Lab2

David Nyberg

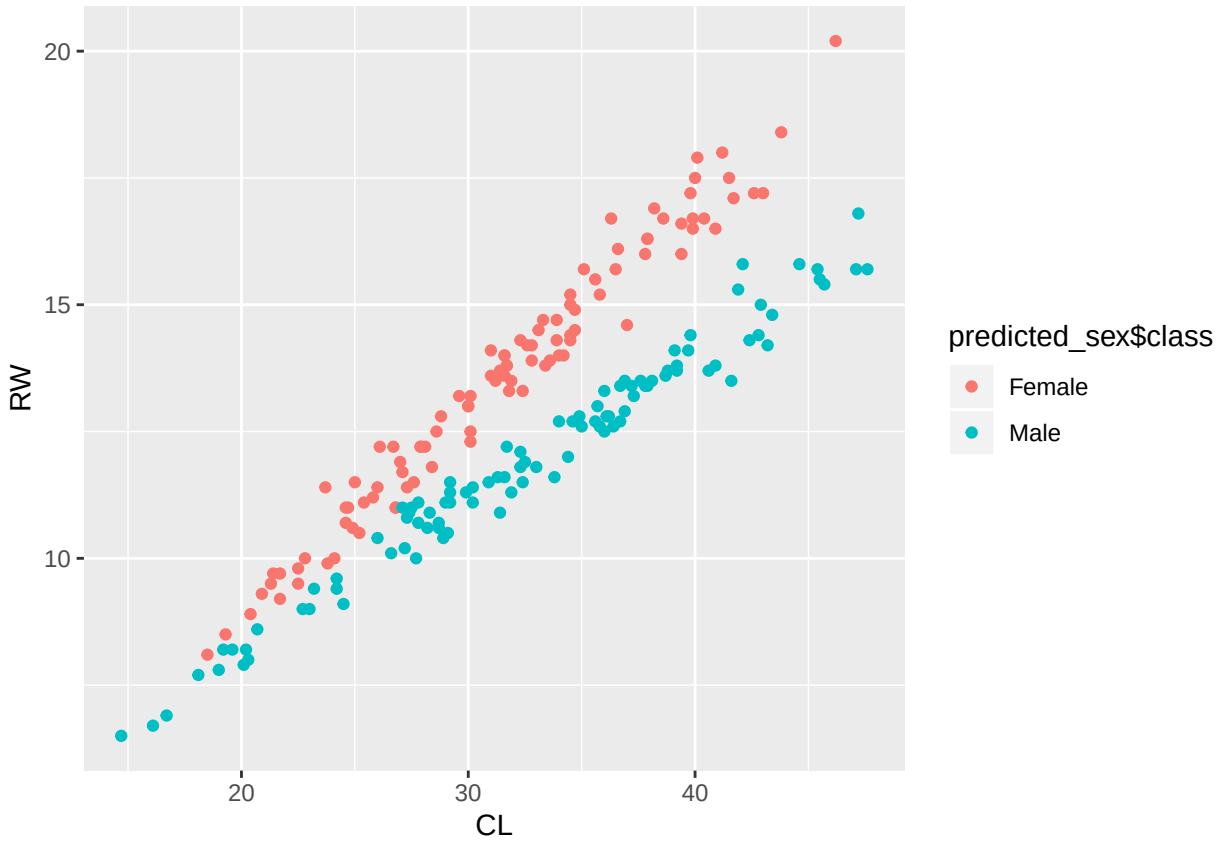
11/25/2019

Assignment 1. LDA and logistic regression

1) This is easily explained by LDA because it is linear based on the plot, and it is also clearly distiguishing between two classes based on sex.

2) Using lda function in mass, comparing our two plots they look almost the exact same, error rate of 0.35 is.

```
##
##          Female Male
##   Female     97    4
##   Male        3   96
```

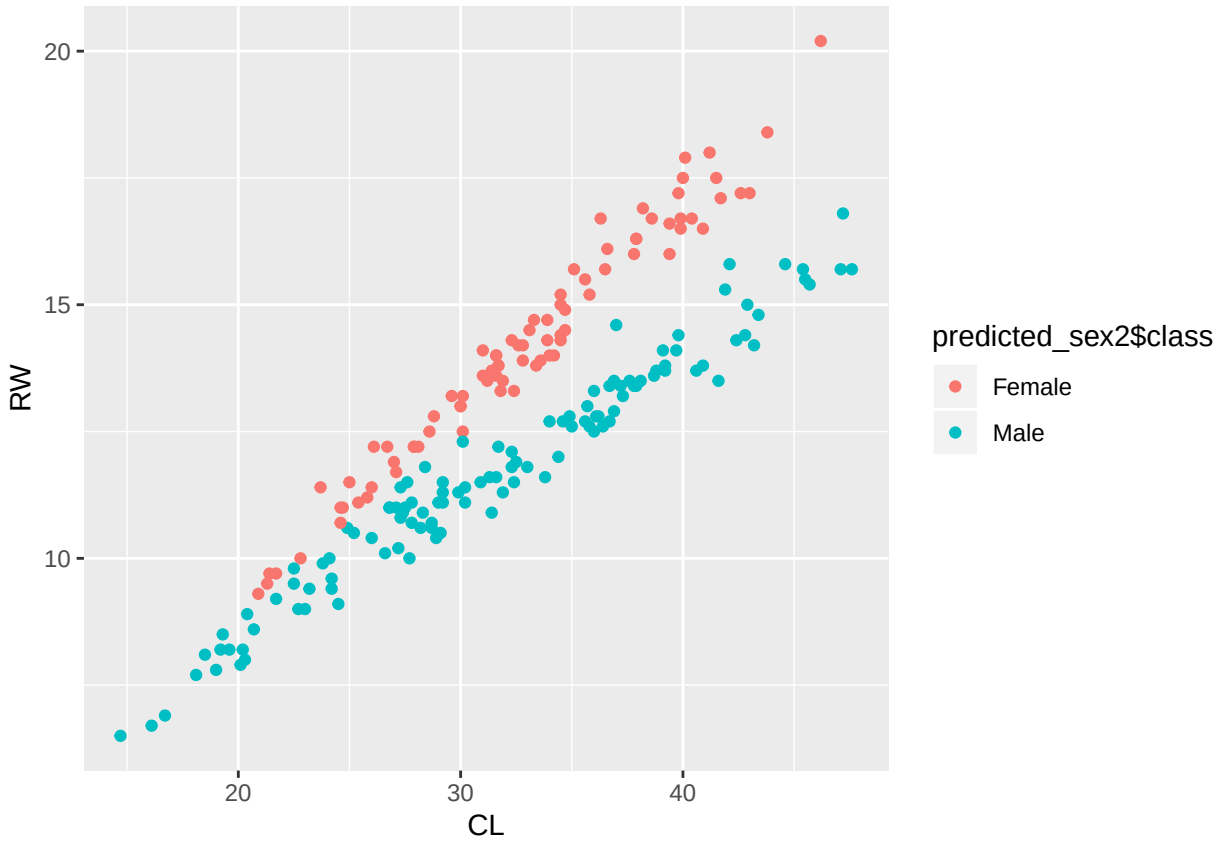



```
## [1] 0.035
```

3) Using prior p of male = .9 female = .1, this gave us a worse missclassification rate of .08, specifically on the table we can see we had more false negatives (female crabs classified as males) this is because we gave it the probability of being male .9 so it gave higher weight towards classifying crabs as males. the graph almost the same, a few more points being male in the lower right section

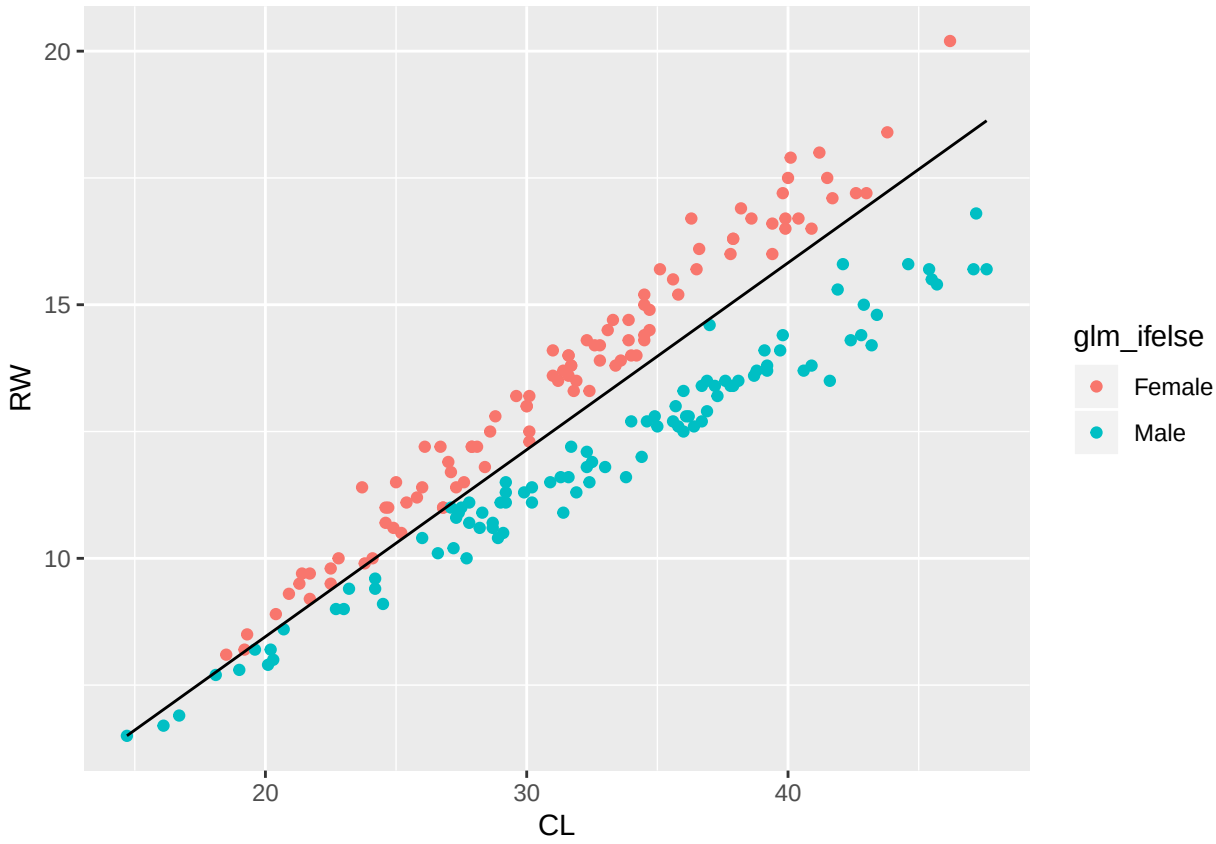
```
##
##          Female Male
##   Female     84    0
##   Male       16  100
```

```
## [1] "Misclassification rate:  0.08"
```



4) repeating with logistic regression, our misslclassification rate is the exact same with 0.035, and the plots look essentially the same. The equation we got from models ceofficients to be y = 0.369x + 1.08.

```
## [1] "Misclassification rate:  0.035"
```



#Assignment 2. Analysis of credit scoring

1) import and split into train/validation/test, 50/25/25

```
data <- read_excel("creditscoring.xls", col_names = TRUE)
#View(data)
n=dim(data)[1]
set.seed(12345)
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,]
```

2)

Misclassification rate for train data for decision trees 0.212 - deviance 0.240 - gini

Misclassification rate for test data for decision trees 0.268 - deviance 0.368 - gini

3) Will choose to continue with deviance tree because it gave less error on train and testing data. We now have our optimal tree of 4 leaves with depth of 3 based on the plot we can see 4 is where we have the least amount of error. We pruned the tree and predicted with it, getting missclassification rate of 0.256, which is a little bit better than our previous result of 0.268. Printed is the optimal decision tree showing the variables chosen: savings, duration, and history.

```
##       devi_pred.train
##        bad good
##   bad   61   86
##   good  20  333
```

```
##       gini_pred.train
##        bad good
##   bad   66   81
##   good  39  314
```

```
##       devi_pred.test
##        bad good
##   bad   28   48
##   good  19  155
```

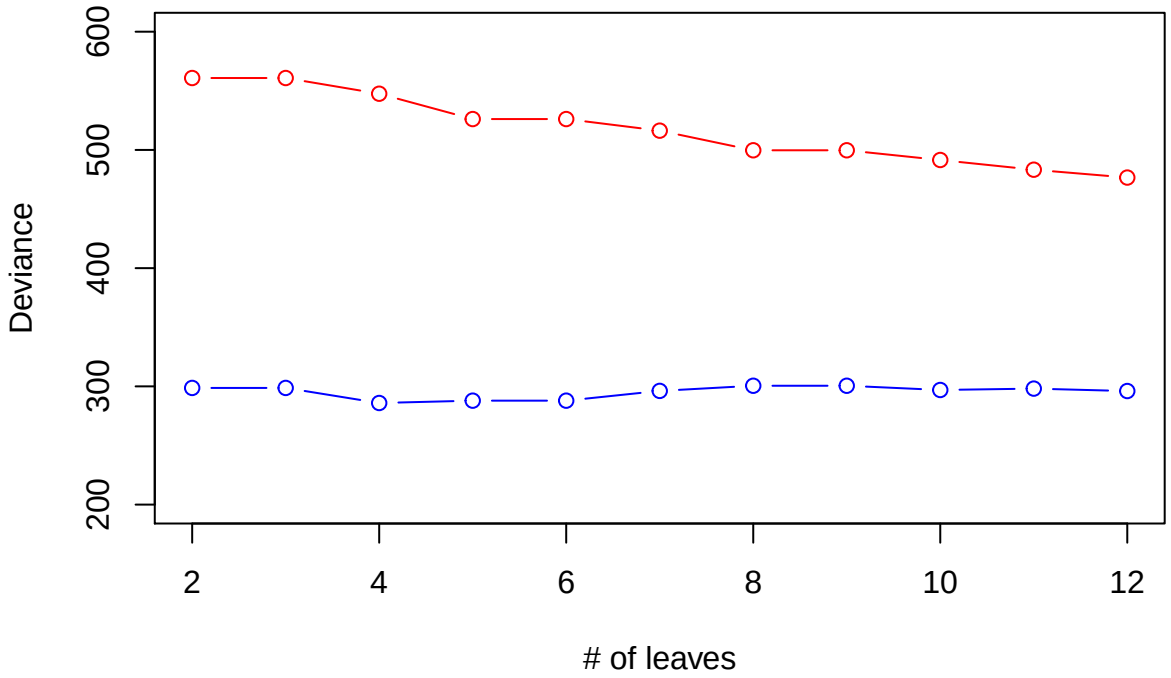
```
##       gini_pred.test
##        bad good
##   bad   18   58
##   good  34  140
```

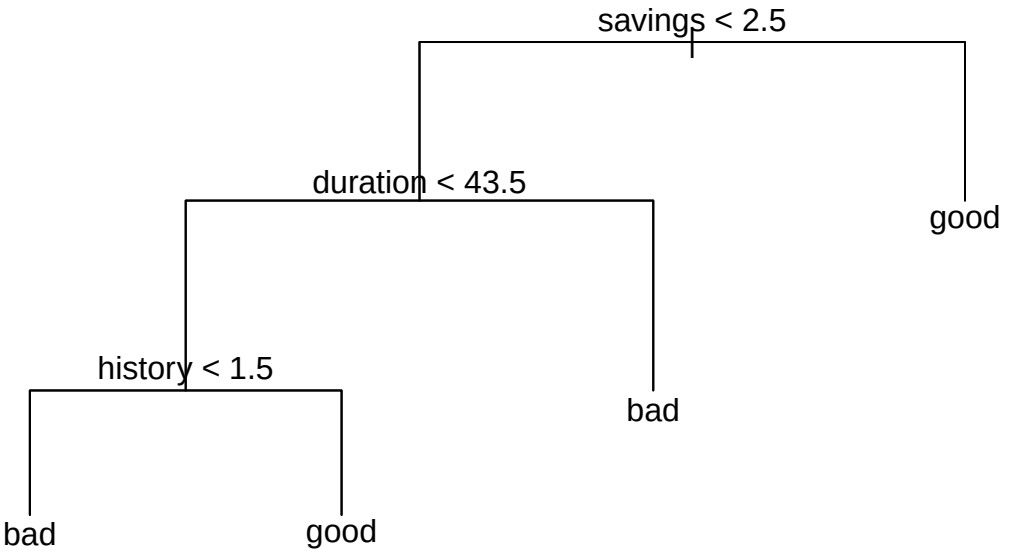
```
## [1] 0.212
```

```
## [1] 0.24
```

```
## [1] 0.268
```

```
## [1] 0.368
```





4) Classification with naive bayes and predicted on test/train data we get misclassification rate of train=0.3, test=0.316. This seems like a good fit overall as the test two rates are close to each other. Comparing to step 3 with our decision tree we can see the tree has a better model based off less error.

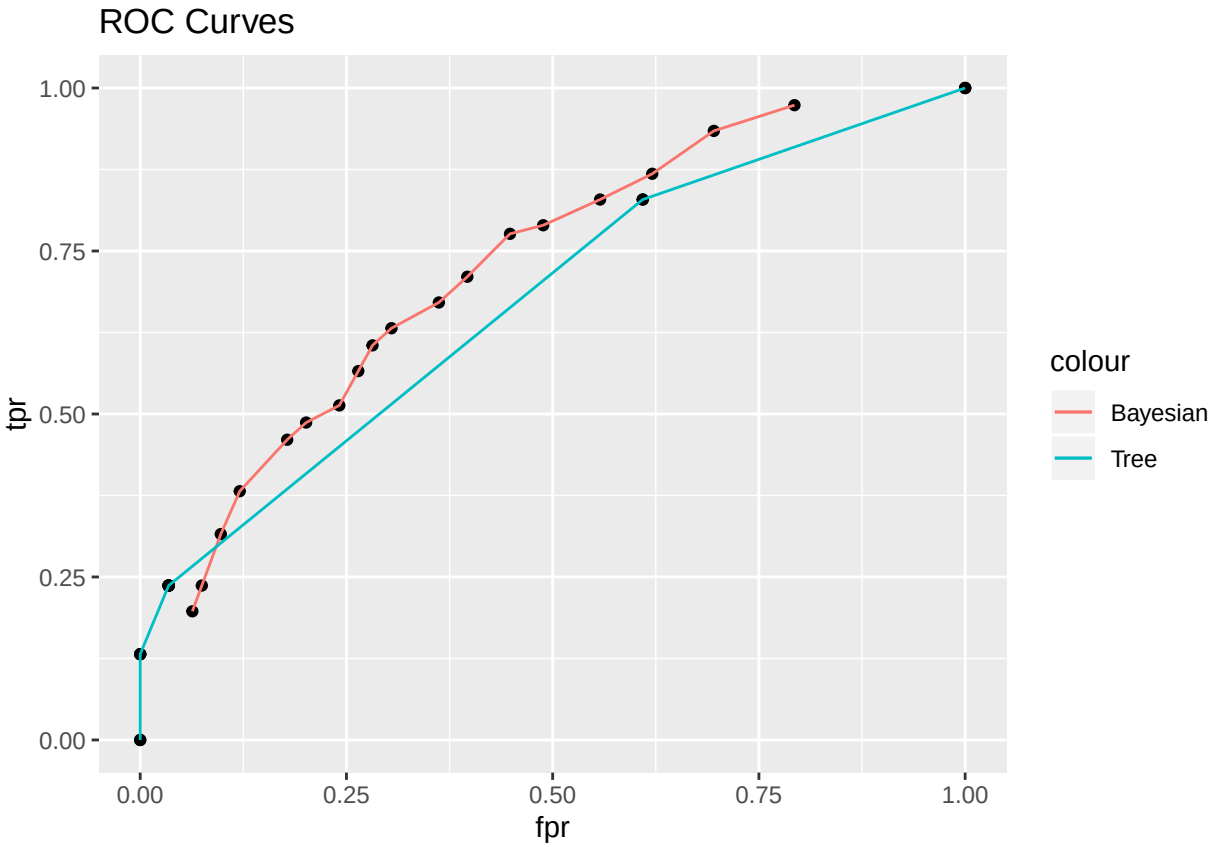
```
## Warning in data.matrix(newdata): NAs introduced by coercion

## Warning in data.matrix(newdata): NAs introduced by coercion

## Confusion Matrix and Statistics
##
##           Reference
## Prediction bad good
##       bad   95   98
##       good  52  255
##
##                Accuracy : 0.7
##                  95% CI : (0.6577, 0.7399)
##     No Information Rate : 0.706
##     P-Value [Acc > NIR] : 0.6366258
##
##                   Kappa : 0.3378
##
##  Mcnemar's Test P-Value : 0.0002386
##
##             Sensitivity : 0.6463
##             Specificity : 0.7224
##          Pos Pred Value : 0.4922
##          Neg Pred Value : 0.8306
##              Prevalence : 0.2940
##          Detection Rate : 0.1900
##    Detection Prevalence : 0.3860
##       Balanced Accuracy : 0.6843
##
##        'Positive' Class : bad
##

## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction bad good
##       bad   46   49
##       good  30  125
##
##                Accuracy : 0.684
##                  95% CI : (0.6224, 0.7411)
##     No Information Rate : 0.696
##     P-Value [Acc > NIR] : 0.68723
##
##                   Kappa : 0.3024
##
##  Mcnemar's Test P-Value : 0.04285
##
##             Sensitivity : 0.6053
##             Specificity : 0.7184
##          Pos Pred Value : 0.4842
##          Neg Pred Value : 0.8065
##              Prevalence : 0.3040
##          Detection Rate : 0.1840
##    Detection Prevalence : 0.3800
##       Balanced Accuracy : 0.6618
##
##        'Positive' Class : bad
##
```

```
## [1] "Misclassification rate:  0.3"
```

```
## [1] "Misclassification rate:  0.316"
```

5) Using the principle given we computed many different models to classify the data based off different probabilities, we then used ROC curves to evaluate the results. We can say based off these curves that the better model has a bigger area under its curve, in our results we can see the bayesian curve performs better as it's above the tree.



6)

Using loss matrix we can see our misclassification rates increased a lot. We are saying that false negatives are ten times worse than others meaning we should not have many of those values which is true looking at our confusion matrix. We have very few values in the false negative bottom left.

```
##
##         bad good
##   bad   137  263
##   good   10   90
```

```
##
##         bad good
##   bad    71  122
##   good    5   52
```

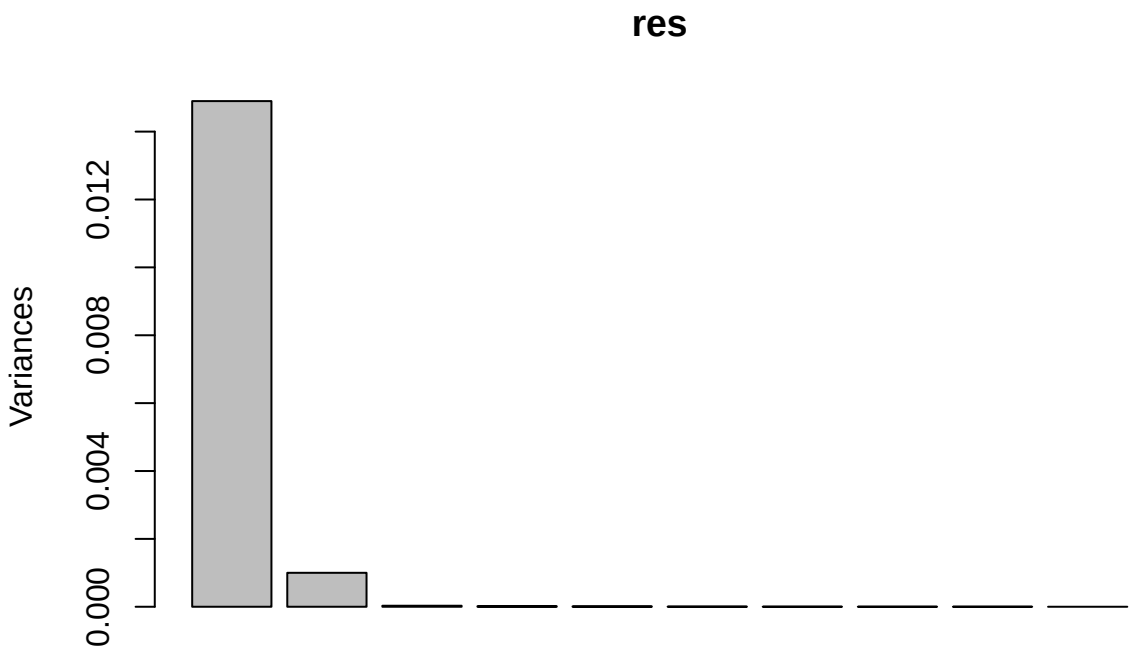
```
## [1] "Misclassification rate train:  0.546"
```

```
## [1] "Misclassification rate test:  0.508"
```

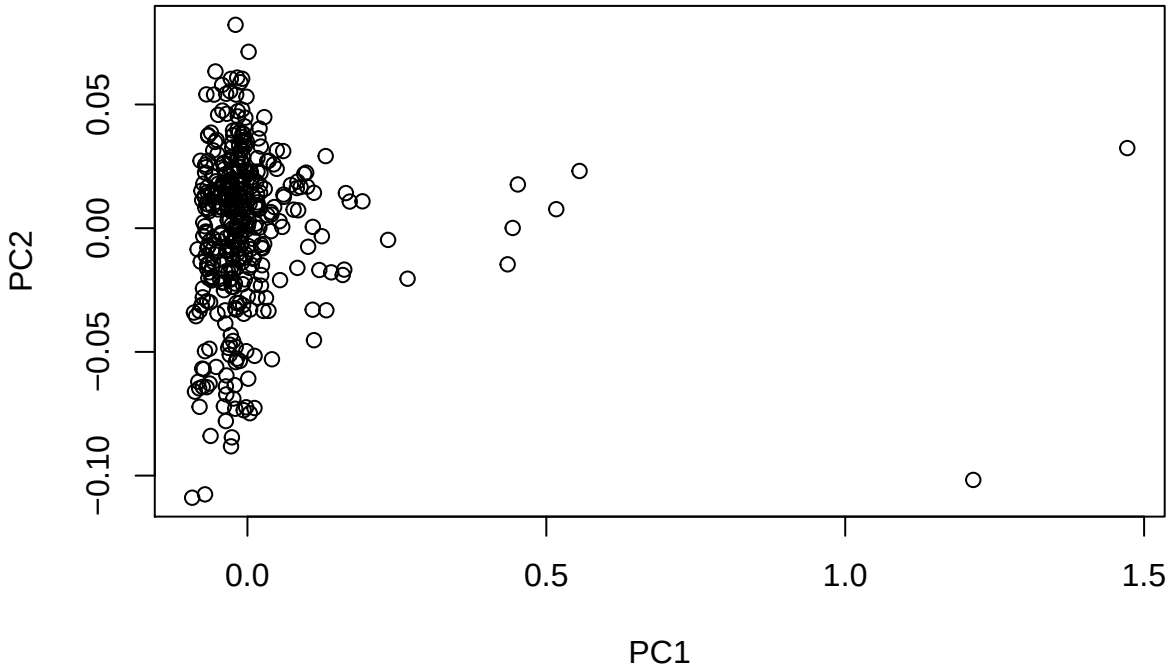
Assignment 4: Principal Components

1) The historgam plot doesn't tell us exactly how many PC's to choose but it shows a good estimate showing how important they are based off their height, we can see it drops off very quickly with most of the variance being captured in the first PC.

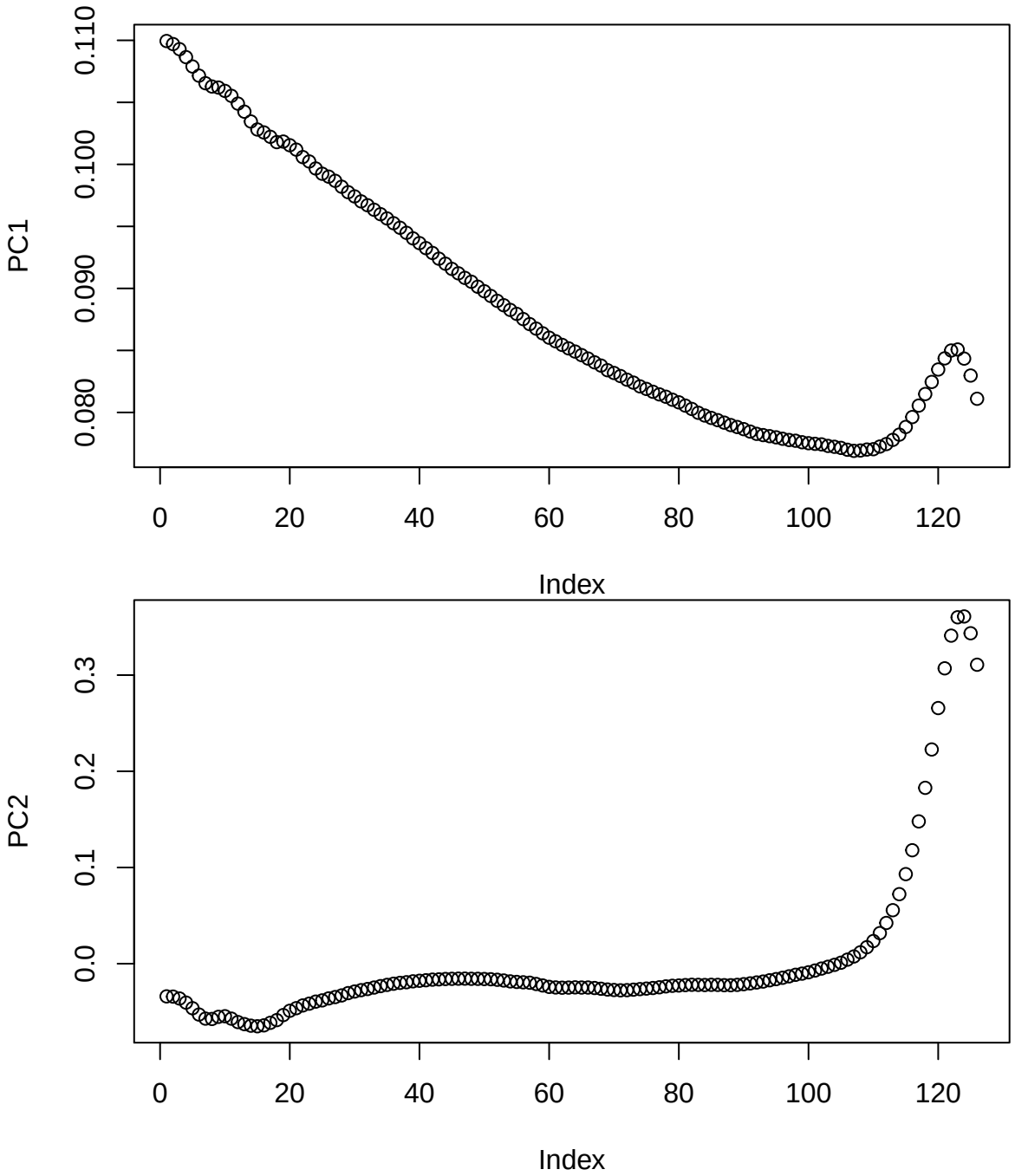
The first two principal components account for 95.38% and 4.229% of the variance, so them two together describe more than 99% of the variance.



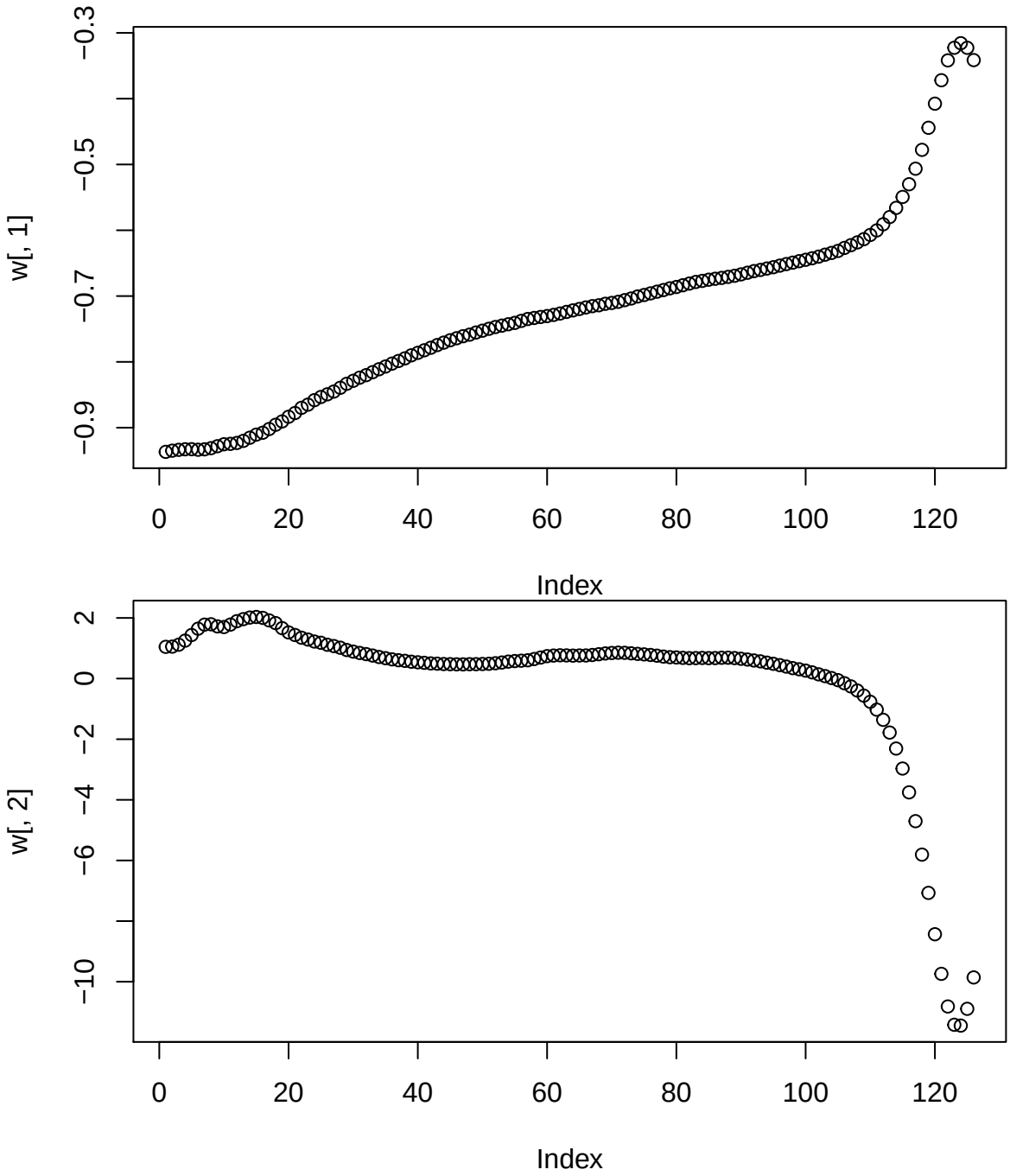
Here we can see our plot of the scores in the coordinates of PC1 and PC2 showing the different fuels. Most are clustered together, some unusual outliers can be seen such as the two points above 100 in the PC1 axis.



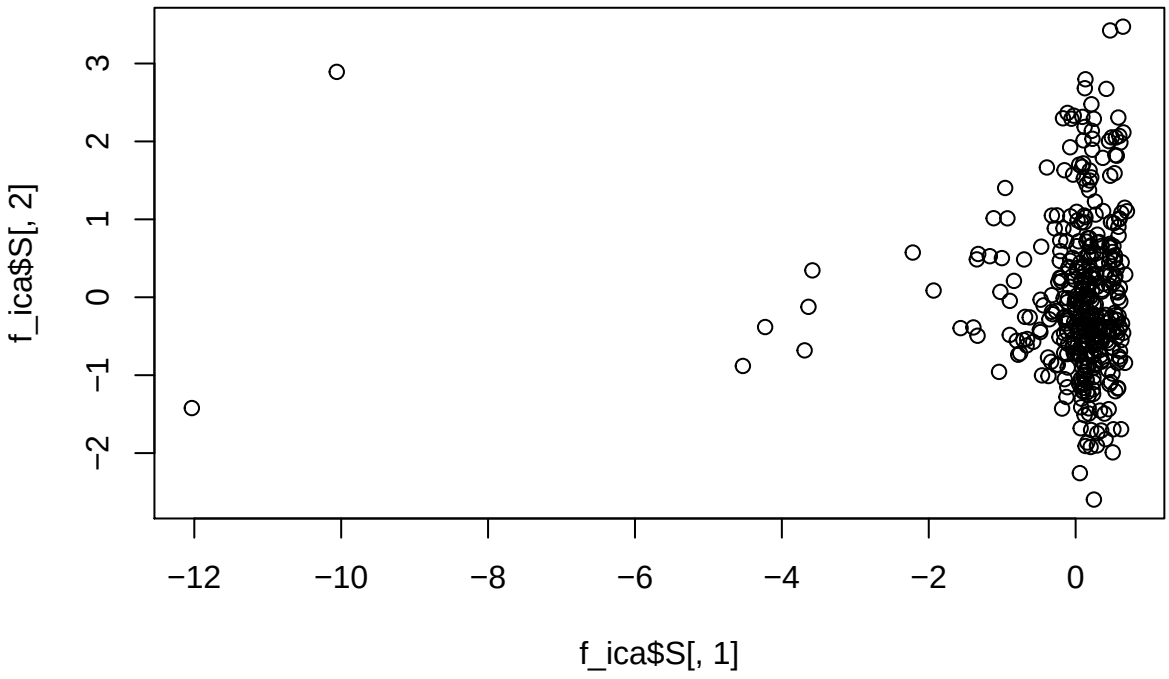
2) The following two traceplots are pc1 and pc2 show how much the features impact the variance. We can see in PC2 that most the values are around zero, except for the last ~10 features have more impact. In PC1 the first features have more importance but slowly decrease in impact.



3a) After computing our Wprime = K * W from ICA results, plotting the first two columns we can see that they resemble the PC1/2 plots but they are flipped. K is projecting our data onto the prinicple components and W is an estimated un-mixing matrix. So Wprime is essentially our unmixed results projected onto our components.



3b) Using the estimated source matrix we can plot the first two columns which gives us a plot that is also very similar to our first score plot but flipped.



Appendix

```
knitr::opts_chunk$set(echo = TRUE)
RNGversion('3.5.1')
library(ggplot2)
library(MASS)
library(readxl)
library(tree)
library(e1071)
library(fastICA)
library(caret)
crabs <-  read.csv("australian-crabs.csv", header = TRUE)
ggplot(crabs, aes(CL, RW)) + geom_point(aes(color = sex))
lda_c <- lda(sex ~ CL + RW , data = crabs)
#lda_c$prior

predicted_sex <- predict(lda_c, crabs)
table(predicted_sex$class, crabs$sex)
ggplot(crabs, aes(CL, RW)) + geom_point(aes(color = predicted_sex$class))
mean(predicted_sex$class != crabs$sex)
p <- c(.1, .9)
lda_c2 <- lda(sex ~ CL + RW , data = crabs, prior = p)
#lda_c2$prior
predicted_sex2 <- predict(lda_c2, crabs)
#predicted_sex2$class

table(predicted_sex2$class, crabs$sex)
print(paste("Misclassification rate: ", mean(predicted_sex2$class != crabs$sex)))

ggplot(crabs, aes(CL, RW)) + geom_point(aes(color = predicted_sex2$class))
```

```r
glm_c <- glm(sex ~ CL + RW, data = crabs, family = binomial)

predicted_glm <- predict(glm_c, crabs, type = "response")

glm_ifelse <- ifelse(predicted_glm > 0.5, "Male", "Female")

print(paste("Misclassification rate: ", mean(glm_ifelse!= crabs$sex)))

#ggplot(crabs, aes(CL, RW)) + geom_point(aes(color = predicted_glm))

#coef(glm_c)
slope <- coef(glm_c)[2]/(-coef(glm_c)[3])
intercept <- coef(glm_c)[1]/(-coef(glm_c)[3])

#graph with decision boundary
ggplot(crabs, aes(CL, RW)) + geom_point(aes(color = glm_ifelse)) +
  stat_function(fun = function(x) {slope * x + intercept})

#https://stats.stackexchange.com/questions/6206/how-to-plot-decision-boundary-in-r-for-logistic-regress

#https://stackoverflow.com/questions/44834659/drawing-the-glm-decision-boundary-with-ggplots-stat-smoot
data <- read_excel("creditscoring.xls", col_names = TRUE)
#View(data)
n=dim(data)[1]
set.seed(12345)
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,]
#good_bad are strings so convert to 2 factors.
m_tree_devi <- tree(as.factor(good_bad) ~ ., data = train, split = "deviance")
m_tree_gini <- tree(as.factor(good_bad) ~ ., data = train, split = "gini")

#plot(m_tree_devi)
#text(m_tree_devi, pretty = 0)

#plot(m_tree_gini)
#text(m_tree_gini, pretty = 0)


devi_pred.test <- predict(m_tree_devi, newdata = test, type = "class") #0.268
gini_pred.test <- predict(m_tree_gini, newdata = test, type = "class") #0.372

devi_pred.train <- predict(m_tree_devi, newdata = train, type = "class") #0.212
gini_pred.train <- predict(m_tree_gini, newdata = train, type = "class") #0.24

table(train$good_bad, devi_pred.train)
```

```r
table(train$good_bad, gini_pred.train)

table(test$good_bad, devi_pred.test)
table(test$good_bad, gini_pred.test)

mean(devi_pred.train != train$good_bad)
mean(gini_pred.train != train$good_bad)

mean(devi_pred.test != test$good_bad)
mean(gini_pred.test != test$good_bad)

#will choose to continue with  deviance tree because it gave less error on train and testing data.
trainScore=rep(0,12)
testScore=rep(0,12)
for(i in 2:12)
  {
  prunedTree=prune.tree(m_tree_devi,best=i)
  pred=predict(prunedTree, newdata=valid, type="tree")
  trainScore[i]=deviance(prunedTree)
  testScore[i]=deviance(pred)
  }
plot(2:12, trainScore[2:12], type="b", col="red", ylim=c(200,600), xlab = "# of leaves", ylab = "Devian
points(2:12, testScore[2:12], type="b", col="blue")
#choose 4 because its the lowest test error
optimal_tree <- prune.tree(m_tree_devi, best = 4)
opt_predict <- predict(optimal_tree, newdata = test, type = "vector")
#opt_predict <- predict(optimal_tree, newdata = test, type = "class")

#mean(opt_predict != as.factor(test$good_bad))
#final tree
#test$good_bad
plot(optimal_tree)
text(optimal_tree, pretty = 0)
bayes <- naiveBayes(as.factor(good_bad) ~ ., data = train)
bay.p.train <- predict(bayes, train)
bay.p.test <- predict(bayes, test)
#bay.p.train

mean1 <- mean(bay.p.train != train$good_bad)
mean2 <- mean(bay.p.test != test$good_bad)

confusionMatrix(bay.p.train, as.factor(train$good_bad))
confusionMatrix(bay.p.test, as.factor(test$good_bad))


print(paste("Misclassification rate: ", mean1))
print(paste("Misclassification rate: ", mean2))

myseq <- seq(0.05, 0.95, by = 0.05)
a <- predict(bayes, test, type="raw")

#https://stackoverflow.com/questions/5738831/r-plus-equals-and-plus-plus-equivalent-from-c-c-java-etc
#function to do i++ ,  not necessary... idk
```

```
inc <- function(x)
{
 eval.parent(substitute(x <- x + 1))
}

tpr <- c()
fpr <- c()
index <- 0
index2 <- 0

tpr2 <- c()
fpr2 <- c()
roc <- rep(0,18)
for (i in myseq) {
  temp <- table(test$good_bad, ifelse(a[,2] > i , 1, 0))
  #print(temp)
  #print(temp[2])
  tpr[index] <- temp[1,1] / (temp[1,1] + temp[1,2])
  fpr[index] <- temp[2,1] / (temp[2,1] + temp[2,2])
  inc(index)
}


for (i in myseq) {
  temp <- ifelse(opt_predict[,2] > i, 1, 0)
  #print(temp)
  #temp <- table(ifelse(opt_predict[,2] > i, "good", "bad"))
 # tpr2[index2] <- temp[1,1] / (temp[1,1] + temp[2,1])
 # fpr2[index2] <- temp[1,2] / (temp[1,2] + temp[2,2])
  temp1 <- confusionMatrix(as.factor(ifelse(opt_predict[,2]>i, "good", "bad")),as.factor(test$good_bad))
  tpr2[index2] <- temp1$table[1]/(temp1$table[1]+temp1$table[2])
  fpr2[index2] <- temp1$table[3]/(temp1$table[3]+temp1$table[4])

  #roc[index2] <- temp
  #tpr2[index2] <- TP/(TP + FN)
  #fpr2[index2]<- FP/(FP + TN)
  inc(index2)
}

#plot(roc)

#table(ifelse(opt_predict[,2] > .1 , 1, 0), test$good_bad)

myt <- table(test$good_bad, opt_predict[,2])
TPR1 <- myt[1,1] / (myt[1,1] + myt[1,2])
FPR1 <- myt[2,1] / (myt[2,1] + myt[2,2])
#myt


df <- data.frame(tpr = tpr, fpr=fpr, tpr2 = tpr2, fpr2 = fpr2)

ggplot() + geom_point(aes(x=fpr, y = tpr)) + geom_line(aes(x=fpr, y = tpr, color = 'Bayesian')) +
  geom_point(aes(x=fpr2, y = tpr2)) + geom_line(aes(x=fpr2, y = tpr2, color = 'Tree')) +
```

```
  ggtitle("ROC Curves")

#plot(x = fpr, y = tpr, type = "b", main = "ROC curves")
#par(new = TRUE)
#plot(x = fpr2, y = tpr2, type ="b")
#fpr2
#confusionMatrix(as.factor(ifelse(bayloss[,2]>10, "good", "bad")),as.factor(test$good_bad))

#Code help from group members
Yfit <- predict(bayes, newdata=test, type="raw")
Yfit[,"bad"] <- Yfit[,"bad"]*10
Yfit2 <- replace(Yfit, Yfit[,1]>Yfit[,2], "bad")
Yfit2 <- replace(Yfit2, Yfit[,1]<=Yfit[,2], "good")

t1<-table(Yfit2[,1],test$good_bad)

Yfit_train <- predict(bayes, newdata=train, type="raw")
Yfit_train[,"bad"] <- Yfit_train[,"bad"]*10
Yfit2_train <- replace(Yfit_train, Yfit_train[,1]>Yfit_train[,2], "bad")
Yfit2_train <- replace(Yfit2_train, Yfit_train[,1]<=Yfit_train[,2], "good")

t2<-table(Yfit2_train[,1],train$good_bad)


t2
t1
print(paste("Misclassification rate train: ",1 - sum(diag(t2))/sum(t2)))
print(paste("Misclassification rate test: ",1 - sum(diag(t1))/sum(t1)))
nir <- read.csv2("NIRspectra.csv", header = TRUE)
res <- prcomp(nir[1:126]) #dont include viscosity, scale = true
#summary(res)
#apply(nir[1:126], 2, mean)

lambda <- res$sdev^2 #square stddev for variance
#lambda #eigenvalues
s <-sprintf("%2.3f",lambda/sum(lambda)*100)
screeplot(res )
#prof <- as.data.frame(res$x)
#ggplot(prof ) + geom_point(aes(x=prof[,1], y=prof[,2]), color = nir$Viscosity)

plot(res$x[,1], res$x[,2], xlab = "PC1", ylab = "PC2")
u <- res$rotation #loading scores
plot(u[,1], ylab = "PC1")
plot(u[,2], ylab = "PC2")
set.seed(12345)
f_ica <- fastICA(nir[1:126], 2)
#plot(f_ica$X[,1], f_ica$X[,2])

w <- f_ica$K %*% f_ica$W
#f_ica$W
#w
plot(w[,1])
plot(w[,2])
```

```r
#use s to estimate source matrix
plot(f_ica$S[,1], f_ica$S[,2])
```