MVA_Assignment_9

Aman

12/11/2020

Assignment 9 - Discriminant Analysis

This document performs Discriminant Analysis on the Heart Failure Prediction dataset.

We note there are multiple types of Discriminant Analysis and we will explore all of them in this document.

Let us load libraries and data

```r
# clear environment
rm(list = ls())

# defining libraries

library(ggplot2)
library(dplyr)
library(PerformanceAnalytics)
library(data.table)
library(sqldf)
library(nortest)
library(MASS)
library(rpart)
library(class)
library(ISLR)
library(scales)
library(ClustOfVar)
library(GGally)
library(reticulate)
library(ggthemes)
library(RColorBrewer)
library(gridExtra)
library(kableExtra)
library(Hmisc)
library(corrplot)
library(energy)
library(nnet)
library(Hotelling)
library(car)
library(devtools)
library(ggbiplot)
library(factoextra)
library(rgl)
library(FactoMineR)
library(psych)
library(nFactors)
library(scatterplot3d)
library(lmtest)
library(mctest)
library(aod)
library(InformationValue)
library(pROC)
library(tidyverse)
library(caret)
library(Information)
library(mda)
library(klaR)
library(ROCR)
```

```
# reading data
data <- read.csv('/Users/mac/Downloads/heart_failure_clinical_records_dataset.csv')
str(data)
```

```
## 'data.frame':    299 obs. of  13 variables:
##  $ age                     : num  75 55 65 50 65 90 75 60 65 80 ...
##  $ anaemia                 : int  0 0 0 1 1 1 1 1 0 1 ...
##  $ creatinine_phosphokinase: int  582 7861 146 111 160 47 246 315 157 123 ...
##  $ diabetes                : int  0 0 0 0 1 0 0 1 0 0 ...
##  $ ejection_fraction       : int  20 38 20 20 20 40 15 60 65 35 ...
##  $ high_blood_pressure     : int  1 0 0 0 0 1 0 0 0 1 ...
##  $ platelets               : num  265000 263358 162000 210000 327000 ...
##  $ serum_creatinine        : num  1.9 1.1 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 ...
##  $ serum_sodium            : int  130 136 129 137 116 132 137 131 138 133 ...
##  $ sex                     : int  1 1 1 1 0 1 1 1 0 1 ...
##  $ smoking                 : int  0 0 1 0 0 1 0 1 0 1 ...
##  $ time                    : int  4 6 7 7 8 8 10 10 10 10 ...
##  $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
```

Data Cleaning - Let’s remove the outliers

```
data <- data[data$ejection_fraction <70,]
data <- data[data$creatinine_phosphokinase <7000,]
str(data)
```

```
## 'data.frame':    295 obs. of  13 variables:
##  $ age                     : num  75 65 50 65 90 75 60 65 80 75 ...
##  $ anaemia                 : int  0 0 1 1 1 1 1 0 1 1 ...
##  $ creatinine_phosphokinase: int  582 146 111 160 47 246 315 157 123 81 ...
##  $ diabetes                : int  0 0 0 1 0 0 1 0 0 0 ...
##  $ ejection_fraction       : int  20 20 20 20 40 15 60 65 35 38 ...
##  $ high_blood_pressure     : int  1 0 0 0 1 0 0 0 1 1 ...
##  $ platelets               : num  265000 162000 210000 327000 204000 ...
##  $ serum_creatinine        : num  1.9 1.3 1.9 2.7 2.1 1.2 1.1 1.5 9.4 4 ...
##  $ serum_sodium            : int  130 129 137 116 132 137 131 138 133 131 ...
##  $ sex                     : int  1 1 1 0 1 1 1 0 1 1 ...
##  $ smoking                 : int  0 1 0 0 1 0 1 0 1 1 ...
##  $ time                    : int  4 7 7 8 8 10 10 10 10 10 ...
##  $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
```

We remove the 4 outliers before proceeding to modeling exercise.

Split into train (70%), test (30%) and normalize data

```r
set.seed(123)
training.samples <- data$DEATH_EVENT %>%
  createDataPartition(p = 0.7, list = FALSE)
train.data <- data[training.samples, ]
test.data <- data[-training.samples, ]
# Estimate preprocessing parameters
preproc.param <- train.data %>%
  preProcess(method = c("center", "scale"))
# Transform the data using the estimated parameters
train.transformed <- preproc.param %>% predict(train.data)
test.transformed <- preproc.param %>% predict(test.data)
```

We see that our train set has 207 observations, while our test set has 88 observations.

Linear Discriminant Analysis

LDA like PCA finds maximum separation only this time between classes instead of independent variables. The directions are termed as linear discriminants and are the linear combination of independent variables.

Assumptions of LDA

1. Independent variables are normally distributed
2. No outliers and scaling

We saw in EDA exercise when we performed univariate checks that not all our variables are normal with exception of age, serum_sodium. Most other numeric variables are +vely skewed. We can try transformations but first let's perform LDA and analyze results. We already did outlier removal and scaling.

Fitting a model

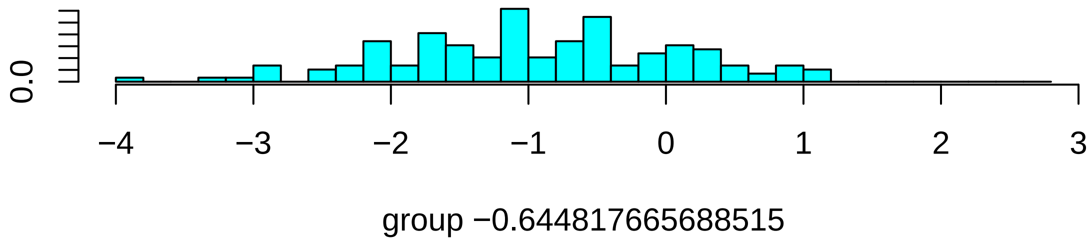
```r
# Fit the model
set.seed(123)
model <- lda(DEATH_EVENT~., data = train.transformed)
model
```

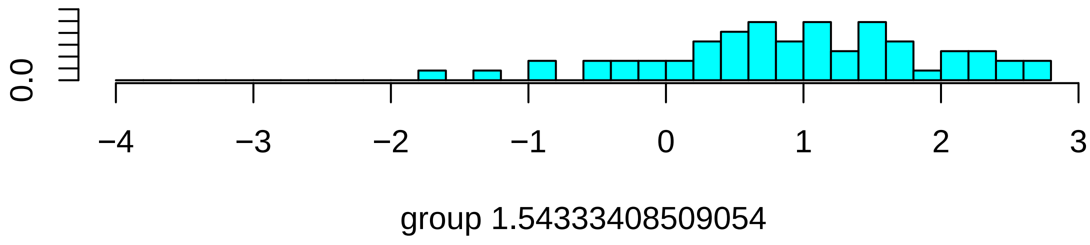
```
## Call:
## lda(DEATH_EVENT ~ ., data = train.transformed)
##
## Prior probabilities of groups:
## -0.644817665688515   1.54333408509054
##           0.705314           0.294686
##
## Group means:
##                           age     anaemia creatinine_phosphokinase
## -0.644817665688515 -0.2193969  0.00152785               0.01710511
## 1.54333408509054    0.5251140 -0.00365682              -0.04094010
##                        diabetes ejection_fraction high_blood_pressure
## -0.644817665688515 -0.04099656         0.1819387         -0.03615552
## 1.54333408509054    0.09812291        -0.4354598          0.08653616
##                       platelets serum_creatinine serum_sodium          sex
## -0.644817665688515  0.05242936       -0.1645340    0.1537749 -0.009070467
## 1.54333408509054   -0.12548667        0.3938026   -0.3680514  0.021709641
##                         smoking       time
## -0.644817665688515 -0.003739125  0.3271218
## 1.54333408509054    0.008949382 -0.7829473
##
## Coefficients of linear discriminants:
##                                   LD1
## age                       0.5287939274
## anaemia                  -0.0003371081
## creatinine_phosphokinase  0.0087822238
## diabetes                  0.1295925527
## ejection_fraction        -0.5198598313
## high_blood_pressure      -0.0258742199
## platelets                -0.0722981492
## serum_creatinine          0.1252698379
## serum_sodium             -0.2707782147
## sex                          -0.1578778018
## smoking                       0.1043772258
## time                         -0.8425348446
```

The outcome is easy enough to interpret. Group means depict the centre of gravity for each variable. We get only one LD1 as our dependent variable is binary.

Let's plot the model

```
plot(model)
```





The first plot is for survival events and the latter plot is for death events. We see above some overlap above -1 till 1.2.

Let's predict

```
# Make predictions
predictions <- model %>% predict(test.transformed)
# Model accuracy
mean(predictions$class==test.transformed$DEATH_EVENT)
```

```
## [1] 0.7727273
```

This base model is 77.2% accurate.

Let's compute accuracies

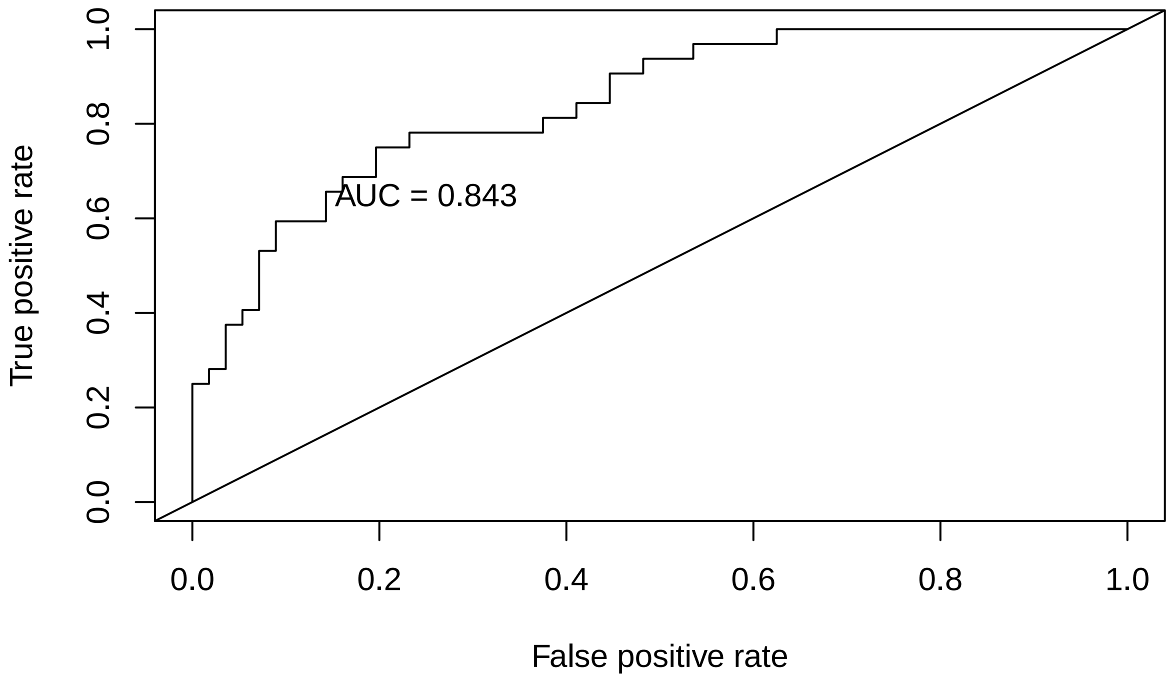
```
confusionMatrix(data = as.factor(predictions$class),
        reference = as.factor(test.transformed$DEATH_EVENT),
      positive='1.54333408509054', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction            -0.644817665688515 1.54333408509054
##   -0.644817665688515                 49               13
##   1.54333408509054                    7               19
##
##                Accuracy : 0.7727
##                  95% CI : (0.6711, 0.8553)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 0.00434
##
##                   Kappa : 0.4884
##
##  Mcnemar's Test P-Value : 0.26355
##
##               Precision : 0.7308
##                  Recall : 0.5938
##                      F1 : 0.6552
##              Prevalence : 0.3636
##          Detection Rate : 0.2159
##    Detection Prevalence : 0.2955
##       Balanced Accuracy : 0.7344
##
##        'Positive' Class : 1.54333408509054
##
```

We see a precision of 0.73 and a recall of 0.59 and an F1-score of 0.65.

Let's compute ROC/AUC

```r
posteriors <- as.data.frame(predictions$posterior)
pred <- prediction(posteriors[,2], test.transformed$DEATH_EVENT)
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train.old <- performance(pred, measure = "auc")
auc.train.old <- auc.train.old@y.values
# Plot
plot(roc.perf)
abline(a=0, b= 1)
text(x = .25, y = .65 ,paste("AUC = ", round(auc.train.old[[1]],3), sep = ""))
```



We get an AUC of 0.843. However, we do note that False positive rates also increase as TPR increases which means we would be telling people with heart failiure risk that they are fine.

Quadratic Discriminant Analysis

QDA doesn't assume equality of variance/covariance. Let's experiment on our data

```
# Fit the model
set.seed(123)
model <- qda(DEATH_EVENT~., data = train.transformed)
model
```

```
## Call:
## qda(DEATH_EVENT ~ ., data = train.transformed)
##
## Prior probabilities of groups:
## -0.644817665688515   1.54333408509054
##           0.705314           0.294686
##
## Group means:
##                           age     anaemia creatinine_phosphokinase
## -0.644817665688515 -0.2193969  0.00152785               0.01710511
## 1.54333408509054    0.5251140 -0.00365682              -0.04094010
##                       diabetes ejection_fraction high_blood_pressure
## -0.644817665688515 -0.04099656         0.1819387         -0.03615552
## 1.54333408509054    0.09812291        -0.4354598          0.08653616
##                      platelets serum_creatinine serum_sodium          sex
## -0.644817665688515  0.05242936       -0.1645340    0.1537749 -0.009070467
## 1.54333408509054   -0.12548667        0.3938026   -0.3680514  0.021709641
##                        smoking       time
## -0.644817665688515 -0.003739125  0.3271218
## 1.54333408509054    0.008949382 -0.7829473
```

```
# Make predictions
predictions <- model %>% predict(test.transformed)
# Model accuracy
mean(predictions$class == test.transformed$DEATH_EVENT)
```

```
## [1] 0.7272727
```

This base model is 72.7% accurate.

In general, QDA works better than LDA for large data.

Let's compute accuracies

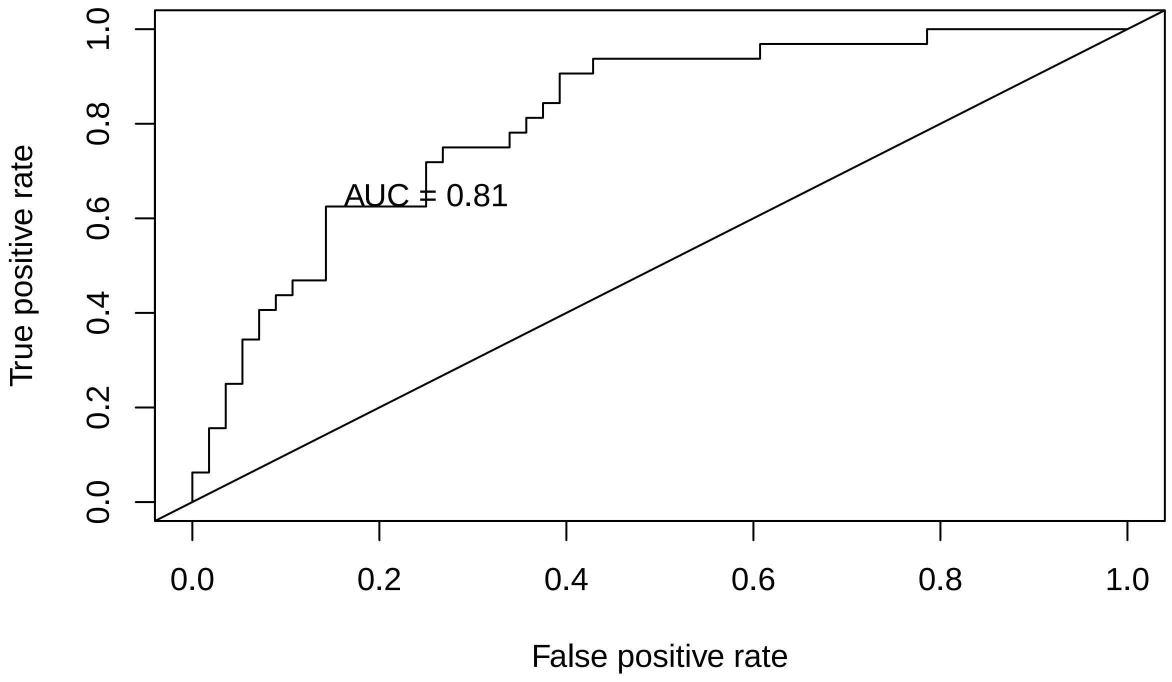
```
confusionMatrix(data = as.factor(predictions$class),
        reference = as.factor(test.transformed$DEATH_EVENT),
        positive='1.54333408509054', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction            -0.644817665688515 1.54333408509054
##   -0.644817665688515                 50               18
##   1.54333408509054                    6               14
##
##                Accuracy : 0.7273
##                  95% CI : (0.6219, 0.8168)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 0.04610
##
##                   Kappa : 0.3592
##
##  Mcnemar's Test P-Value : 0.02474
##
##               Precision : 0.7000
##                  Recall : 0.4375
##                      F1 : 0.5385
##              Prevalence : 0.3636
##          Detection Rate : 0.1591
##    Detection Prevalence : 0.2273
##       Balanced Accuracy : 0.6652
##
##        'Positive' Class : 1.54333408509054
##
```

We see a precision of 0.7 and a recall of 0.43 and an F1-score of 0.53.

Let's compute ROC/AUC

```r
posteriors <- as.data.frame(predictions$posterior)
pred <- prediction(posteriors[,2], test.transformed$DEATH_EVENT)
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
# Plot
plot(roc.perf)
abline(a=0, b= 1)
text(x = .25, y = .65 ,paste("AUC = ", round(auc.train[[1]],3), sep = ""))
```



We get an AUC of 0.81 however we can get a higher TPR for the same increase in FPR from before

Mixed Discriminant Analysis

While LDA assumes, each class comes from normal distribution, while in MDA each class is assumed to be a mixture of subclasses.

```r
# Fit the model
set.seed(123)
model <- mda(DEATH_EVENT~., data = train.transformed)
model
```

```
## Call:
## mda(formula = DEATH_EVENT ~ ., data = train.transformed)
##
## Dimension: 5
##
## Percent Between-Group Variance Explained:
##  v1  v2  v3  v4  v5
## 100 100 100 100 100
##
## Degrees of Freedom (per dimension): 13
##
## Training Misclassification Error: 0.11594 ( N = 207 )
##
## Deviance: 132.932
```

The MDA gives percent between group variance for 5 dimensions. It is easier to think of this as LDA1-5 performed simultaneously.

```r
# Make predictions
predictions <- model %>% predict(test.transformed)
# Model accuracy
mean(predictions == test.transformed$DEATH_EVENT)
```

```
## [1] 0.7840909
```

This base model is 78.4% accurate which is better than both LDA and QDA.

Let's compute accuracies

```r
confusionMatrix(data = as.factor(predictions),
        reference = as.factor(test.transformed$DEATH_EVENT),
        positive='1.54333408509054', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction           -0.644817665688515 1.54333408509054
##   -0.644817665688515                 51               14
##   1.54333408509054                    5               18
##
##                Accuracy : 0.7841
##                  95% CI : (0.6835, 0.8647)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 0.002085
##
```

```
##                   Kappa : 0.5036
##
##  Mcnemar's Test P-Value : 0.066457
##
##               Precision : 0.7826
##                  Recall : 0.5625
##                      F1 : 0.6545
##              Prevalence : 0.3636
##          Detection Rate : 0.2045
##    Detection Prevalence : 0.2614
##       Balanced Accuracy : 0.7366
##
##        'Positive' Class : 1.54333408509054
##
```

We see a precision score of 0.78, recall of 0.56 and F1-score of 0.65

Flexible Discriminant Analysis

FDA is just an extension of LDA for modeling nonlinearities.

```
# Fit the model
set.seed(123)
model <- fda(DEATH_EVENT~., data = train.transformed)
model
```

```
## Call:
## fda(formula = DEATH_EVENT ~ ., data = train.transformed)
##
## Dimension: 1
##
## Percent Between-Group Variance Explained:
##  v1
## 100
##
## Degrees of Freedom (per dimension): 13
##
## Training Misclassification Error: 0.1401 ( N = 207 )
```

FDA produces only one dimension as expected (binary case)

```
# Make predictions
predictions <- model %>% predict(test.transformed)
# Model accuracy
mean(predictions == test.transformed$DEATH_EVENT)
```

```
## [1] 0.7727273
```

This base model is 77.2% accurate which is exactly what we got from LDA as well.

Let's compute accuracies

```
confusionMatrix(data = as.factor(predictions),
        reference = as.factor(test.transformed$DEATH_EVENT),
        positive='1.54333408509054', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##                     Reference
## Prediction            -0.644817665688515 1.54333408509054
##   -0.644817665688515                  49               13
##   1.54333408509054                     7               19
##
##                Accuracy : 0.7727
##                  95% CI : (0.6711, 0.8553)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 0.00434
##
##                   Kappa : 0.4884
##
##  Mcnemar's Test P-Value : 0.26355
##
```

```
##                Precision : 0.7308
##                   Recall : 0.5938
##                       F1 : 0.6552
##               Prevalence : 0.3636
##           Detection Rate : 0.2159
##     Detection Prevalence : 0.2955
##        Balanced Accuracy : 0.7344
##
##         'Positive' Class : 1.54333408509054
##
```

We see a precision score of 0.73, recall of 0.59 and F1-score of 0.65

Regularized Discriminant Analysis

RDA builds classification rules by regularizing group covariance matrices.

```
# Fit the model
set.seed(123)
model <-rda(DEATH_EVENT~., data = train.transformed)
model
```

```
## Call:
## rda(formula = DEATH_EVENT ~ ., data = train.transformed)
##
## Regularization parameters:
##     gamma    lambda
## 0.2029372 0.9777945
##
## Prior probabilities of groups:
## -0.644817665688515   1.54333408509054
##           0.705314           0.294686
##
## Misclassification rate:
##        apparent: 14.493 %
## cross-validated: 17.232 %
```

```
# Make predictions
predictions <- model %>% predict(test.transformed)
# Model accuracy
mean(predictions$class == test.transformed$DEATH_EVENT)
```

```
## [1] 0.7840909
```

This base model is 78.4% accurate which is exactly what we got from MDA as well.

Let's compute accuracies

```
confusionMatrix(data = as.factor(predictions$class),
        reference = as.factor(test.transformed$DEATH_EVENT),
        positive='1.54333408509054', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction           -0.644817665688515 1.54333408509054
##   -0.644817665688515                 49               12
##   1.54333408509054                    7               20
##
##                Accuracy : 0.7841
##                  95% CI : (0.6835, 0.8647)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 0.002085
##
##                   Kappa : 0.5173
##
##  Mcnemar's Test P-Value : 0.358795
##
```

```
##               Precision : 0.7407
##                  Recall : 0.6250
##                      F1 : 0.6780
##              Prevalence : 0.3636
##          Detection Rate : 0.2273
##    Detection Prevalence : 0.3068
##       Balanced Accuracy : 0.7500
##
##        'Positive' Class : 1.54333408509054
##
```

We see a precision of 0.74 and a recall of 0.62 and an F1-score of 0.67.

We found that for a base model MDA, and RDA gave us same accuracies on the test set however RDA gave better F1-score whereas LDA was second best with slightly lower accuracies. However, QDA which assumes different covariance matrices for each class gave us slightly worse results.

We now try an iteration with PCA set being used to perform LDA.

LDA with PCA combined

conduct PCA on training dataset

```
pca <- prcomp(train.transformed[,1:12], retx=TRUE,
              center=TRUE, scale=TRUE)
expl.var <- round(pca$sdev^2/sum(pca$sdev^2)*100)
# percent explained variance
expl.var
```

```
##  [1] 15 13 12 10  9  8  7  7  6  5  5  4
```

The explained variance in components is same as before

prediction of PCs for validation dataset

```
pred <- predict(pca, newdata=test.transformed[,1:12])
head(pred,5)
```

```
##          PC1       PC2         PC3        PC4         PC5         PC6
## 4 -0.8546098 1.2461924  0.04886738 -1.0948007 -0.08455843 -1.08616478
## 5 -1.0107408 3.8959756  2.99373217  0.5603782 -1.64753425 -0.39720766
## 6 -1.4039602 2.3859553 -2.65204260  0.3486357 -0.40635802 -0.06331031
## 7 -1.1592340 1.7418334 -0.70591001 -0.9286161  0.61819339 -0.32142023
## 9  1.2614250 0.8525322 -0.74137074  0.2415185 -0.30453962  1.31240666
##          PC7         PC8       PC9       PC10       PC11       PC12
## 4  1.7410677  0.55202108 0.7359676  0.3994598 -0.5726221 -1.1868597
## 5 -0.4266287  0.89552334 1.8295235 -1.9535535  0.1691180  0.1420112
## 6 -0.2577857 -0.04528154 0.6458834 -0.9100669  0.7857541  0.8137051
## 7  2.0515249 -0.17673113 0.1494599 -0.8975012 -1.2904292 -0.5091571
## 9 -1.2721227 -0.29156118 1.2357032  1.6414888 -1.4955318 -0.2728390
```

Let's create the same sets but with PCA variables added

```
new_data.train <- cbind(train.transformed,pca$x)
new_data.test <- cbind(test.transformed,pred)
```

Let's compute for 7 components -

```
model <- lda(DEATH_EVENT ~ PC1 + PC2 + PC3 + PC4
             + PC5 + PC6 + PC7, data = new_data.train)
model
```

```
## Call:
## lda(DEATH_EVENT ~ PC1 + PC2 + PC3 + PC4 + PC5 + PC6 + PC7, data = new_data.train)
##
## Prior probabilities of groups:
## -0.644817665688515   1.54333408509054
##           0.705314           0.294686
```

```
##
## Group means:
##                             PC1        PC2        PC3        PC4         PC5
## -0.644817665688515  0.2033724 -0.3879932  0.0430567 -0.1602339  0.05106649
## 1.54333408509054   -0.4867602  0.9286394 -0.1030538  0.3835106 -0.12222471
##                              PC6        PC7
## -0.644817665688515  0.03305010 -0.0998100
## 1.54333408509054   -0.07910351  0.2388895
##
## Coefficients of linear discriminants:
##            LD1
## PC1 -0.3665570
## PC2  0.7855377
## PC3 -0.0988275
## PC4  0.4469625
## PC5 -0.1587414
## PC6 -0.1095894
## PC7  0.3673526
```

```
# Make predictions
predictions <- model %>% predict(new_data.test)
# Model accuracy
mean(predictions$class == new_data.test$DEATH_EVENT)
```

```
## [1] 0.8181818
```

We see that with PCA dimensions reduced to 7 from 12, we're able to increase our accuracy to 81.8%

Let's compute accuracies

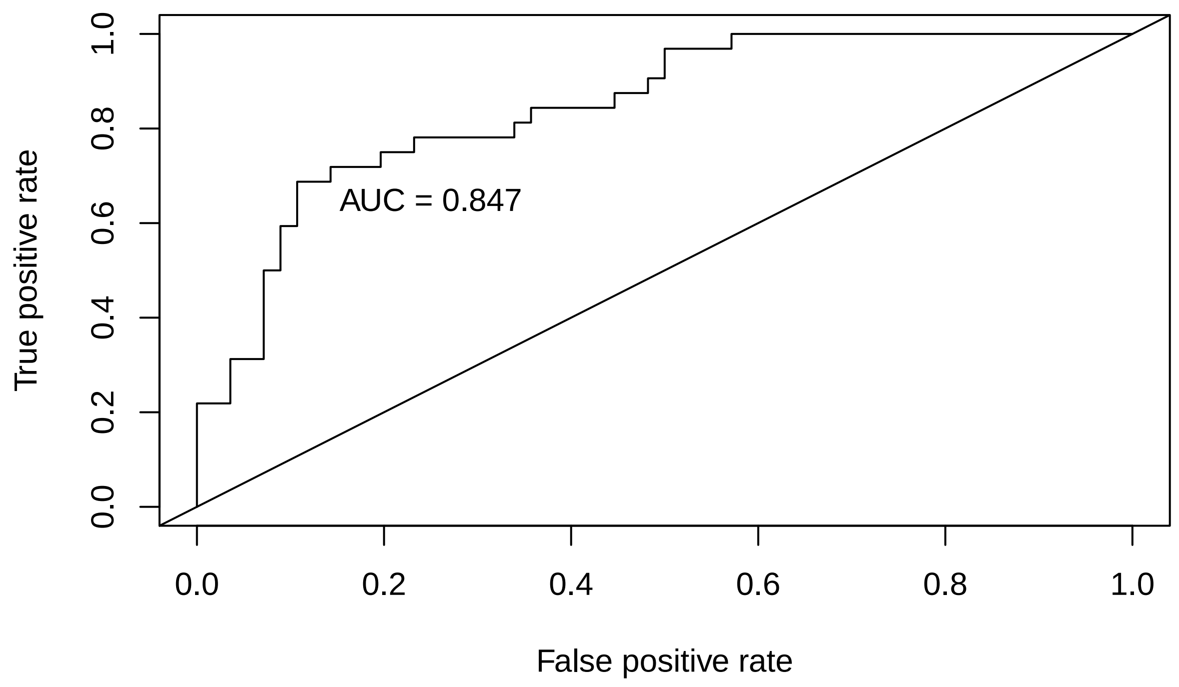
```
confusionMatrix(data = as.factor(predictions$class),
        reference = as.factor(test.transformed$DEATH_EVENT),
        positive='1.54333408509054', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##                       Reference
## Prediction             -0.644817665688515 1.54333408509054
##   -0.644817665688515                   50               10
##   1.54333408509054                      6               22
##
##                Accuracy : 0.8182
##                  95% CI : (0.7216, 0.8924)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 0.0001575
##
##                   Kappa : 0.5963
##
##  Mcnemar's Test P-Value : 0.4532547
##
##               Precision : 0.7857
##                  Recall : 0.6875
##                      F1 : 0.7333
##              Prevalence : 0.3636
##          Detection Rate : 0.2500
##    Detection Prevalence : 0.3182
##       Balanced Accuracy : 0.7902
##
##        'Positive' Class : 1.54333408509054
##
```

We see a precision of 0.78 and a recall of 0.68 and an F1-score of 0.73. We see this is even better than before.

Let's compute ROC/AUC

```
posteriors <- as.data.frame(predictions$posterior)
pred <- prediction(posteriors[,2], new_data.test$DEATH_EVENT)
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train.new <- performance(pred, measure = "auc")
auc.train.new <- auc.train.new@y.values
# Plot
plot(roc.perf)
abline(a=0, b= 1)
text(x = .25, y = .65 ,paste("AUC = ", round(auc.train.new[[1]],3), sep = ""))
```



Our new AUC is 0.847 which is good and marginally better than non-PCA set (0.843).

Trying LDA on WOE dataset as before

Computing IV

```
library(Information)
library(gridExtra)
data <- read.csv('/Users/mac/Downloads/heart_failure_clinical_records_dataset.csv')
data <- data[data$ejection_fraction <70,]
data <- data[data$creatinine_phosphokinase <7000,]
data$anaemia <- factor(data$anaemia)
data$diabetes <- factor(data$diabetes)
data$high_blood_pressure <- factor(data$high_blood_pressure)
data$sex <- factor(data$sex)
data$smoking <- factor(data$smoking)
# this package needs the dependent variable in numeric format
# hence we reload data here
IV <- create_infotables(data=data, y="DEATH_EVENT",
                 bins=10, parallel=FALSE)
IV_Value = data.frame(IV$Summary)
IV$Summary
```

```
##                     Variable           IV
## 12                      time 1.840224e+00
## 5          ejection_fraction 9.763676e-01
## 8            serum_creatinine 9.235629e-01
## 1                        age 4.849249e-01
## 9               serum_sodium 4.030774e-01
## 3   creatinine_phosphokinase 2.157046e-01
## 7                  platelets 1.132326e-01
## 6        high_blood_pressure 2.194425e-02
## 2                    anaemia 2.158339e-02
## 10                       sex 3.028463e-04
## 11                   smoking 7.862779e-05
## 4                   diabetes 8.479320e-06
```

Replacing WOE

```
library(fuzzyjoin)
woe_replace <- function(df_orig, IV) {
  df <- cbind(df_orig)
  df_clmtyp <- data.frame(clmtyp = sapply(df, class))
  df_col_typ <-
    data.frame(clmnm = colnames(df), clmtyp = df_clmtyp$clmtyp)
  for (rownm in 1:nrow(df_col_typ)) {
    colmn_nm <- toString(df_col_typ[rownm, "clmnm"])
    if(colmn_nm %in% names(IV$Tables)){
    column_woe_df <- cbind(data.frame(IV$Tables[[toString(df_col_typ[rownm, "clmnm"])]]))
    if (df_col_typ[rownm, "clmtyp"] == "factor" | df_col_typ[rownm, "clmtyp"] == "character") {
      df <-
        dplyr::inner_join(
          df,
          column_woe_df[,c(colmn_nm,"WOE")],
```

```r
          by = colmn_nm,
          type = "inner",
          match = "all"
        )
      df[colmn_nm]<-NULL
      colnames(df)[colnames(df)=="WOE"]<-colmn_nm
    } else if (df_col_typ[rownm, "clmtyp"] == "numeric" | df_col_typ[rownm, "clmtyp"] == "integer") {
      column_woe_df$lv<-as.numeric(str_sub(
        column_woe_df[,colmn_nm],
        regexpr("\\[", column_woe_df[,colmn_nm]) + 1,
        regexpr(",", column_woe_df[,colmn_nm]) - 1
      ))
      column_woe_df$uv<-as.numeric(str_sub(
        column_woe_df[,colmn_nm],
        regexpr(",", column_woe_df[,colmn_nm]) + 1,
        regexpr("\\]", column_woe_df[,colmn_nm]) - 1
      ))
      column_woe_df[colmn_nm]<-NULL
      column_woe_df<-column_woe_df[,c("lv","uv","WOE")]
      colnames(df)[colnames(df)==colmn_nm]<-"WOE_temp2381111111111111697"
      df <-
        fuzzy_inner_join(
          df,
          column_woe_df[,c("lv","uv","WOE")],
          by = c("WOE_temp2381111111111111697"="lv","WOE_temp2381111111111111697"="uv"),
          match_fun=list(`>=`,`<=`)
        )
      df["WOE_temp2381111111111111697"]<-NULL
      df["lv"]<-NULL
      df["uv"]<-NULL
      colnames(df)[colnames(df)=="WOE"]<-colmn_nm
    }}
  }
  return(df)
}
df_woe <- woe_replace(data, IV)
str(df_woe)
```

```
## 'data.frame':    295 obs. of  13 variables:
##  $ DEATH_EVENT             : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ age                     : num  0.3879 0.00248 -0.32294 0.00248 1.32221 ...
##  $ anaemia                 : num  -0.132 -0.132 0.163 0.163 0.163 ...
##  $ creatinine_phosphokinase: num  0.265 0.651 0.401 0.651 -0.659 ...
##  $ diabetes                : num  0.00248 0.00248 0.00248 -0.00342 0.00248 ...
##  $ ejection_fraction       : num  2.67 2.67 2.67 2.67 -1.33 ...
##  $ high_blood_pressure     : num  0.197 -0.112 -0.112 -0.112 0.197 ...
##  $ platelets               : num  -0.4565 0.0825 0.4274 0.3161 0.4274 ...
##  $ serum_creatinine        : num  1.641 0.157 1.641 1.206 1.206 ...
##  $ serum_sodium            : num  0.999 0.999 -0.834 0.999 0.439 ...
##  $ sex                     : num  -0.0128 -0.0128 -0.0128 0.0237 -0.0128 ...
##  $ smoking                 : num  0.00615 -0.01279 0.00615 0.00615 -0.01279 ...
##  $ time                    : num  2.94 2.94 2.94 2.94 2.94 ...
```

Let's now use the new dataframe for prediction.

Splitting into train and test - 70%, 30% split

```
set.seed(123)
trainIndex <- createDataPartition(df_woe$DEATH_EVENT, p = .7,
                                  list = FALSE,
                                  times = 1)
train_data<-df_woe[trainIndex,]
test_data<-df_woe[-trainIndex,]
```

lda on data

```
model <- lda(DEATH_EVENT ~ ., data = train_data)
model
```

```
## Call:
## lda(DEATH_EVENT ~ ., data = train_data)
##
## Prior probabilities of groups:
##        0        1
## 0.705314 0.294686
##
## Group means:
##          age       anaemia creatinine_phosphokinase      diabetes
## 0 -0.1898510 -0.0005872665               -0.1020196  0.0001749998
## 1  0.3887369 -0.0013510348                0.1663468 -0.0002297532
##   ejection_fraction high_blood_pressure   platelets serum_creatinine
## 0        -0.3869069        -0.022869324 -0.04474466       -0.4603820
## 1         0.5798594        -0.005407226  0.05294180        0.5766226
##   serum_sodium           sex       smoking       time
## 0   -0.2134232 -0.0002995398  5.182266e-05 -0.5926379
## 1    0.2838168 -0.0008318841 -6.087346e-05  1.2415213
##
## Coefficients of linear discriminants:
##                                   LD1
## age                       0.329014438
## anaemia                   0.598581875
## creatinine_phosphokinase  0.647533005
## diabetes                 15.275373684
## ejection_fraction         0.532850445
## high_blood_pressure      -0.330177898
## platelets                 0.009546738
## serum_creatinine          0.259068147
## serum_sodium              0.546844940
## sex                       7.002993484
## smoking                 -12.216711924
## time                      0.669753411
```

```
# Make predictions
predictions <- model %>% predict(test_data)
# Model accuracy
mean(predictions$class == test_data$DEATH_EVENT)
```

```
## [1] 0.8409091
```

This model gives us an accuracy of 84.0%

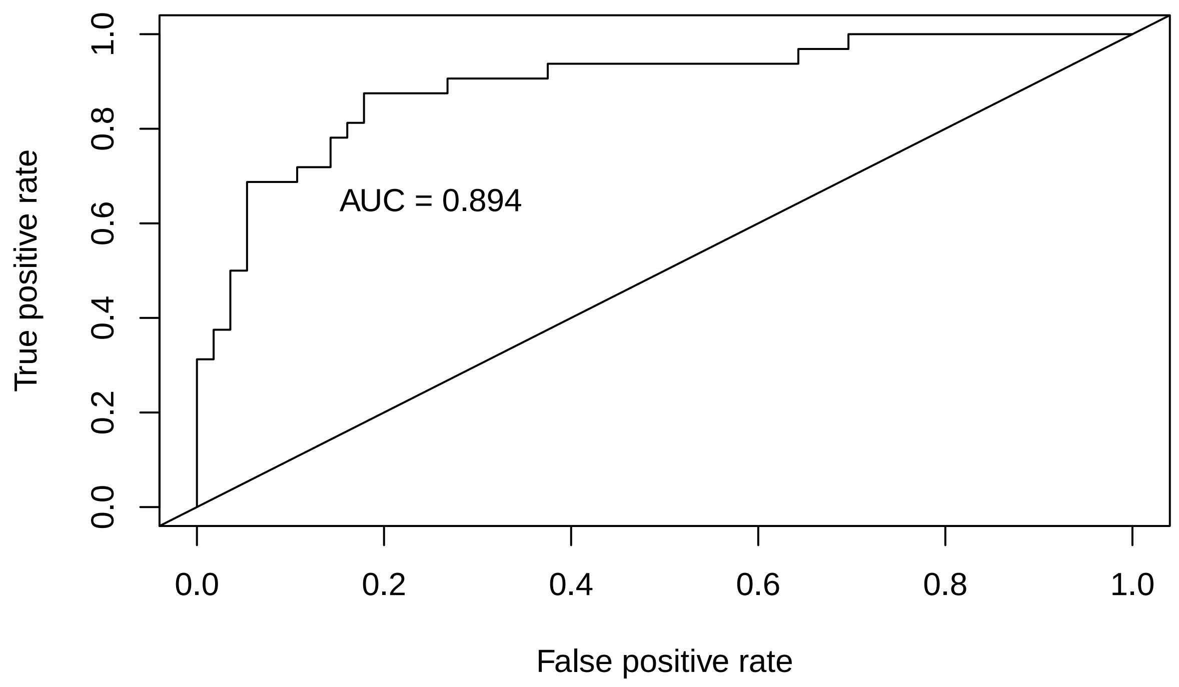
```r
confusionMatrix(data = as.factor(predictions$class),
        reference = as.factor(test_data$DEATH_EVENT),
        positive='1', mode = "prec_recall")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 52 10
##          1  4 22
##
##                Accuracy : 0.8409
##                  95% CI : (0.7475, 0.9102)
##     No Information Rate : 0.6364
##     P-Value [Acc > NIR] : 1.977e-05
##
##                   Kappa : 0.6419
##
##  Mcnemar's Test P-Value : 0.1814
##
##               Precision : 0.8462
##                  Recall : 0.6875
##                      F1 : 0.7586
##              Prevalence : 0.3636
##          Detection Rate : 0.2500
##    Detection Prevalence : 0.2955
##       Balanced Accuracy : 0.8080
##
##        'Positive' Class : 1
##
```

We see a precision of 0.84 and a recall of 0.68 and an F1-score of 0.75. We see this is again marginally better than our previous iteration.

Let's compute ROC/AUC

```r
posteriors <- as.data.frame(predictions$posterior)
pred <- prediction(posteriors[,2], test_data$DEATH_EVENT)
roc.perf = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train.new <- performance(pred, measure = "auc")
auc.train.new <- auc.train.new@y.values
# Plot
plot(roc.perf)
abline(a=0, b= 1)
text(x = .25, y = .65 ,paste("AUC = ", round(auc.train.new[[1]],3), sep = ""))
```



We obtain our highest AUC yet of 0.89

Summarizing all model results in a table

Model	Type	Accuracy	Precision	Recall	F1-Score	Comments
Model 1	LDA	0.772	0.73	0.59	0.65	Linear
Model 2	QDA	0.727	0.70	0.43	0.53	Quadratic
Model 3	MDA	0.784	0.78	0.56	0.65	Mixed
Model 4	FDA	0.772	0.73	0.59	0.65	Flexible
Model 5	RDA	0.784	0.74	0.62	0.67	Regularized
Model 6	LDA with PCA	0.818	0.78	0.68	0.73	PCA of 7 components
Model 7	LDA on WoE	0.841	0.84	0.69	0.76	WoE dataset

Once again, we see that WoE dataset has best accuracies with LDA and even LDA with PCA works well. Among the base models, we saw MDA and RDA outperform LDA and FDA with QDA being the wrong model for this data.

This concludes our approach to Discriminant Analysis in our dataset