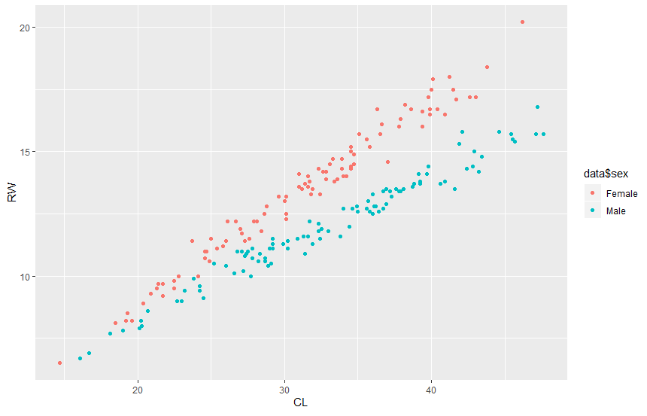
Vikgu708

Lab2 TDDE01

Assignment 1 – LDA and Logistic Regression

Excercise 1.

```
ggplot(data,aes(x=data$CL,y=data$RW,col=data$sex)) + geom_point() + labs("Sex by RW&CL",x="CL",y="RW")
```



As the distributions are not clearly overlapping, an LDA method will be well suited to classify this data as the a decision boundary can clearly split the two data sets.

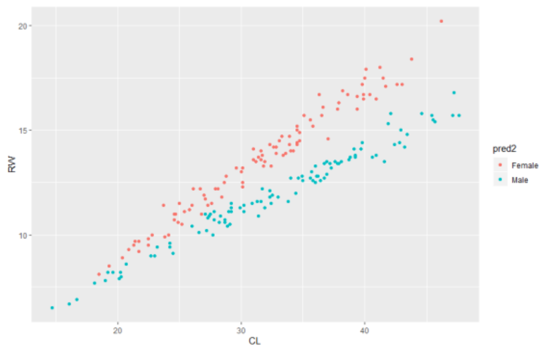
Excercise 2.

```
lda2 = lda(sex ~ RW + CL, data = data )
pred2 = predict(lda2,data)$class
table2 = table(pred2,data$sex)
mcr2 = mcr(table2)
```

```
> table2

pred2     Female Male
  Female      97    4
  Male         3   96
> mcr2
[1] 0.035
```

With only a 3.5% misclassification rate, it seems that this model is very well suited to classify the data.

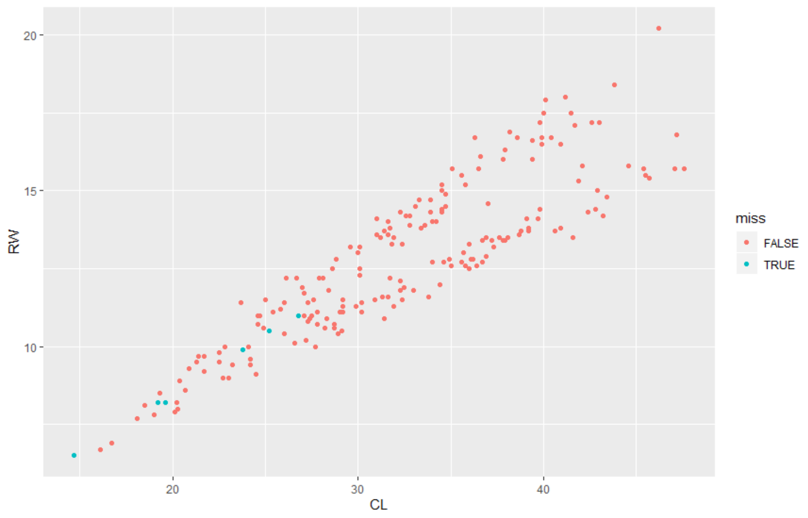
```
ggplot(data,aes(x=data$CL,y=data$RW,col=pred2)) + geom_point() + labs("LDA prediction",x="CL",y="RW")
```



The above chart is very similar to the original labels, which corresponds to the low classification error.

The misclassifications are found at clusters where data points of both labels are near each other's. The following plot shows the miss classifications colored in blue.

```
miss = !data$sex==pred2
ggplot(data,aes(x=data$CL,y=data$RW,col=miss)) + geom_point() + labs("LDA prediction",x="CL",y="RW")
```



Excercise 3.

```
lda_prior = lda(sex ~ RW + CL, data = data, prior=c(0.1,0.9))
pred_prior = predict(lda_prior,data)$class
table_prior = table(pred_prior,data$sex)
mcr_prior = mcr(table_prior)
```

```
> table_prior

pred_prior Female Male
    Female     84    0
    Male       16  100
> mcr_prior
[1] 0.08
```

A prior that increases the prior likelihood of being a male (suggesting an overrepresentation of males in the population) will naturally produce a result where more observations are classified as male, which can be seen in the confusion table where 116 observations were classified as male compared to the last result without prior where 99 were classified as male.

Even with a very large prior distribution for male, the mcr is still quite small, at 8%. The amount of misclassified points has increased from 7 to 16. The only slight increase can be attributed to the high separation between the data points in the training data, which results in LDA approximating quite separated distributions for the two classes.

Exercise 4.

```
Y = as.numeric(data$sex)-1
threshold = 0.5
logistic = glm(Y ~ RW + CL, data = data)
pred_glm = predict(logistic,data)
prediction = pred_glm
prediction[which(prediction>threshold)] = "Male"
prediction[which(prediction<=threshold)] = "Female"
```

```
table_glm = table(as.numeric(pred_glm>threshold),data$sex)
```

Vikgu708

```
> table_glm

   Female Male
 0     97    4
 1      3   96
```

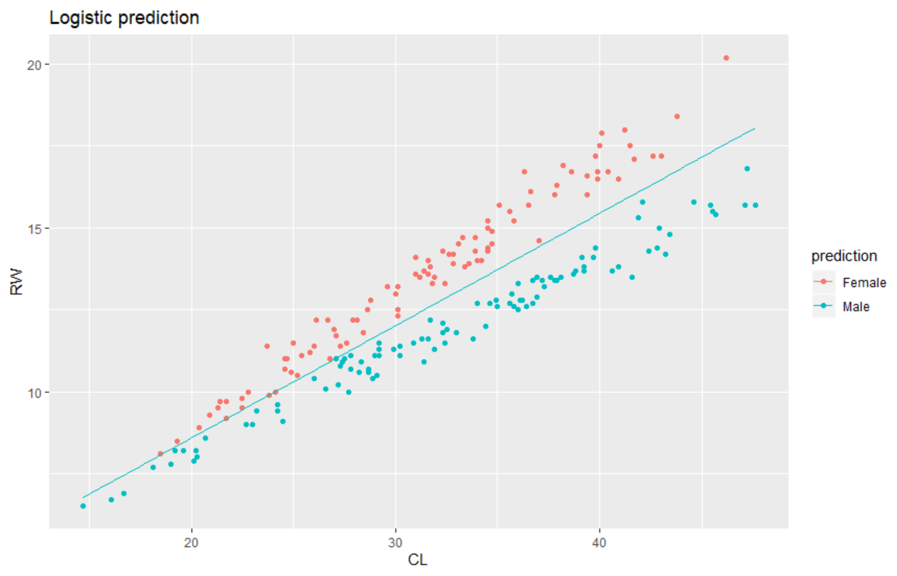
The misclassification result is the same as the LDA method.

As the decision boundary is the point the regression where both classifications have the same probability, the decision boundary equation is found from solving for RW in the equation where p=0,5.

$$p = w_1 + w_2 * RW + w_3 * CL$$

As CL is the x coordinate in our plot.

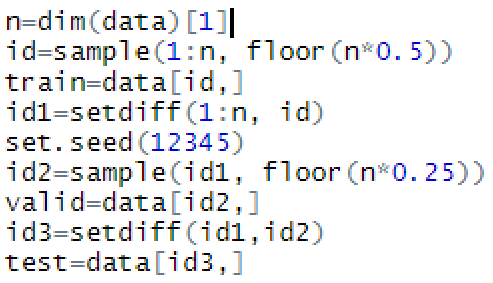
```
w = logistic$coefficients
decision = function(x)  (threshold-w[3]*x - w[1])/w[2]
glmplot + stat_function(fun=decision)
```



Vikgu708

Assignment 2

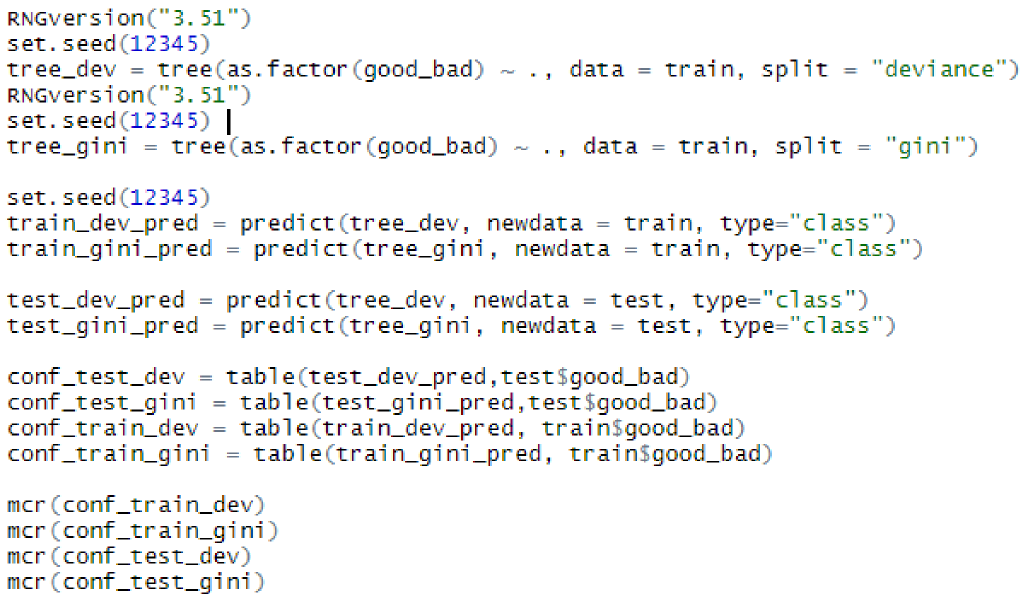
Exercise 1



```
n=dim(data)[1]
id=sample(1:n, floor(n*0.5))
train=data[id,]
id1=setdiff(1:n, id)
set.seed(12345)
id2=sample(id1, floor(n*0.25))
valid=data[id2,]
id3=setdiff(id1,id2)
test=data[id3,]
```

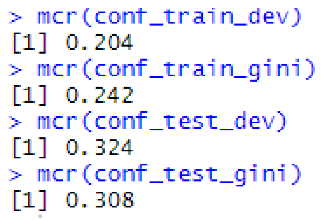
This code splits the data into 50% training, 25% test and 25% validation.

Exercise 2



```
RNGversion("3.51")
set.seed(12345)
tree_dev = tree(as.factor(good_bad) ~ ., data = train, split = "deviance")
RNGversion("3.51")
set.seed(12345)
tree_gini = tree(as.factor(good_bad) ~ ., data = train, split = "gini")

set.seed(12345)
train_dev_pred = predict(tree_dev, newdata = train, type="class")
train_gini_pred = predict(tree_gini, newdata = train, type="class")

test_dev_pred = predict(tree_dev, newdata = test, type="class")
test_gini_pred = predict(tree_gini, newdata = test, type="class")

conf_test_dev = table(test_dev_pred,test$good_bad)
conf_test_gini = table(test_gini_pred,test$good_bad)
conf_train_dev = table(train_dev_pred, train$good_bad)
conf_train_gini = table(train_gini_pred, train$good_bad)

mcr(conf_train_dev)
mcr(conf_train_gini)
mcr(conf_test_dev)
mcr(conf_test_gini)
```

This code predicts test and training data on tree models based on deviance and gini.



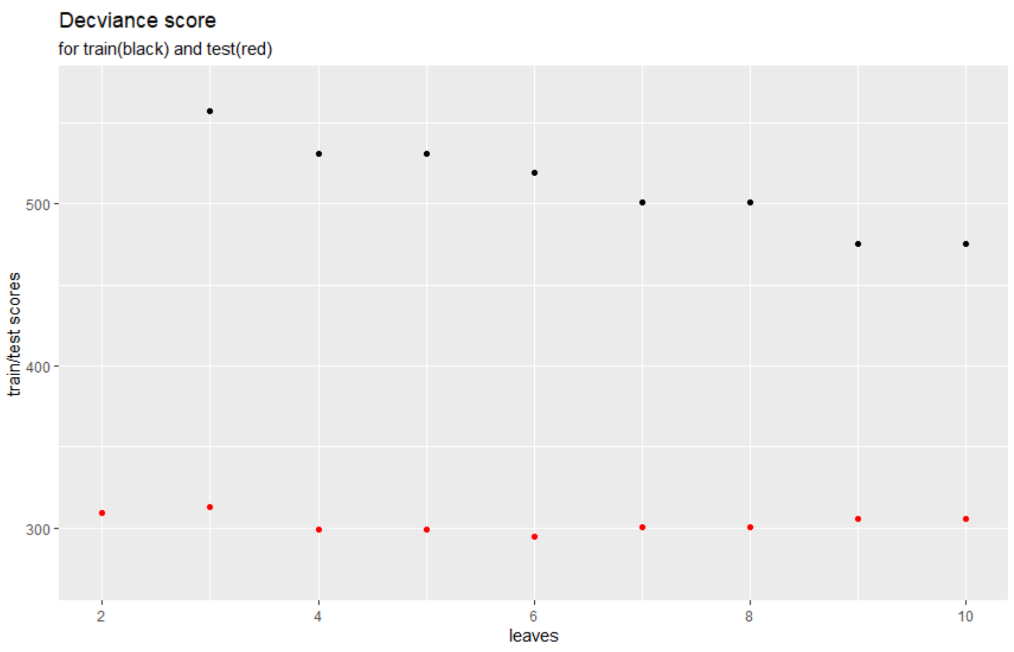
```
> mcr(conf_train_dev)
[1] 0.204
> mcr(conf_train_gini)
[1] 0.242
> mcr(conf_test_dev)
[1] 0.324
> mcr(conf_test_gini)
[1] 0.308
```

As we can see the deviance model is the superior one based on mcr.

Vikgu708

Exercise 3

```
leaves = 10

trainScore=rep(0,leaves)
testScore=rep(0,leaves)


for(i in 2:leaves) {
  RNGversion("3.51")
  set.seed(12345)
  prunedTree = prune.tree(tree_dev, best = i)
  pred = predict(prunedTree, newdata=valid, type="tree")
  trainScore[i] = deviance(prunedTree)
  testScore[i] = deviance(pred)
}

qplot(x=2:leaves,y= trainScore[2:leaves], ylim=c(270,570), ylab="train/test scores", xlab="leaves") +
```

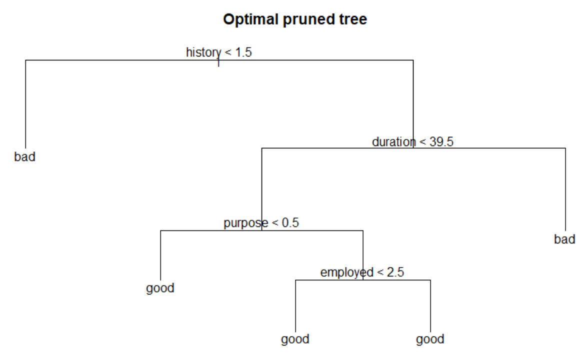


From the graph, the point where test error is smallest at 4 leaves.

```
optTree = prune.tree(tree_dev, best = 4)
summary(optTree)
```

```
Misclassification error rate: 0.2525 = 125 / 495
```

```
plot(optTree)
text(optTree)
title("Optimal pruned tree")
```



The above graph shows us a **depth of 4.**

```
Variables actually used in tree construction:
[1] "history"  "duration" "purpose"  "employed"
```

A history less than 1.5 years immediately gives a bad credit score, which might make sense as there is too. If history is larger than 1.5 years and duration is higher than 39.5 a bad credit score is give, else a good credit score is given (as the remaining tree only has good as terminal nodes). This means that the ‘purpose’ and ‘employed’ are unnecessary subtrees and could be removed from the model.

Exercise 4

```
naive = naiveBayes(as.factor(good_bad)~., data=train)
test_pred = predict(naive, newdata = test)
test_pred_conf = table(test_pred,test$good_bad)
naiveMCR_test = 1- sum(diag(test_pred_conf))/sum(test_pred_conf)

train_pred = predict(naive, newdata = train)
train_pred_conf = table(train_pred,train$good_bad)
naiveMCR_train = 1- sum(diag(train_pred_conf))/sum(train_pred_conf)
```

```
> test_pred_conf

test_pred bad good
     bad   67  155
     good   5   23
> naiveMCR_test
[1] 0.64

> train_pred_conf

train_pred bad good
      bad  147  317
      good   2   34
> naiveMCR_train
[1] 0.638
```

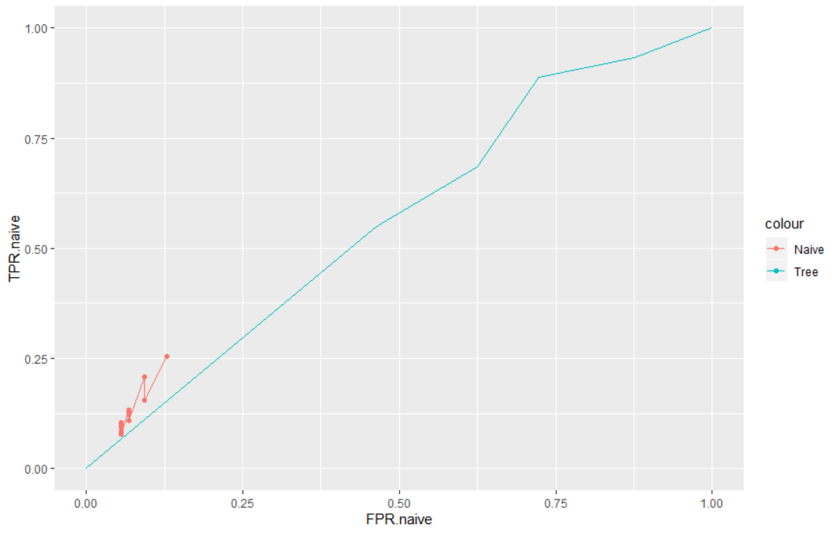
This is a pretty bad classification. The tree classification is much better.

Exercise 5

phi = seq(0.05,0.95,0.05)

```
TPR.naive = rep(0,length(phi))
FPR.naive = rep(0,length(phi))
TPR.tree = rep(0,length(phi))
FPR.tree = rep(0,length(phi))
```

```
pred_naive = predict(naive,newdata=test, type="raw")
pred_tree = predict(optTree,newdata=test)

for(i in 1:length(phi)){
  predict.naive = ifelse(pred_naive[,2]>phi[i],"good","bad")
  predict.tree = ifelse(pred_tree[,2]>phi[i],"good","bad")
  TP.naive <- length(which(predict.naive == "good" & test$good_bad == "good"))
  FP.naive <- length(which(predict.naive == "good" & test$good_bad == "bad"))

  TPR.naive[i] = (TP.naive/(TP.naive+FN.naive))
  FPR.naive[i] = (FP.naive/(FP.naive+TN.naive))

  TP.tree <- length(which(predict.tree == "good" & test$good_bad == "good"))
  TN.tree <- length(which(predict.tree == "bad" & test$good_bad == "bad"))
  FP.tree <- length(which(predict.tree == "good" & test$good_bad == "bad"))
  FN.tree <- length(which(predict.tree == "bad" & test$good_bad == "good"))
  TPR.tree[i] = (TP.tree/(TP.tree+FN.tree))
  FPR.tree[i] = (FP.tree/(FP.tree+TN.tree))

  #tN = table(predict.naive,test$good_bad)
  #TPR.naive[i] = tN[2,2]/(tN[2,2]+tN[2,1])
  #FPR.naive[i] = tN[1,2]/(tN[1,2]+tN[1,1])

  #tT = table(predict.tree,test$good_bad)
  #TPR.tree[i] = tT[1,2]/(tT[1,2]+tT[1,1])
  #FPR.tree[i] = tT[1,1]/(tT[1,2]+tT[1,1])
  #print(tN)
}
TPR.naive
FPR.naive
qplot(x=FPR.naive,y=TPR.naive,xlim=c(0,1),ylim=c(0,1),col="Naive") + geom_line() +
geom_line(aes(x=FPR.tree,y=TPR.tree,col="Tree"))
```



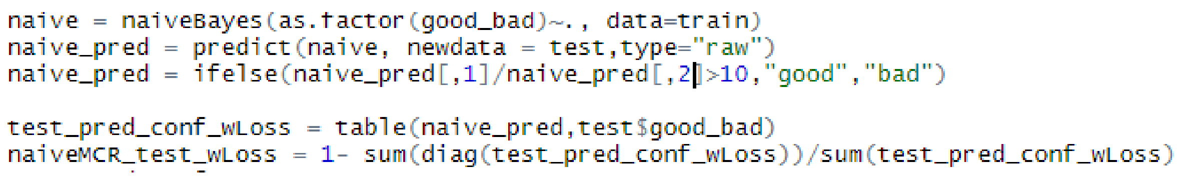
The plot shows us that naïve is better than tree as the TPR is always larger for naïve than tree for the same FPR. It does seem that the naïve values are in a strange value range, as they should have higher and lower values. This could be due to the results from exercise 4 indicating a subpar classification method for naïve.

Exercise 6

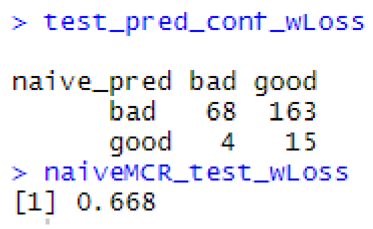
We use the loss matrix when classifying by returning the raw probabilities for each class from the predictors and then using the following equation to classify.

$$\frac{Loss_1}{Loss_2} > \frac{p_1}{p_2} \rightarrow Class = c1$$

Vikgu708



```
naive = naiveBayes(as.factor(good_bad)~., data=train)
naive_pred = predict(naive, newdata = test,type="raw")
naive_pred = ifelse(naive_pred[,1]/naive_pred[,2]>10,"good","bad")

test_pred_conf_wLoss = table(naive_pred,test$good_bad)
naiveMCR_test_wLoss = 1- sum(diag(test_pred_conf_wLoss))/sum(test_pred_conf_wLoss)
```



```
> test_pred_conf_wLoss

naive_pred bad good
      bad   68  163
      good   4   15
> naiveMCR_test_wLoss
[1] 0.668
```

Rows are predictions and columns are true values.

A mcr of 67% is much worse than what was observed in exercise 4 of about 25% mcr. This is due to the high risk of misclassifying bad as good, which means that most observations are classified as bad to minimize the cost function.

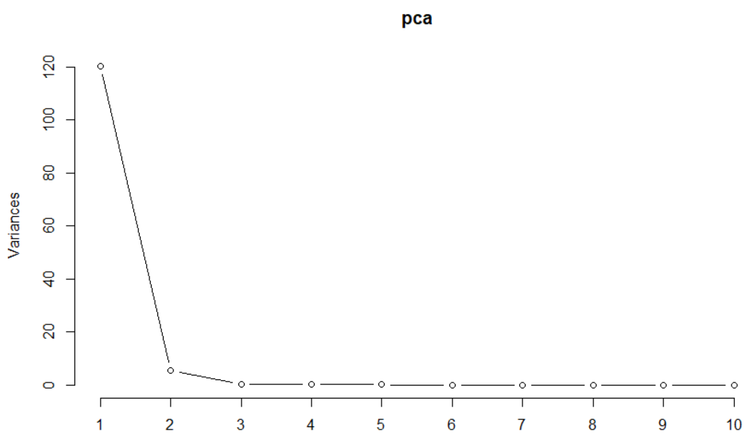
Vikgu708

Assignment 4.

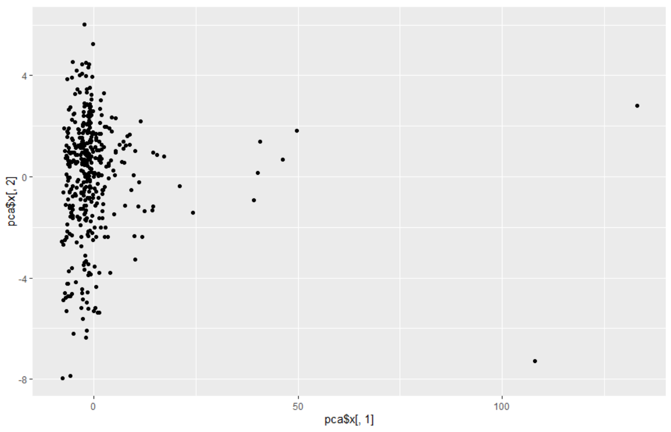
Exercise 1

```
pca <-prcomp(x = feats, center = TRUE, scale. = TRUE)

plot(pca, type="l")
```

Feats are all columns except viscosity from the dataset.



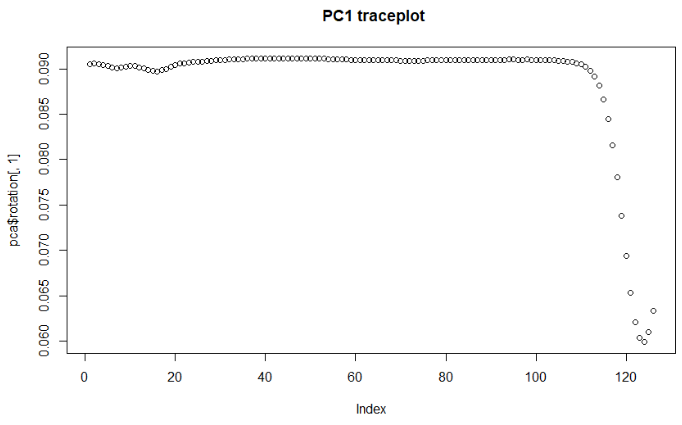
This graph is netted from the plot. As we can see, the first two variables explain 99% of the variance.

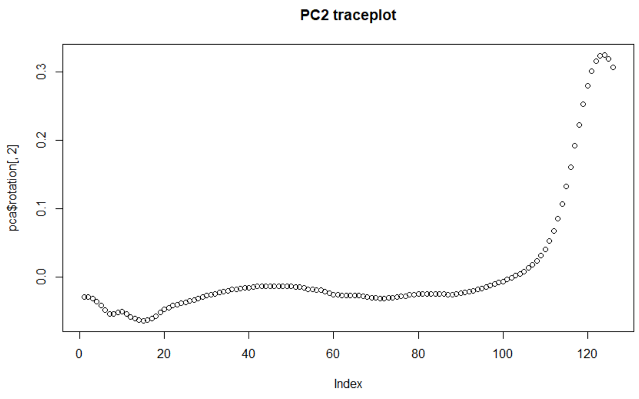


There are two outliers in PC1 which have much higher values than the others which suggest that these two corresponds to some outliers in the fuels. Otherwise the factors seem to be evenly distributed around 0 for PC2.

Vikgu708

Exercise 2





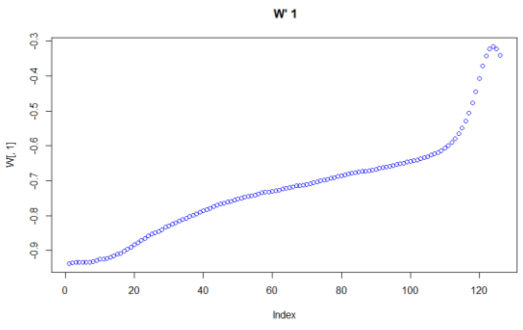
PC1 uses most components evenly except for the ones after about X850, which are barely used to explain PC1. PC2 is explained mostly by features after X860. Thus, PC2 is explained mostly by a couple of original features, mainly X860-X880.

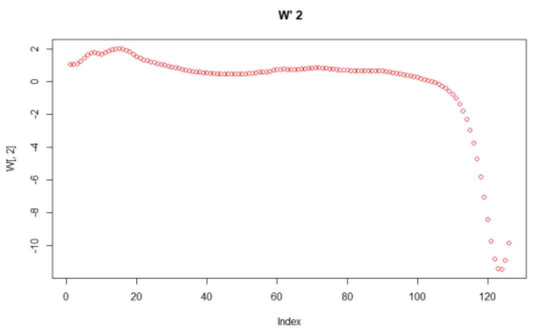
Exercise 3

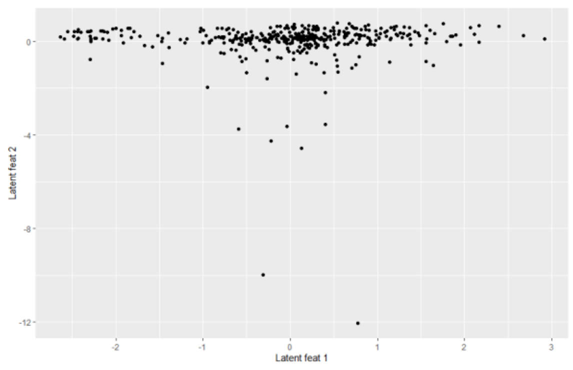
```
ICA = fastICA(X = feats, 2, fun = "logcosh",   alpha = 1, method = "R",
                 maxit = 200, alg.typ = "parallel", tol = 0.0001, row.norm = FALSE)
W = ICA$K %*% ICA$W
```

The measure of W looks similar to the PC plots before, this because W measures how much the features depends on the corresponding PC component.

Here is the plots for the ICA analysis of latent features.







It is apparent that the latent feats are similar to the PCA plot, only with inverted axis, which is correct as to the reasoning above.

Appendix

Assignment 1

```
library("MASS")

library("ggplot2")


data = read.table("australian-crabs.csv",header=TRUE,sep=",")


n=dim(data)[1]

set.seed(12345)

id=sample(1:n, floor(n*0.5))

train=data[id,]

test=data[-id,]


mcr = function (table){

  return (1-sum(diag(table))/sum(table))

}


#1.

ggplot(data,aes(x=data$CL,y=data$RW,col=data$sex)) + geom_point() + labs("Sex by
RW&CL",x="CL",y="RW")




#2.

lda2 = lda(sex ~ RW + CL, data = data )

pred2 = predict(lda2,data)$class

table2 = table(pred2,data$sex)

mcr2 = mcr(table2)


miss = !data$sex==pred2

ggplot(data,aes(x=data$CL,y=data$RW,col=pred2)) + geom_point() + labs("LDA
prediction",x="CL",y="RW")
```

Vikgu708

```
#3

lda_prior = lda(sex ~ RW + CL, data = data, prior=c(0.1,0.9))

pred_prior = predict(lda_prior,data)$class

table_prior = table(pred_prior,data$sex)

mcr_prior = mcr(table_prior)

table_prior

mcr_prior

miss = !data$sex==pred_prior


ggplot(data,aes(x=data$CL,y=data$RW,col=pred_prior)) + geom_point() + labs(title="LDA w
Prior",x="CL",y="RW")


#4

Y = as.numeric(data$sex)-1

threshold = 0.5

logistic = glm(Y ~ RW + CL, data = data)

pred_glm = predict(logistic,data)

prediction = pred_glm

prediction[which(prediction>threshold)] = "Male"

prediction[which(prediction<=threshold)] = "Female"


pred_glm>threshold


glmplot = ggplot(data,aes(x=data$CL,y=data$RW,col=prediction)) + geom_point() +
labs(x="CL",y="RW",title="Logistic prediction")

table_glm = table(as.numeric(pred_glm>threshold),data$sex)

mcr(table_glm)


w = logistic$coefficients

decision = function(x)  (threshold-w[3]*x - w[1])/w[2]

glmplot + stat_function(fun=decision)
```

Vikgu708

```
summary(lda2)
```

Assignment 4

```
library("ggplot2")

library("fastICA")


set.seed(12345)


data = data.frame(read.csv2("NIRSpectra.csv"))

feats = data

feats$Viscosity = c()


pca =prcomp(x = feats, scale. = TRUE, center =TRUE)


feats

plot(pca, type = "l" )


# PC1 + PC2 explains >99% of the variance

qplot(x = pca$x[,1], y = pca$x[,2])


# 2.2

plot(pca$rotation[,1], main =" PC1 traceplot")

plot(pca$rotation[,2], main ="PC2 traceplot")


# 2.3

RNGversion("3.51")

set.seed(12345)

ICA = fastICA(X = feats, 2, fun = "logcosh",   alpha = 1, method = "R", maxit = 200, alg.typ = "parallel",
tol = 0.0001, row.norm = FALSE)

W = ICA$K %*% ICA$W

plot(W[,1], main = " W' 1",col="blue")
```

Vikgu708

```
plot(W[,2], main = " W' 2",col="red")
qplot(x = a$S[,1], y = a$S[,2], xlab = "Latent feat 1", ylab = "Latent feat 2" , data = feats)
```