(test$y[j] == "yes"){
        TP1 <- TP1 + 1
      }
      else{
        FP1 <- FP1 + 1
      }
    }
    if(naivePredTest[j,2] > sequence[i]){
      if(test$y[j] == "yes"){
        TP2 <- TP2 + 1
      }
      else{
        FP2 <- FP2 + 1
      }
    }
  }

  TPRFinal[i] = TP1/sum(test$y == "yes")
  TPRNaive[i] = TP2/sum(test$y == "yes")
  FPRFinal[i] = FP1/sum(test$y == "no")
  FPRNaive[i] = FP2/sum(test$y == "no")
}
plot(FPRFinal, TPRFinal)
plot(FPRNaive, TPRNaive)
```

Assignment 3 code:

```
setwd("~/Programming/TDDE01")


library(ggplot2)

library(boot)


csvData <- read.csv("communities.csv", header=TRUE)


#Task 1

xData <- csvData

xData$ViolentCrimesPerPop <- c()


centered.xData <- scale(xData, scale = FALSE)

centered.scaled.xData <- scale(xData)

covData <- cov(centered.scaled.xData)


eigenX <- eigen(covData)


print(sum(eigenX$values[1:34]))

print(sum(eigenX$values[1:35]))

#=> 35 features are required to obtain at least 95% of variance in the data.


#PC1:

print(eigenX$values[1])

#=> 25%


#PC2:
```

```
print(eigenX$values[2])

#=> 17%


#Task 2

princompx <- princomp(xData, cor = TRUE, scores = TRUE)

#First principle component

plot(princompx$scores[,1])

plot(princompx$loadings[,1])

absScores <- abs(princompx$loadings[,1])

top5Scores <- tail(sort(unlist(absScores)), 5)

absScores

top5Scores


pcs <- data.frame(Pc1=princompx$scores[,1],Pc2=princompx$scores[,2])

pcsbind <- cbind(pcs, ViolentCrimesPerPop=csvData$ViolentCrimesPerPop)

ggplot(pcsbind, mapping = aes(x=Pc1,y=Pc2,colour=ViolentCrimesPerPop))+geom_point()


#Task 3

pcsbind3 <- pcsbind

pcsbind3[,2] <- c()

fit <- lm(formula = pcsbind3[,2]~poly(pcsbind3[,1], degree = 2), data=pcsbind3)

fitPred <- predict(fit)


plot(pcsbind3, xvar=pcsbind3[,1],yvar=pcsbind3[,2])

points(x=pcsbind3[,1], y=fitPred, col="red")


#Task 4

mle <- fit

data2 <- data.frame(Crimes=pcsbind3[,2], Pc1=pcsbind3[,1])
```

```
rng = function(data, mle) {
  data1=data.frame(Crimes=data$Crimes, Pc1=data$Pc1)
  n=length(data$Crimes)
  data1$Crimes=rnorm(n,predict(mle, newdata = data1), sd(mle$residuals))
  #data1$Crimes = runif(n, min = 0, max = 1)
  return(data1)
}

#Confidence band.
f1=function(data1){
  res <- lm(formula = Crimes~poly(Pc1, degree = 2), data = data1)
  crimePred <- predict(res,newdata=data2)
  return(crimePred)
}

set.seed(12345)
res <- boot(data=data2, statistic=f1, R=1000, mle=mle, ran.gen = rng, sim="parametric")
e <- envelope(res, level = 0.95)

fitT4=lm(Crimes~poly(Pc1, degree = 2), data=data2)
crimesP = predict(fitT4)



ggplot(data2, aes(x=Pc1,y=Crimes))+
  geom_point()+
  geom_ribbon(aes(ymin=e$point[2,], ymax=e$point[1,]), colour="red")+
  geom_smooth( mapping = aes(x=Pc1, y=crimesP))
```

```
#Prediction band.

f2=function(data1) {

  res <- lm(formula = Crimes~poly(Pc1, degree = 2), data = data1)

  crimePred <- predict(res,newdata=data2)

  n=length(data1$Crimes)

  predictedC=rnorm(n, crimePred,sd(mle$residuals))

  return(predictedC)

}


res2=boot(data2, statistic = f2, R=1000, mle=mle, ran.gen = rng, sim="parametric")

e2 <- envelope(res2, level = 0.95)



ggplot(data2, aes(x=Pc1,y=Crimes))+

  geom_ribbon(aes(ymin=e2$point[2,], ymax=e2$point[1,]), colour="red")+

  geom_point()+

  geom_smooth( mapping = aes(x=Pc1, y=crimesP))
```