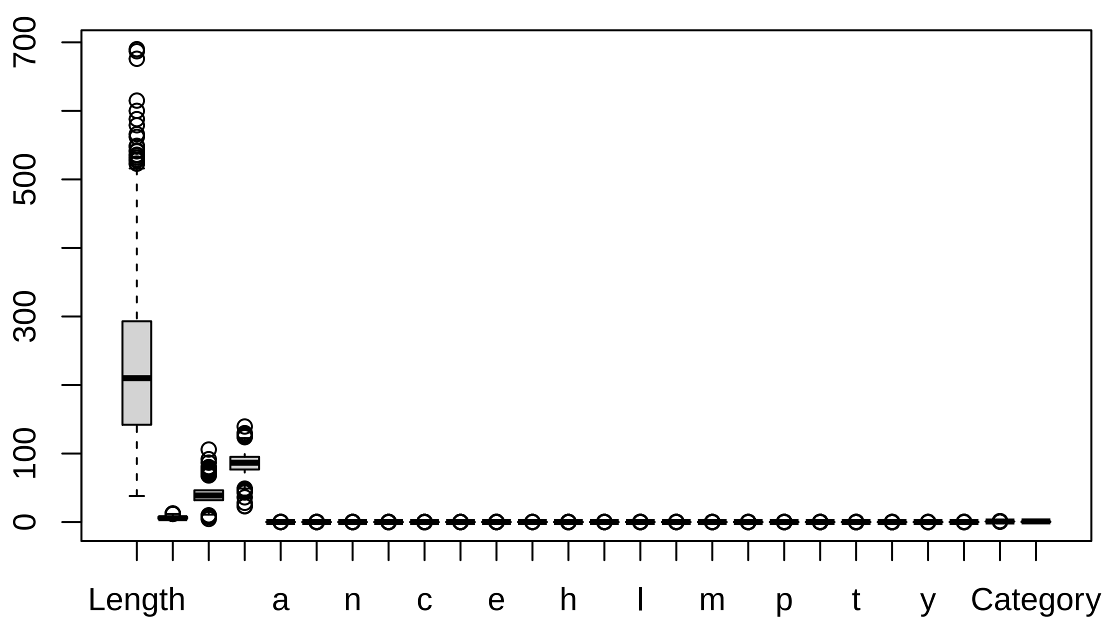
MAT00058H - Homework 3

Matthew Knowles

02/02/2021

Loading and Overview of the data

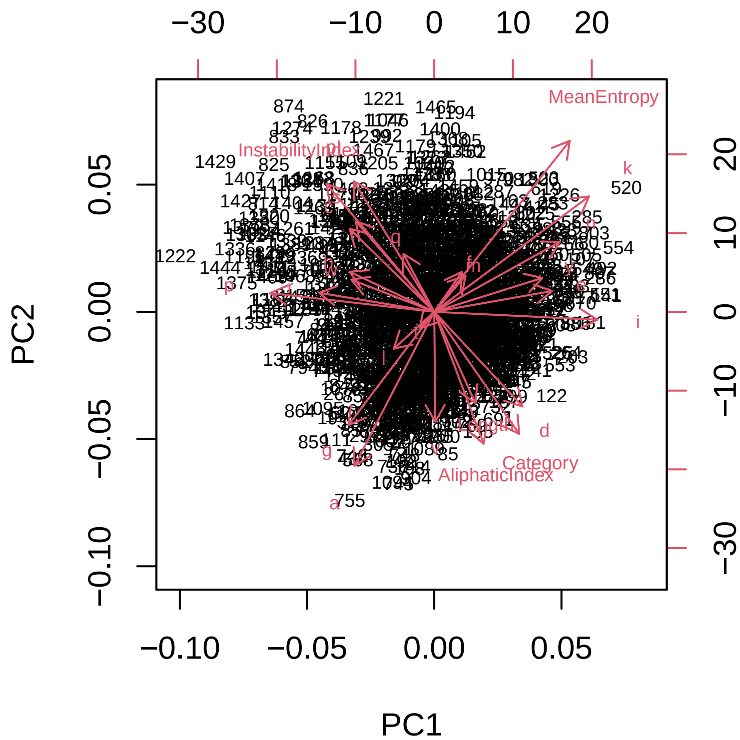
```r
#Training Data
crystalData <- read.table("/home/matthew/Documents/University/PDS/Datasets/train1500.txt",header = T)
#Test Data
testData <- read.table("/home/matthew/Documents/University/PDS/Datasets/test144.txt", header = T)
boxplot(crystalData)
```



We can see from the boxplot of the test data that the first couple of variables massively dominate the analysis. For this reason, we scale the data when performing the PCA.

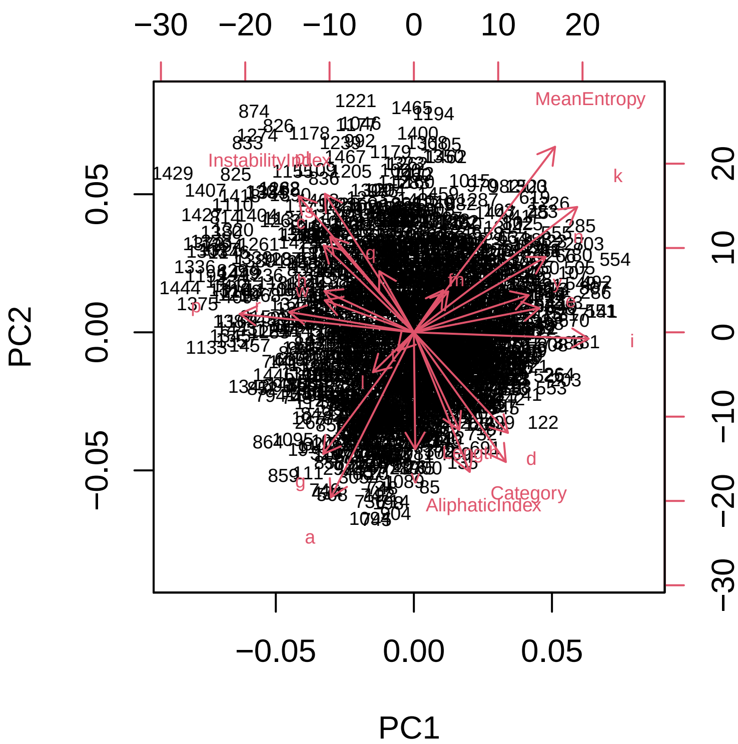
PCA

```r
crysPCA <- prcomp(crystalData,scale=T)
biplot(crysPCA, cex= 0.6)
```



Due to the shear amount of data points, it's hard to make much out in the middle, but that's okay. We can see 3 potential outliers: 520, 755 and 1222. We can remove these with ease and redo the PCA.

```r
crystalData <- rbind(crystalData[1:519,], crystalData[521:754,],
                     crystalData[756:1221,], crystalData[1223:1500,])
crysPCA <- prcomp(crystalData, scale=T)
biplot(crysPCA, cex=0.6)
```




```
summary(crysPCA)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.9705 1.8633 1.3823 1.31466 1.25137 1.12699 1.08906
## Proportion of Variance 0.1493 0.1335 0.0735 0.06647 0.06023 0.04885 0.04562
## Cumulative Proportion  0.1493 0.2829 0.3564 0.42284 0.48307 0.53192 0.57753
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.04552 0.99310 0.95463 0.95084 0.91876 0.89593 0.87406
## Proportion of Variance 0.04204 0.03793 0.03505 0.03477 0.03247 0.03087 0.02938
## Cumulative Proportion  0.61958 0.65751 0.69256 0.72733 0.75980 0.79067 0.82006
##                           PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     0.84686 0.83331 0.80366 0.79498 0.78011 0.73434 0.70919
## Proportion of Variance 0.02758 0.02671 0.02484 0.02431 0.02341 0.02074 0.01934
## Cumulative Proportion  0.84764 0.87435 0.89919 0.92350 0.94690 0.96764 0.98699
##                           PC22    PC23    PC24      PC25      PC26
## Standard deviation     0.50453 0.28798 0.02894 0.0007833 1.239e-08
## Proportion of Variance 0.00979 0.00319 0.00003 0.0000000 0.000e+00
## Cumulative Proportion  0.99678 0.99997 1.00000 1.0000000 1.000e+00
```

We can see from the above graph that this data is now okay to be classified. Further, from the summary, we can see that using 18 prinicpal components will be required for performing LDA, which will be relevant later on.

Naiive-Bayes with CARET

We implement this in the standard way, but since the 'category' variable is a number we must use *as.factor()* to make this work.

```
crystalData$Category <- as.factor(crystalData$Category)
nb = train(Category~., method="naive_bayes", data=crystalData,
           trControl=trainControl(method="cv",number=3, savePredictions = T),
           preProcess=c("center","scale"))
nb$results
```

```
##   usekernel laplace adjust  Accuracy     Kappa AccuracySD    KappaSD
## 1     FALSE       0      1 0.7488290 0.4968832 0.01338712 0.02680282
## 2      TRUE       0      1 0.7454876 0.4900984 0.01070668 0.02137785
```

```
table(true=nb$pred[,2],predicted=nb$pred[,1])
```

```
##     predicted
## true    0    1
##    0  945  541
##    1  216 1292
```

We can get the accuracy of this table by doing some summing on the table:

```
sum(diag(prop.table(table(true=nb$pred[,2],predicted=nb$pred[,1]))))
```

```
## [1] 0.747161
```

This is good, but let's change a few parameters and see if it can be improved.

```
crystalData$Category <- as.factor(crystalData$Category)
nb = train(Category~., method="naive_bayes", data=crystalData,
           trControl=trainControl(method="cv",number=3, savePredictions = T),
           preProcess=c("center","scale"), tuneGrid=expand.grid(laplace = 0, usekernel=T, adjust=1))
nb$results
```

```
##   laplace usekernel adjust  Accuracy     Kappa AccuracySD    KappaSD
## 1       0      TRUE      1 0.7508062 0.5008158 0.02244767 0.04500097
```

```
table(true=nb$pred[,2],predicted=nb$pred[,1])
```

```
##     predicted
## true   0   1
##    0 477 266
##    1 107 647
```

```
sum(diag(prop.table(table(true=nb$pred[,2],predicted=nb$pred[,1]))))
```

```
## [1] 0.750835
```

By this time changing some of the variables we have improved the accuracy by 1%!.

```r
crystalData$Category <- as.factor(crystalData$Category)
nb = train(Category~., method="naive_bayes", data=crystalData,
           trControl=trainControl(method="cv",number=3, savePredictions = T),
           preProcess=c("center","scale"), tuneGrid=expand.grid(laplace = 1, usekernel=F, adjust=0))
nb$results
```

```
##   laplace usekernel adjust  Accuracy     Kappa  AccuracySD    KappaSD
## 1       1     FALSE      0 0.7515144 0.5022508 0.009995279 0.02004973
```

```r
table(true=nb$pred[,2],predicted=nb$pred[,1])
```

```
##     predicted
## true   0   1
##    0 481 262
##    1 110 644
```

```r
sum(diag(prop.table(table(true=nb$pred[,2],predicted=nb$pred[,1]))))
```

```
## [1] 0.751503
```

I found through trial and error of these combinations that the accuracy tended to float around 75%, I couldn't get it higher, but this is sill quite good.

Naiive-Bayes: But Manually

The below code has been ran twice, but only included once, this is because setting $usekernel = T$ makes the classifier 2% more accurate.

```r
mNB = naive_bayes(crystalData[,1:25],crystalData[,26], usekernel = T)
predNB <- predict(mNB, testData[,1:25])
real_data <- testData[,26]
table(true=real_data, predicted=predNB)
```

```
##     predicted
## true  0  1
##    0 49 23
##    1 11 61
```

```r
sum(diag(prop.table(table(true=real_data,predicted=predNB))))
```

```
## [1] 0.7638889
```

LDA: with CARET

```r
crystalData$Category <- as.factor(crystalData$Category)
  nb = train(Category~., method="lda", data=crystalData,
             trControl=trainControl(method="cv", number=3,savePredictions = T),
             preProcess=c("center","scale"))
  nb$results
```

```
##   parameter  Accuracy     Kappa AccuracySD  KappaSD
## 1      none 0.7615171 0.5222799 0.01458103 0.028965
```

```r
  table(true=nb$pred[,2],predicted=nb$pred[,1])
```

```
##     predicted
## true   0   1
##    0 487 256
##    1 101 653
```

```r
  sum(diag(prop.table(table(true=nb$pred[,2],predicted=nb$pred[,1]))))
```

```
## [1] 0.761523
```

We see this gives a slightly improved accuracy, which is nice. But I believe it would be best to do this manually as we can specify the number of principal components to use, and hopefully gain more accuracy.

LDA: But Manually

Recall earlier we suggested using 18 principal components, so let us now do that in the manual way, instead of with CARET.

```r
  pcaScores <- crysPCA$x[,1:18]
  crysLDA <- lda(pcaScores, crystalData[,26], CV=T)
  table(crystalData[,26], crysLDA$class)
```

```
##
##       0   1
##   0 699  44
##   1  14 740
```

```r
  sum(diag(prop.table(table(crystalData[,26],crysLDA$class))))
```

```
## [1] 0.9612558
```

So we see this is considerably more accurate. When doing PCA we saw that the data had to be scaled, so it makes sense to check the same here. Given that we already have 96% accuracy, hopefully this will obtain even more accuracy.

```
pcaScores <- crysPCA$x[,1:18]
crysLDA <- lda(pcaScores, crystalData[,26], CV=T, scale=T)
table(crystalData[,26], crysLDA$class)
```

```
##
##       0   1
##   0 699  44
##   1  14 740
```

```
sum(diag(prop.table(table(crystalData[,26],crysLDA$class))))
```

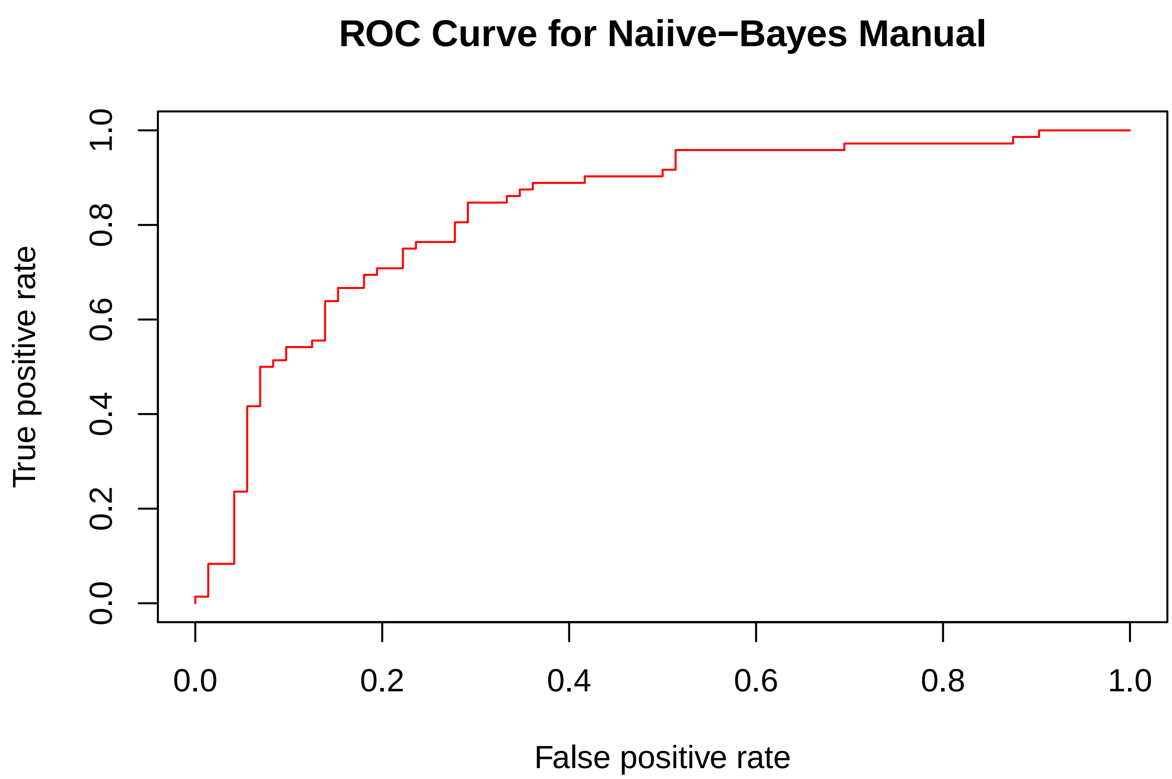
```
## [1] 0.9612558
```

Unexpectedly, we see that scaling makes no difference at all.

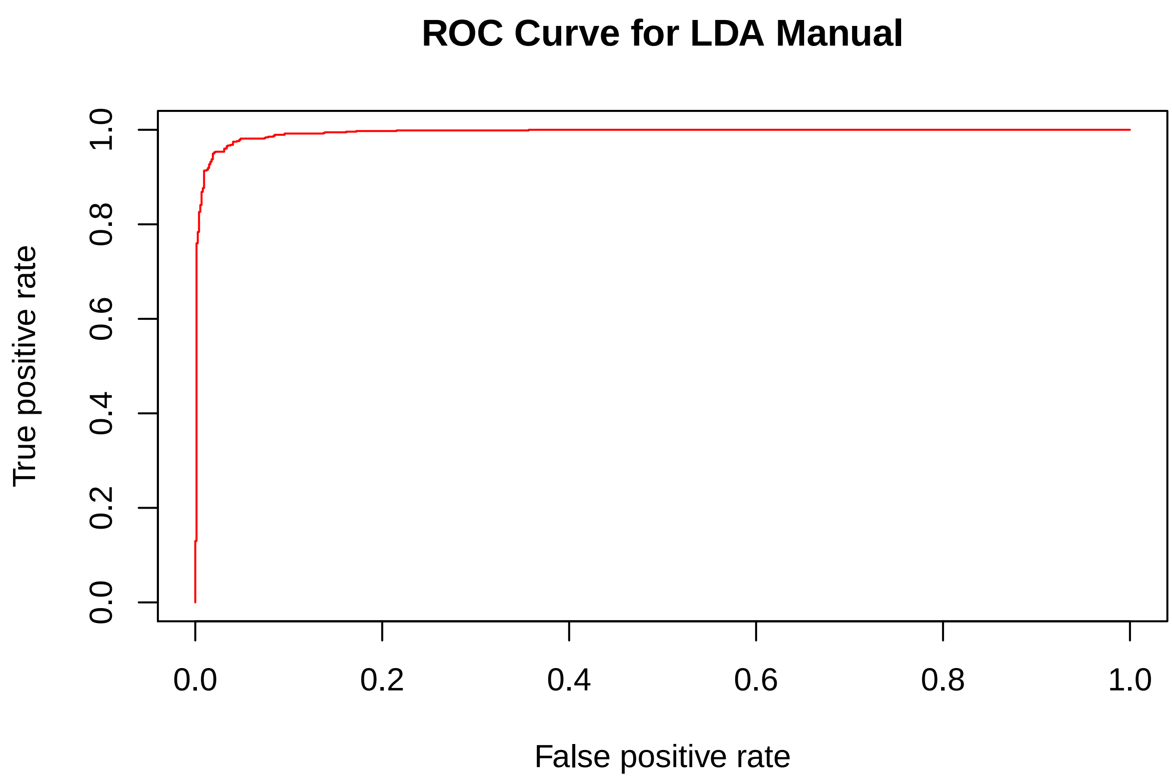
ROC Curves and Concluding Remarks

We have trained two classifiers in two different ways. Doing them both manually instead of with the CARET package turned out to give the best reults. Let's look now at the ROC curves for them.

```
predictNB <- predict(mNB, testData[,1:25], type = "prob")
ratesNB <- prediction(predictNB[,2], testData[,26])
perfNB <- performance(ratesNB, "tpr", "fpr")
plot(perfNB, col="red", main="ROC Curve for Naiive-Bayes Manual")
```



```r
predictLDA <- prediction(crysLDA$posterior[,2], crystalData[,26])
performLDA <- performance(predictLDA, "tpr", "fpr")
plot(performLDA, col="red", main="ROC Curve for LDA Manual")
```



From this graph, the LDA's ROC curve 'hugs' the top right corner more, suggesting this is the better classifier to use.