# Exploration_WFM1

Milton Osiel Candela Leal

7/17/2021

## Packages used

The following packages were used:

```
library(R.matlab)
library(dplyr)
library(caret)
library(MASS)
```

## Reading information

The information is on **.mat** files, and so it must be transformed into a Data Frames via **R.matlab** and **dplyr** packages.

```
data <- readMat('Data/Raw/FeatureMatrices.mat')

unmin <- as.data.frame(data$WFM.1) # Using the first document of 1 minute

feature_names <- unlist(data$FMinfo) # Get column names
colnames(unmin) <- feature_names

unmin <- as.data.frame(unmin)
unmin <- mutate(unmin, Clas = as.factor(Clas)) # Transform the predicted variable to a factor

unmin <- unmin[,c(1:120,length(colnames(unmin)))] # Remove Empatica's variables (EDA TEMP HR HF)

print(head(unmin))
```

```
##   Delta (FP2) Delta (FP1)    Delta (C4)   Delta (C3)   Delta (P8)   Delta (P7)
## 1  0.11643540 -0.10807710 -0.0007967785 -0.021824544 -0.052001271 -0.052366123
## 2  0.08216888  0.09031544 -0.0685819650 -0.025703930 -0.032567698 -0.125933534
## 3  0.13396455  0.03674585  0.1240085263  0.090566646  0.008587199  0.273517209
## 4  0.08577410  0.04645360  0.0868305541  0.028435993  0.065335451  0.215342637
## 5  0.08974345  0.10104639  0.0002082496 -0.008906065  0.001834152  0.004868526
## 6  0.02974833 -0.12397229 -0.0798713713  0.032517547 -0.004881908  0.039584253
##    Delta (O1)  Delta (O2) Theta (FP2) Theta (FP1)   Theta (C4)   Theta (C3)
## 1 -0.14262681 -0.06574524  0.06979229 -0.10023482  0.028478698 -0.092361944
## 2 -0.03696367 -0.03600595  0.07782443  0.02637755 -0.133740130 -0.023413436
## 3  0.19458963  0.16689514  0.13841042  0.05687610  0.143305425  0.113590601
## 4  0.17783968  0.02827166 -0.02372679  0.02989482  0.026600574 -0.001333520
```

```
## 5 -0.07280683 -0.08045784  0.09114603  0.07725917 -0.008292030  0.002163053
## 6  0.02801259 -0.01867924 -0.03278990 -0.24379312 -0.007344334 -0.027462357
##    Theta (P8)  Theta (P7)  Theta (O1)   Theta (O2)  Alpha (FP2)   Alpha (FP1)
## 1 -0.06544173 -0.07803322 -0.08154963 -0.042612373  0.064095119 -1.192098e-01
## 2 -0.05694315 -0.16648136 -0.06419076 -0.013380342 -0.017314926 -1.545495e-03
## 3  0.01735426  0.24849272  0.23423077  0.173923024  0.143913348  6.351154e-02
## 4  0.08405931  0.11828813  0.10736765  0.071394115  0.030066745 -1.902632e-05
## 5 -0.03675589 -0.00619311 -0.04312092 -0.115237194  0.041747758  2.902554e-02
## 6  0.02219570  0.05668009  0.04270302  0.007331088 -0.008774279 -8.920060e-02
##     Alpha (C4)  Alpha (C3)   Alpha (P8)   Alpha (P7)  Alpha (O1)  Alpha (O2)
## 1  0.004581389 -0.01000584 -0.035460413 -0.060126831 -0.09600703 -0.01363575
## 2 -0.045272500 -0.07419167 -0.004620613 -0.065084467 -0.06396609 -0.02152070
## 3  0.205249355  0.10303887  0.040745658  0.165030680  0.27547192  0.12999659
## 4  0.050404753  0.01224851  0.052043003  0.089883492  0.08762201  0.03695271
## 5  0.026771125  0.02443231 -0.021233805 -0.009411168 -0.06158332 -0.04199567
## 6 -0.036117189 -0.02386994  0.019305129  0.039668642 -0.07905597 -0.02377555
##     Beta (FP2)  Beta (FP1)    Beta (C4)    Beta (C3)   Beta (P8)    Beta (P7)
## 1 -0.006055553 -0.09151702 -0.006521960 -0.041936267 -0.02285706 -0.074626616
## 2  0.008422903  0.03103659 -0.069233398 -0.024787664 -0.02198951 -0.056246086
## 3  0.133964159  0.02572089  0.138657713  0.048892236 -0.02480671  0.098044607
## 4 -0.008250105  0.03366403  0.038406550 -0.008405902  0.05658794  0.086350493
## 5 -0.003041084  0.01483961 -0.005814856 -0.006915921  0.01369237 -0.022534921
## 6 -0.100519434 -0.08934101 -0.022341702  0.004814541 -0.01834323  0.001537814
##     Beta (O1)    Beta (O2)  Gamma (FP2) Gamma (FP1)   Gamma (C4)   Gamma (C3)
## 1 -0.06195664 -0.016539408  0.032580386 -0.07589831 -0.000907479 -0.019117830
## 2 -0.06287619 -0.066806878  0.024391976  0.02794868 -0.042955610 -0.021346301
## 3  0.24008555  0.122599988  0.076839674  0.02380477  0.006026729  0.065713828
## 4  0.07967724  0.046137551  0.003852808  0.01239023  0.039379248  0.010437164
## 5 -0.02613204 -0.034037742 -0.004180727 -0.01069456 -0.026747773 -0.052203955
## 6 -0.05457313 -0.006554387 -0.061319071 -0.06863447 -0.032774287 -0.005948348
##     Gamma (P8)   Gamma (P7)   Gamma (O1)   Gamma (O2) d/t (FP2) d/t (FP1)
## 1  0.002288771 -0.022355512 -0.063759173 -0.072860482  1.094513  1.050034
## 2 -0.036451291 -0.075935705 -0.058142849  0.025417432  1.053013  1.125253
## 3 -0.022307056  0.144374762  0.186022865  0.097877342  1.044690  1.039091
## 4  0.001789249  0.097088453  0.113611042  0.020565270  1.166420  1.076298
## 5 -0.019672734  0.008026207 -0.025371014 -0.006619357  1.047438  1.082657
## 6 -0.052534621  0.017598768 -0.008216903  0.020651569  1.116598  1.227107
##    d/t (C4) d/t (C3) d/t (P8) d/t (P7)  d/t (O1) d/t (O2) d/a (FP2) d/a (FP1)
## 1 1.0441860 1.135998 1.073156 1.085699 0.9872727 1.036226  1.150863  1.132103
## 2 1.1556220 1.051608 1.085286 1.107678 1.0883742 1.037533  1.207956  1.220829
## 3 1.0566391 1.032286 1.048825 1.077465 1.0236354 1.055527  1.087368  1.089836
## 4 1.1378361 1.085501 1.039669 1.147967 1.1249086 1.019145  1.156231  1.169928
## 5 1.0839918 1.042437 1.100325 1.068050 1.0247928 1.103626  1.147446  1.196218
## 6 0.9962521 1.119089 1.029917 1.039203 1.0427032 1.034465  1.139538  1.075291
##   d/a (C4) d/a (C3) d/a (P8) d/a (P7) d/a (O1) d/a (O2) d/b (FP2) d/b (FP1)
## 1 1.094563 1.109797 1.055851 1.111904 1.051007 1.029509  1.304028  1.154395
## 2 1.073587 1.182032 1.044111 1.031023 1.140123 1.070841  1.245856  1.243432
## 3 1.026277 1.110506 1.041079 1.205490 1.037883 1.122424  1.160955  1.188466
## 4 1.138616 1.141168 1.087847 1.229744 1.200076 1.077832  1.271021  1.190377
## 5 1.071985 1.086653 1.099592 1.118696 1.094902 1.043293  1.269002  1.275710
## 6 1.050501 1.188089 1.048782 1.102709 1.236989 1.092606  1.329091  1.131113
##   d/b (C4) d/b (C3) d/b (P8) d/b (P7) d/b (O1) d/b (O2) d/g (FP2) d/g (FP1)
## 1 1.127422 1.172853 1.083099 1.186834 1.087593 1.085376  1.331941  1.176806
## 2 1.121747 1.147659 1.104322 1.073377 1.222827 1.180252  1.301380  1.293237
```

```
## 3 1.106541 1.194380 1.154626 1.344161 1.146269 1.187623  1.297250  1.234674
## 4 1.173236 1.191419 1.125640 1.296568 1.298110 1.123029  1.332431  1.260287
## 5 1.127753 1.146436 1.103337 1.191446 1.132895 1.087635  1.348090  1.356977
## 6 1.055000 1.180409 1.131706 1.202981 1.293868 1.128597  1.351414  1.146819
##   d/g (C4) d/g (C3) d/g (P8) d/g (P7) d/g (O1) d/g (O2) t/a (FP2) t/a (FP1)
## 1 1.142469 1.173755 1.080409 1.160558 1.159745 1.228039 1.0514839 1.0781580
## 2 1.111754 1.171762 1.146886 1.132530 1.294905 1.145686 1.1471423 1.0849372
## 3 1.276311 1.204451 1.178377 1.332428 1.275575 1.295298 1.0408529 1.0488361
## 4 1.194494 1.197969 1.214740 1.326369 1.339470 1.227889 0.9912647 1.0869933
## 5 1.173982 1.230771 1.167342 1.193561 1.204790 1.128101 1.0954787 1.1048919
## 6 1.086718 1.222548 1.199733 1.223180 1.312690 1.171724 1.0205444 0.8762815
##    t/a (C4)  t/a (C3)  t/a (P8)  t/a (P7) t/a (O1)  t/a (O2) t/b (FP2)
## 1 1.0482452 0.9769354 0.9838739 1.0241365 1.064556 0.9935175  1.191423
## 2 0.9290124 1.1240237 0.9620605 0.9307966 1.047547 1.0321033  1.183134
## 3 0.9712658 1.0757727 0.9926148 1.1188207 1.013919 1.0633776  1.111292
## 4 1.0006854 1.0512817 1.0463397 1.0712368 1.066821 1.0575851  1.089677
## 5 0.9889239 1.0424156 0.9993337 1.0474189 1.068413 0.9453321  1.211530
## 6 1.0544527 1.0616577 1.0183169 1.0611100 1.186329 1.0562035  1.190303
##   t/b (FP1)  t/b (C4) t/b (C3) t/b (P8)  t/b (P7) t/b (O1)  t/b (O2) t/g (FP2)
## 1 1.0993878 1.0797139 1.032443 1.009265 1.0931516 1.101614 1.0474312  1.216926
## 2 1.1050237 0.9706866 1.091338 1.017540 0.9690332 1.123536 1.1375563  1.235863
## 3 1.1437560 1.0472267 1.157024 1.100876 1.2475218 1.119802 1.1251470  1.241756
## 4 1.1059920 1.0311117 1.097575 1.082691 1.1294470 1.153970 1.1019329  1.142325
## 5 1.1783147 1.0403709 1.099765 1.002738 1.1155342 1.105487 0.9855108  1.287036
## 6 0.9217717 1.0589689 1.054795 1.098832 1.1575994 1.240878 1.0909951  1.210295
##   t/g (FP1)  t/g (C4) t/g (C3) t/g (P8) t/g (P7) t/g (O1) t/g (O2) a/b (FP2)
## 1 1.1207311 1.0941240 1.033237 1.006759 1.068950 1.174696 1.185107  1.133087
## 2 1.1492850 0.9620399 1.114258 1.056759 1.022436 1.189761 1.104241  1.031375
## 3 1.1882257 1.2078965 1.166780 1.123522 1.236633 1.246122 1.227158  1.067674
## 4 1.1709462 1.0497948 1.103609 1.168392 1.155407 1.190737 1.204823  1.099280
## 5 1.2533773 1.0830172 1.180667 1.060907 1.117514 1.175642 1.022176  1.105937
## 6 0.9345714 1.0908066 1.092450 1.164884 1.177037 1.258929 1.132686  1.166342
##   a/b (FP1) a/b (C4)  a/b (C3) a/b (P8) a/b (P7) a/b (O1) a/b (O2) a/g (FP2)
## 1  1.019691 1.030020 1.0568176 1.025808 1.067389 1.034810 1.054266  1.157341
## 2  1.018514 1.044859 0.9709205 1.057667 1.041080 1.072539 1.102173  1.077341
## 3  1.090500 1.078208 1.0755280 1.109067 1.115033 1.104429 1.058088  1.193018
## 4  1.017478 1.030406 1.0440349 1.034741 1.054339 1.081690 1.041933  1.152392
## 5  1.066452 1.052023 1.0550160 1.003406 1.065032 1.034700 1.042502  1.174862
## 6  1.051913 1.004283 0.9935359 1.079067 1.090933 1.045981 1.032940  1.185931
##   a/g (FP1) a/g (C4)  a/g (C3) a/g (P8) a/g (P7) a/g (O1) a/g (O2) b/g (FP2)
## 1  1.039487 1.043767 1.0576303 1.023260 1.043757 1.103461 1.192839  1.021405
## 2  1.059310 1.035551 0.9913115 1.098433 1.098453 1.135759 1.069894  1.044567
## 3  1.132899 1.243631 1.0845968 1.131881 1.105300 1.229016 1.154019  1.117399
## 4  1.077234 1.049076 1.0497743 1.116647 1.078573 1.116154 1.139221  1.048315
## 5  1.134389 1.095147 1.1326259 1.061614 1.066922 1.100363 1.081288  1.062323
## 6  1.066520 1.034477 1.0290035 1.143930 1.109250 1.061197 1.072412  1.016796
##   b/g (FP1)  b/g (C4) b/g (C3)  b/g (P8)  b/g (P7) b/g (O1) b/g (O2) Clas
## 1  1.019414 1.0133463 1.000769 0.9975163 0.9778605 1.066341 1.131441    2
## 2  1.040055 0.9910921 1.021002 1.0385433 1.0551093 1.058944 0.970713    2
## 3  1.038880 1.1534240 1.008432 1.0205704 0.9912714 1.112806 1.090665    2
## 4  1.058729 1.0181193 1.005497 1.0791555 1.0229848 1.031861 1.093372    2
## 5  1.063703 1.0409915 1.073563 1.0580102 1.0017752 1.063461 1.037205    2
## 6  1.013886 1.0300649 1.035698 1.0601103 1.0167909 1.014547 1.038213    2
```

## Information partitioning

We further use the **caret** package in order to divide the whole dataset into a separate training and testing dataset, a seed is used to test with the same division of data.

```
set.seed(1002)

trainIndex <- createDataPartition(unmin$Clas, p = 0.75, list = FALSE)
training <- unmin[trainIndex,]
testing <- unmin[-trainIndex,]
```

## ML models fitting

Data is already normalized, so it does not require pre-processing and can be directly used by ML algorithms. The following function tests a generated model on training and testing data, the parameters are: the ML algorithm and $k$, the number of cross-validations used.

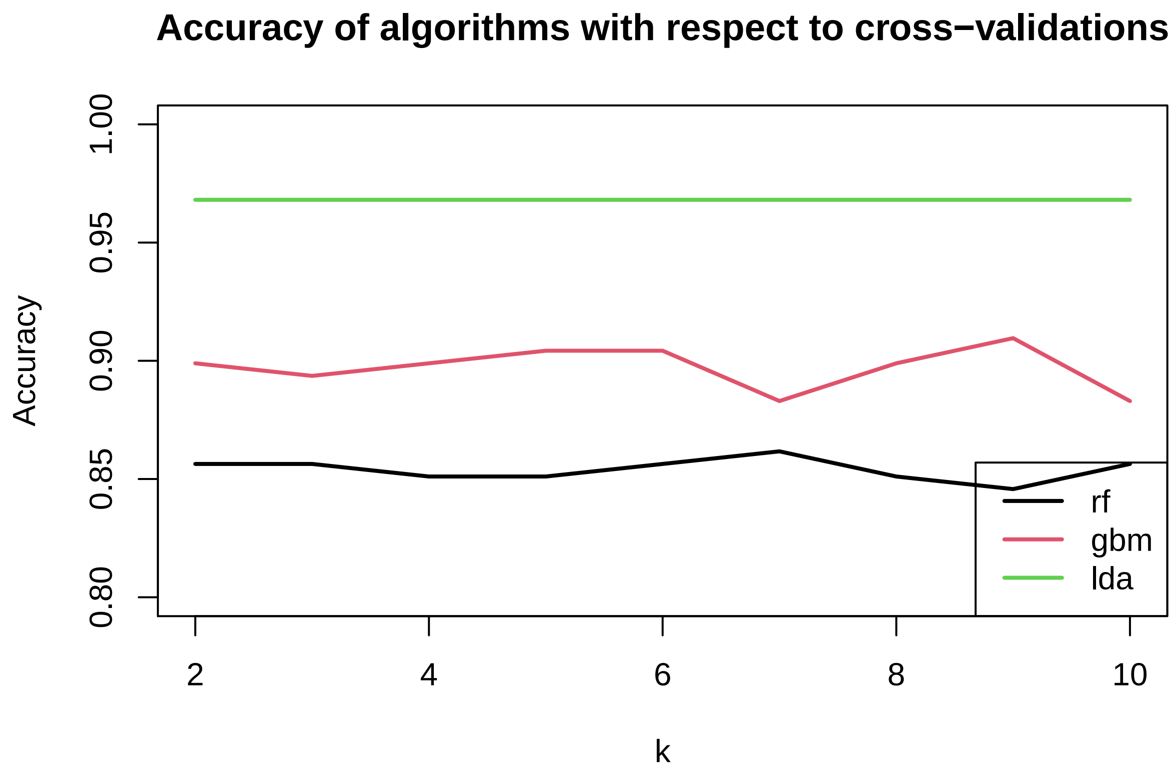
A further for loop iterate three algorithms (rf, gmb, lda) with an incremental number of cross-validations, from 2 to **n_max**.

```
prob_modelo <- function(training, met, testing, control){
     if(met == 'gbm'){
          modelo <- train(Clas ~ ., data = training, method = met,
                          trControl = control, verbose = FALSE)
     } else{
          modelo <- train(Clas ~ ., data = training, method = met,
                          trControl = control)}
     Pred <- predict(modelo, testing)
     acc <- round(confusionMatrix(testing$Clas, Pred)$overall[1], 5)
     return(acc)
}

n_max <- 10 # Maximum number of cross-validations (CVs)
metodos <- c('rf', 'gbm', 'lda')
df <- data.frame('rf' = 2:n_max, 'gbm' = 2:n_max, 'lda' = 2:n_max)

for(metodo in metodos){
        acc_l <- c()
        for(num in 2:n_max){
                control <- trainControl(method = 'cv', number = num)
                acc <- prob_modelo(training, metodo, testing, control)
                acc_l <- c(acc_l, acc) }
        df[,metodo] <- acc_l
}

plot(y = df[,1], x = 2:n_max, ylim = c(0.8, 1), xlim = c(2,10), type = 'l', col = 1, lwd = 2,
     ylab = 'Accuracy', xlab = 'k',
     main = 'Accuracy of algorithms with respect to cross-validations')
lines(y = df[,2], x = 2:n_max, col = 2, lwd = 2)
lines(y = df[,3], x = 2:n_max, col = 3, lwd = 2)
legend('bottomright', legend = c('rf', 'gbm', 'lda'), col = 1:3, lty = 1, lwd = 2)
```




As it can be seen on the plot, the LDA (Linear Discriminant Analysis) model outperformed *rf* and *gbm*, it is also not affected by the number of CVs, so it will be further analyzed.

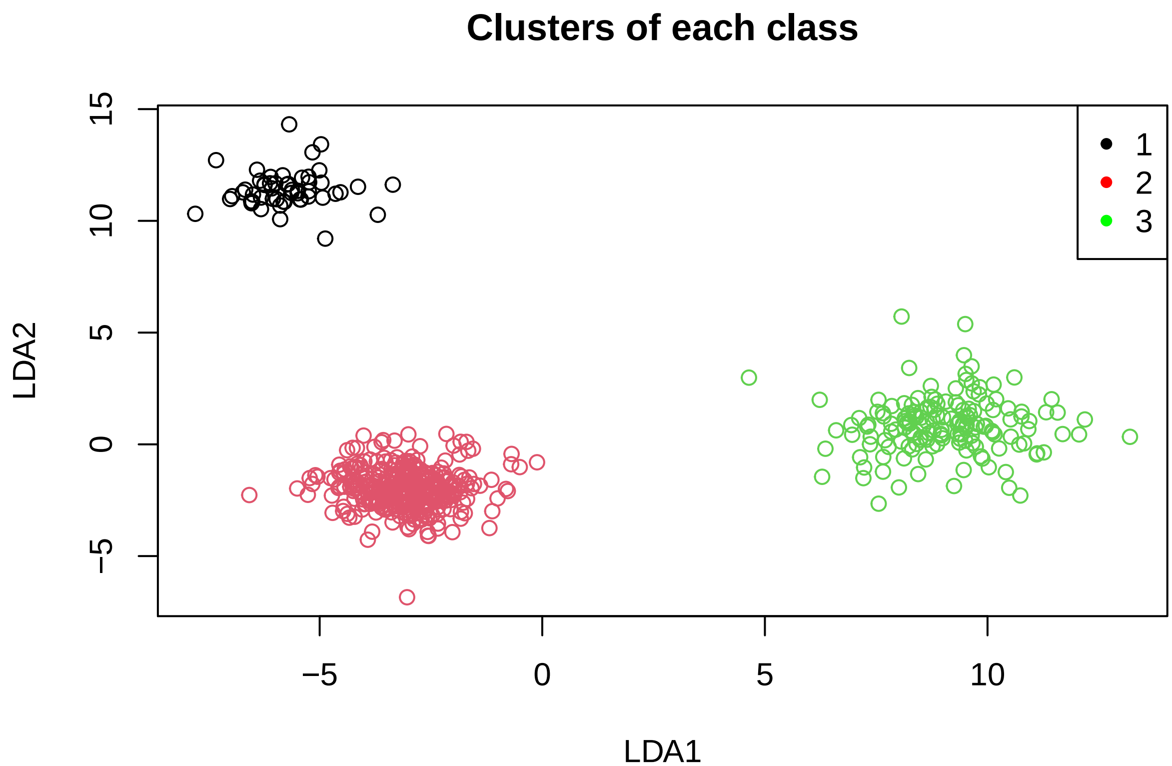
```
modelo_lda <- lda(Clas ~ ., data = training)
```

### Model evaluation

In order to visualize the LDAs predictions, we obtain the two most relevant components (LDA1 and LDA), create a plot of the transformed values and its predicted class on training data.

```
valores <- predict(modelo_lda)

plot(valores$x[,1], valores$x[,2], col = training$Clas, xlab = 'LDA1', ylab = 'LDA2',
     main = 'Clusters of each class')
legend('topright', legend = c(1:3), col = c('black', 'red', 'green'), pch = 20)
```




Using the current weights, the plot on testing data is affected due to an outlier, and thus we compute a confusion matrix on testing data:

```r
pred <- predict(modelo_lda, testing)
print(confusionMatrix(testing$Clas, pred$class))
```

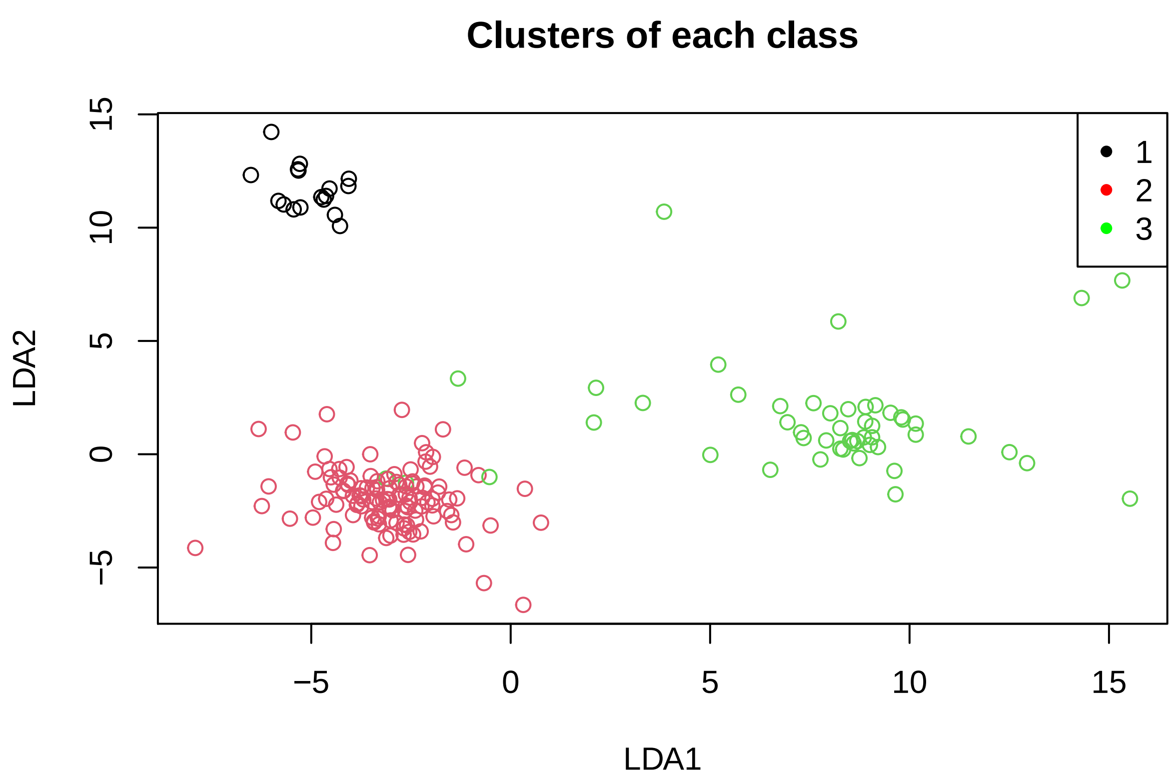
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1  17   0   0
##          2   0 120   0
##          3   1   5  45
##
## Overall Statistics
##
##                Accuracy : 0.9681
##                  95% CI : (0.9318, 0.9882)
##     No Information Rate : 0.6649
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9364
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity           0.94444   0.9600   1.0000
## Specificity           1.00000   1.0000   0.9580
## Pos Pred Value        1.00000   1.0000   0.8824
## Neg Pred Value        0.99415   0.9265   1.0000
## Prevalence            0.09574   0.6649   0.2394
## Detection Rate        0.09043   0.6383   0.2394
## Detection Prevalence  0.09043   0.6383   0.2713
## Balanced Accuracy     0.97222   0.9800   0.9790
```

The outlier is then padded with the median of the predictions and thus the plot can be visualized.

```
outliers <- pred$x[,1] > 1000
pred$x[outliers,] <- c(median(pred$x[,1]), median(pred$x[,2]))

plot(pred$x[,1], pred$x[,2], col = testing$Clas, xlab = 'LDA1', ylab = 'LDA2',
     main = 'Clusters of each class')
legend('topright', legend = c(1:3), col = c('black', 'red', 'green'), pch = 20)
```




### Plotting the outlier

A visualization of the outlier is presented via an histogram of the value of its features.

```
out_df <- testing[outliers,]
barplot(as.matrix(out_df), las = 2, ylab = 'Standarized value', xlab = 'Feature name',
        cex.names = 0.5, main = paste('Value of row number', row.names(out_df)))
```