LDA CLASSIFICATION

-

Dataset

Il dataset è stato scaricato da UCI: Machine learning repository, il dataset originale contiene 76 attributi, ma tutti gli esperimenti pubblicati fanno riferimento all'utilizzo di un sottoinsieme di 14 di essi. È possibile scaricare 4 possibili database: Cleveland, Ungheria, Svizzera e VA Long Beach. In questa trattazione verrà utilizzato il database Cleveland, poichè è l'unico che è stato utilizzato dai ricercatori ML fino ad oggi. Le unità (pazienti) presenti nel dataset sono 303 e le variabili considerate sono le seguenti:

- *age*: età del paziente
- *sex*: sesso (1 = uomo, 0= donna)
- *cp*: tipo di dolore toracico (valore 1: angina tipica; 2:angina atipica; 3: dolore non toracico; 4: asintomatico)
- *trestbps*: pressione arteriosa a riposo (in mm/Hg al momento del ricovero in ospedale)
- *chol*: colestorale sierico in mg/dl
- *fbs*: glicemia a digiuno (se > 120 mg/dl ci sarà l'etichetta 1, altrimenti 0)
- *restecg*: risultati elettrocardiografici a riposo (0: valore normale; 1: anormalità dell'onda ST-T; 2: probabile ipertrofia ventricolare sinistra)
- *thalach*: frequenza cardiaca massima raggiunta
- *exang*: angina indotta da esercizio (1 = vero; 0 = falso)
- *oldpeak*: sottoslivellamento ST indotto dall'esercizio rispetto al riposo
- *slope*: la pendenza del picco del segmento ST durante l'esercizio (Valore 1: ascendente; Valore 2: piatto; Valore 3: discendente)
- *ca*: numero di vasi maggiori (0-3) colorati da fluoroscopia
- *thal*: 3 = normale; 6 = difetto risolto; 7 = difetto reversibile

L'obiettivo è prevedere la presenza di malattie cardiache nel paziente. Nel dataset la variabile risposta è denominata *num* ed è un valore intero da 0 (nessuna presenza) a 4. L'obiettivo può riguardare anche semplicemente distinguere la presenza (valori 1,2,3,4) dall'assenza (valore 0).

Prima di procedere con l'applicazione degli algoritmi di classificazione, effettuo una breve analisi esplorativa.

Library

```
library(plyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(psych)
library(mlbench)
library(caret)
```

```
## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha

## Loading required package: lattice
```

```
library(AppliedPredictiveModeling)
library(MASS)
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(ROCR)
library(ggord)
```

```
## Warning: package 'ggord' was built under R version 4.2.2
```

Analisi esplorativa sul dataset Carichiamo il dataset:

```
df <- read.csv("~/Desktop/statistical learning/heart_cleveland.csv", sep=";")
head(df)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal num
## 1  63   1  1      145  233   1       2     150     0     2.3     3  0    6   0
## 2  67   1  4      160  286   0       2     108     1     1.5     2  3    3   2
## 3  67   1  4      120  229   0       2     129     1     2.6     2  2    7   1
## 4  37   1  3      130  250   0       0     187     0     3.5     3  0    3   0
## 5  41   0  2      130  204   0       2     172     0     1.4     1  0    3   0
## 6  56   1  2      120  236   0       0     178     0     0.8     1  0    3   0
```

```
str(df)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : num  6 3 7 3 3 3 3 3 7 7 ...
##  $ num     : int  0 2 1 0 0 0 3 0 2 1 ...
```

```
sum(is.na(df))
```

```
## [1] 6
```

Sono presenti 6 dati mancanti, contenute nelle variabili ca e thal.

```
names(which(colSums(is.na(df))>0))
```

```
## [1] "ca"   "thal"
```

```
table(df$ca)
```

```
##
##   0   1   2   3
## 176  65  38  20
```

Le variabili contenenti dati mancanti sono discrete. Nel caso della variabile *ca* essa può assumere valore: 0,1,2,3 e 4. La frequenza maggiore riguarda il valore 0 (a 176 pazienti non si è colorato nessun vaso in seguito alla fluoroscopia). Quattro pazienti non hanno nessun valore, si presume dunque che ancora non siano stati sottoposti all'esame radiologico della fluoroscopia. Sostituire i valori NA con il valore 0, non sembra un'ottima soluzione in quanto 0 indica che nessun vaso si è colorato. Per la variabile *thal* si potrebbe fare un discorso analogo. Per tale ragione, anche se perdiamo informazioni, eliminiamo queste osservazioni contenenti dati mancanti.

```
df <- na.omit(df)
sum(is.na(df))
```

```
## [1] 0
```

Infine, abbiamo detto che possiamo trattare la variabile risposta come dicotomica (1= presenza di malattie cardiache; 0= assenza). Quindi, creiamo una nuova variabile (colonna) in cui i valori 1,2,3,4 della variabile *num* sono sostituiti con il valore 1.

```
df_10<-df  %>%
   mutate(Class = case_when(num ==1 ~ 1,
                            num ==2 ~ 1,
                            num ==3 ~ 1,
                            num ==4 ~ 1,
                            TRUE ~ 0))
```

Adesso nel dataframe df_10 abbiamo 15 variabili, è presente la variabile Class appena creata. Nel dataset denominato "df_10" eliminiamo la colonna 14 (*num*).

```
df_10<-df_10[,-14]
head(df_10)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
## 1  63   1  1      145  233   1       2     150     0     2.3     3  0    6
## 2  67   1  4      160  286   0       2     108     1     1.5     2  3    3
## 3  67   1  4      120  229   0       2     129     1     2.6     2  2    7
## 4  37   1  3      130  250   0       0     187     0     3.5     3  0    3
## 5  41   0  2      130  204   0       2     172     0     1.4     1  0    3
## 6  56   1  2      120  236   0       0     178     0     0.8     1  0    3
##   Class
## 1     0
## 2     1
## 3     1
## 4     0
## 5     0
## 6     0
```

Dividiamo il dataset df_10 (dove la variabile class contiene solamente due classi) in training set e test set.

```
set.seed(1998)
training.samples = createDataPartition(df_10$Class, p = .8, list = FALSE)
train.data <- df_10[training.samples, ]
test.data <- df_10[-training.samples, ]
```

Il training set contiene 238 osservazioni e il test set 59.

LDA

Ricordiamo che l'analisi discriminante può avvenire secondo due approcci: geometrico (ADL di Fisher) o probabilistico. Nell'analisi discriminante di Fisher si ricerca una combinazione lineare delle variabili esplicative con l'obiettivo di massimizzare la separazione dei gruppi e ottenere una nuova variabile detta *variabile*

canonica (sulla quale verranno proiettate le osservazioni ed i rispettivi baricentri dei gruppi). Il numero massimo di variabili canoniche che si possono ottenere è dato dal min(p, G-1). Nel nostro caso p=13 e G=2 (in df_10). Quindi si potrà ottenere solamente una variabile canonica. Infine, l'osservazione verrà assegnata al gruppo la cui media risulta più vicina all'osservazione. Nell'approccio probabilistico, ogni osservazione verrà assegnata alla classe con la più elevata probabiltà a posteriori (calcolato considerando la probabilità a priori di appartenere ad una popolazione).

```
heart_lda<-lda(Class ~ ., data= train.data)
heart_lda
```

```
## Call:
## lda(Class ~ ., data = train.data)
##
## Prior probabilities of groups:
##         0         1
## 0.5420168 0.4579832
##
## Group means:
##        age       sex       cp trestbps     chol       fbs   restecg  thalach
## 0 52.82946 0.5813953 2.829457 130.0388 241.2248 0.1550388 0.8372093 158.3488
## 1 56.76147 0.8073394 3.550459 135.6239 252.8991 0.1376147 1.1467890 139.0367
##       exang  oldpeak    slope        ca     thal
## 0 0.1472868 0.620155 1.457364 0.3100775 3.821705
## 1 0.5229358 1.591743 1.825688 1.0917431 5.935780
##
## Coefficients of linear discriminants:
##                   LD1
## age      -0.005494066
## sex       0.552216442
## cp        0.304800253
## trestbps  0.010812780
## chol      0.003530355
## fbs      -0.576682439
## restecg   0.152395717
## thalach  -0.013564515
## exang     0.451599670
## oldpeak   0.127793894
## slope     0.154696393
## ca        0.485386898
## thal      0.286266164
```

Per eseguire l'LDA usiamo la funzione lda() del package MASS. Nella formula specifichiamo la variabile risposta categoriale (Class) ed i predittori da usare, in data indichiamo il dataset contenente i dati.

Nell'output ottenuto:

- *Prior probabilities of groups*: indica la specificazione delle probabilità a priori (π_1,π_2). Non essendo specificata nella formula, verrà calcolata come frequenza relativa delle classi nel training.

```
table(train.data$Class)
```

```
##
##   0   1
## 129 109
```

$$\pi_1 = 109/238 = 0.4579832$$

Indica la probabilità a priori, che un valore osservato provenga dalla popolazione 1 (e sia etichettato come 1 <- presenza di malattie cardiache).

$$\pi_2 = 129/238 = 0.5420168$$

Indica la probabilità a priori, che un valore osservato provenga dalla popolazione 2 (e sia etichettato come 0 <- assenza di malattie cardiache)

- *Group means*: abbiamo la media di ogni variabile in ogni gruppo. Ad esempio gli individui con malattie cardiache hanno una media di 56 anni rispetto ai 54 nel gruppo etichettato come 0.
- *Coefficients of linear discriminants*: questi sono gli autovettori (α). È presente una sola variabile canonica perchè i gruppi sono due. Quindi la mia variabile canonica sarà data dalla combinazione lineare delle x, come segue:

$$LD1 = (-0.005494066 * age) + (0.552216442 * sex) + ... + (0.286266164 * thal)$$

Adesso visualizzo la performance sul training set, utilizzando il comando predict.

```
prev_train_lda<-predict(heart_lda, train.data)
prev_train_lda$class[10:20]
```

```
##  [1] 0 1 0 0 0 0 0 0 1 1 0
## Levels: 0 1
```

Ad esempio se prendo i soggetti dal decimo al ventesimo, notiamo che ai pazienti 11, 18 e 19 è stata prevista la presenza di malattie cardiache. Per vedere la probabilità a posteriori richiamo $posterior.

```
prev_train_lda$posterior[10:20,]
```

```
##              0          1
## 11 0.529626369 0.47037363
## 13 0.248025685 0.75197431
## 14 0.846034656 0.15396534
## 15 0.771740878 0.22825912
## 16 0.961774687 0.03822531
## 19 0.965099762 0.03490024
## 20 0.975373625 0.02462638
## 23 0.908055136 0.09194486
## 24 0.094856191 0.90514381
## 25 0.007437219 0.99256278
## 26 0.978556643 0.02144336
```

L'11esimo soggetto è stato classificato come 1 con la probabilità a posteriori del 75.2%. Il 13esimo paziente ha invece una probabilità di appartenere al gruppo 0 dell'84.6. Va specificato che in questo caso la threshold è posta a 0.5, quindi un soggetto con una probabilità superiore alla soglia verrà etichettato come 1. Credo, che in questo contesto alzare la soglia non ha molto senso, poichè è preferibile effettuare ulteriori esami ipotizzando la presenza di malattie cardiache piuttosto che non approfondire.

Adesso, il modello stimato lo utilizziamo sul test set.

```
prev_test_lda<-predict(heart_lda, test.data, type="prob")
head(prev_test_lda$class)
```

```
## [1] 1 0 0 0 0 0
## Levels: 0 1
```

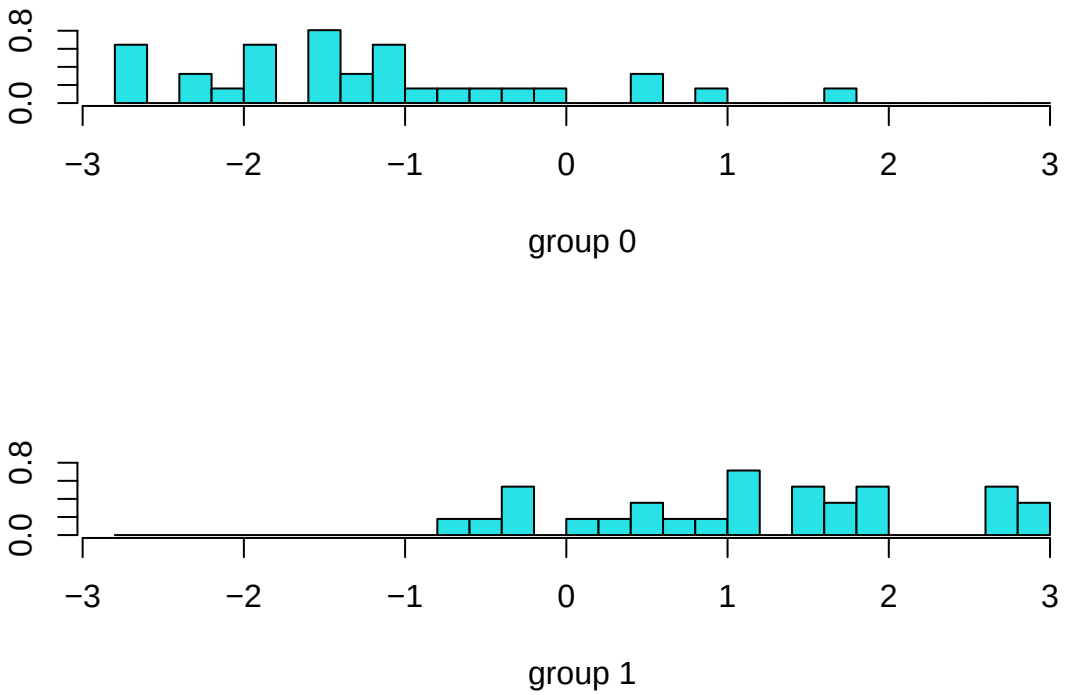
Anche in questo caso vediamo come sono stati classificati i primi elementi e osserviamo che per il primo soggetto appartenente al test set si prevede la presenza di malattie cardiache, con una probabilità del 99%.

```
head(prev_test_lda$posterior)
```

```
##              0           1
## 2  0.005814078 0.994185922
## 12 0.944159836 0.055840164
## 17 0.763853820 0.236146180
## 18 0.877871284 0.122128716
## 21 0.913949077 0.086050923
## 22 0.994483216 0.005516784
```

Attraverso l'istagramma in pila è possibile verificare la separazione tra i gruppi usando LD1.

```
ldahist(data = prev_test_lda$x[,1], g = test.data$Class)
```



Notiamo che i due gruppi sono un pò separati ma tendono ad esserci delle sovrapposizioni.

CONFUSION MATRIX A questo punto posso costruire la matrice di confusione, con il comando confusionMatrix del pacchetto caret. Nella matrice di confusione, effettuaiamo il confronto tra i valori previsti e quelli reali.

```r
x<-as.factor(test.data$Class)
confusionMatrix( data = prev_test_lda$class, reference = x, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 27  6
##          1  4 22
##
##                Accuracy : 0.8305
##                  95% CI : (0.7103, 0.9156)
##     No Information Rate : 0.5254
##     P-Value [Acc > NIR] : 9.357e-07
##
##                   Kappa : 0.659
##
##  Mcnemar's Test P-Value : 0.7518
##
##             Sensitivity : 0.7857
##             Specificity : 0.8710
##          Pos Pred Value : 0.8462
##          Neg Pred Value : 0.8182
##              Prevalence : 0.4746
##          Detection Rate : 0.3729
##    Detection Prevalence : 0.4407
##       Balanced Accuracy : 0.8283
##
##        'Positive' Class : 1
##
```

In alto vi è la matrice di confusione e si nota:

- *TP=22*: il numero di unità previste con label 1 e che effettivamente nel collettivo hanno malattie cardiache.
- *TN=27*: il numero di unità previste con label 0 e che effettivamente nel collettivo non hanno malattie cardiache
- *FP=4* : coloro a cui è stata prevista una label pari ad 1, ma in realtà hanno una label pari a 0.
- *FN=6* : coloro a cui è stata prevista una label pari a 0, ma in realtà hanno una label pari ad 1 (cioè in realtà hanno malattie cardiache).

In seguito alla matrice di confusione sono riportate alcune metriche:

- *accuratezza*: è data dal rapporto tra gli elementi sulla diagonale principale e il numero totale di osservazioni.

```r
acc<-(22+27)/59
acc
```

```
## [1] 0.8305085
```

Indica, dunque, che l'85% circa delle unità sono state correttamente classificate.

- *no information rate*: indica la proporzione maggiore delle classi osservate (in questo caso essendo pari a 0.52 la presenza di dati appartenenti alla classe 0 è piuttosto simile rispetto al numero di dati appartenenti alla classe 1).
- *P-Value [Acc > NIR]*: il p-value sarebbe quello di un chi-quadro (come se trattassi la matrice di confidenza come una tabella di contingenza). Se l'indice è 0 è come se le y fossero indipendente dalle x (accetto l'ipotesi nulla di indipendenza dei dati dai predittori). Se il test è significativo (come in questo caso), rifiuto l'ipotesi nulla.
- *Kappa*: più il valore è vicino ad 1 più il classificatore sarà performante. E' calcolato come:

```
a<- (22+4)/59
b<- (22+6)/59
p_yes<- a*b
c<- (27+6)/59
d<- (27+4)/59
p_no<-c*d
p_e<-p_yes+p_no
k<-(acc-p_e)/(1-p_e)
k
```

```
## [1] 0.6589595
```

- *McNemar's Test P-Value*: Il test di McNemar controlla se i disaccordi tra evento e non evento coincidono. Cioè è come se prendesse in considerazione i falsi negativi e falsi positivi della matrice di confusione (0/1 e 1/0). Il test controlla se c'è una differenza significativa tra i conteggi in queste due celle. Se i conteggi sono simili, vuol dire che gli errori sono commessi nella stessa proporzione. Quindi, il risultato del test non è significativo.
- *Sensitività*: il numero di positivi correttamente classificati.

```
22/(22+6)
```

```
## [1] 0.7857143
```

- *Specificità*: numero di negativi correttamente classificati.

```
27/(27+4)
```

```
## [1] 0.8709677
```

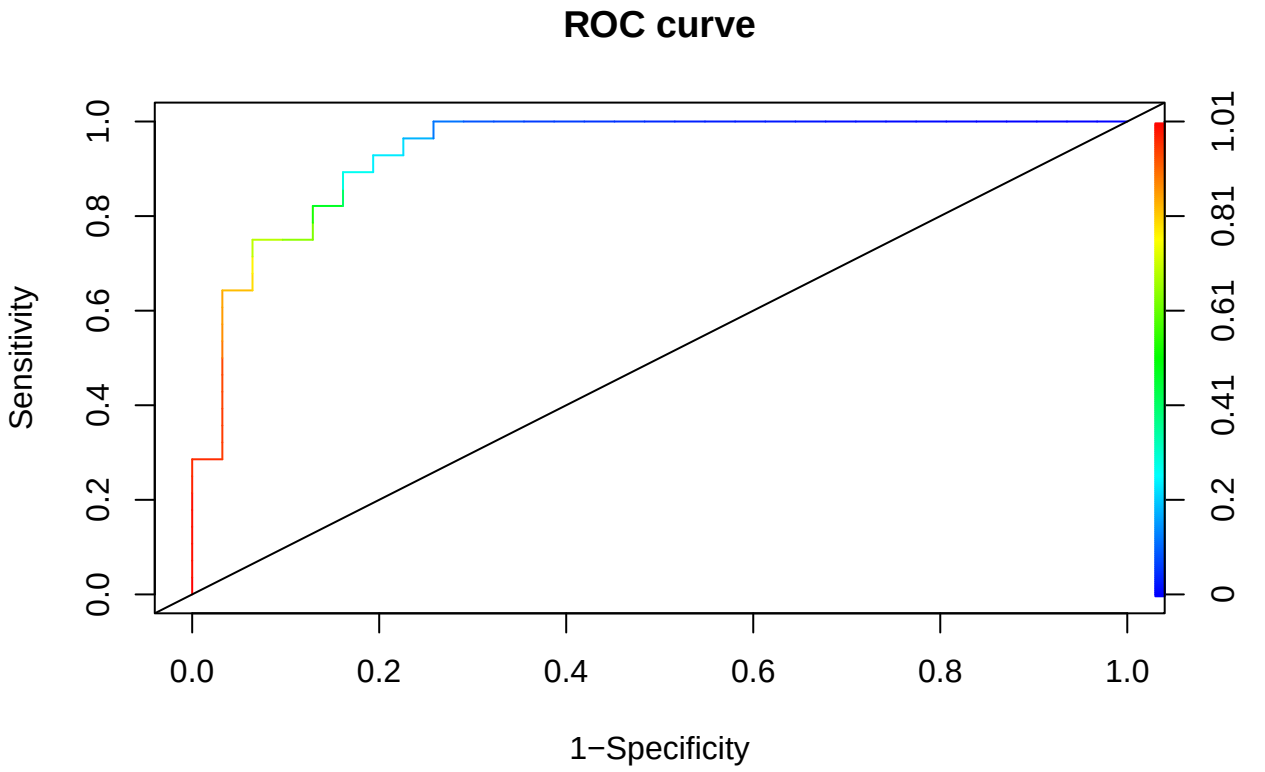
Va specificato che vi è un trade-off tra sensitività e specificità. Cioè l'aumento della prima, comporta la diminuzione della seconda e viceversa.

- *Pos Pred Value*: indica la probabilità che un risultato positivo del test corrisponda alla reale presenza della malattia, quindi ad una vera positività.
- *Neg Pred Value*: è la probabilità che un risultato negativo del test corrisponda ad una vera negatività (reale assenza di malattia cardiaca). In un contesto medico, credo abbia senso cercare di ottenere un più alto valore di NPV, in modo tale da essere il più certi possibile che coloro ritenuti sani lo siano realmente.
- *prevalenza*: il rapporto tra le unità che hanno effettivamente label pari ad 1 ed n (l'abbiamo usato nel calcolo di K).

- *Detection Rate*: il numero di eventi (classe 1) classificati correttamente sul totale.
- *Detection Prevalence*: il rapporto tra il numero di pazienti previsti come 1 sul totale (anche questo è stato usato nel calcolo di K).
- *Balanced Accuracy*: il balanced accuracy (detto anche F1) è la media armonica della sensitività e specificità.

ROC CURVE Nel caso dell'LDA sappiamo che un soggetto viene assegnato ad una popolazione piuttosto che all'altra in base alla probabilità a posteriori di appartenere ad essa. Generalmente viene imposta una threashold (default = 0.5), se la probabilità a posteriori sarà superiore a questa soglia il soggetto verrà assegnato alla popolazione 1; altrimenti verrà etichettato come 0. Ovviamente, il valore della soglia può essere modificato tenendo conto che non è possibile massimizzare sensitività e specificità contemporaneamente e considerando anche il problema di riferimento. Nel nostro caso, presumo sia più opportuno minimizzare il numero di falsi negativi (coloro che prevediamo essere privi di malattie cardiache ma in realtà sono malati) piuttosto che di falsi positivi. In questo modo, verrebbero eseguiti ulteriori controlli sul paziente in modo da prevenire o accorgersi in tempo di eventuali patologie. D'altro canto non avrebbe senso neanche prevedere tutti come positivi ed effettuare ulteriori accertamenti su tutti i pazienti, poichè questo richiederebbe ovviamente dei costi. Dunque la curva ROCR ci permette di trovare un compromesso tra sensitività e specificità.

```
pred <- prediction(prev_test_lda$posterior[,"1"], test.data$Class)
roc_ROCR <- ROCR::performance(pred, "tpr", "fpr")
plot(roc_ROCR, main = "ROC curve", colorize = T, xlab="1-Specificity", ylab="Sensitivity" )
abline(a = 0, b = 1)
```



Notiamo che sulle ascisse è posta 1-specificità, desideriamo sia il più piccolo possibile. Sulle ordinate, invece, è presente la sensitività che vogliamo sia il più alto possibile. Un modello completamente efficace sarebbe quello in cui la curva coincidesse con i lati del triangolo superiore (perchè non verrebbe commesso nessun errore). Invece più si avvicina alla bisettrice in nero, più vuol dire che il classificatore si comporterebbe in modo casuale. Quindi, l'obiettivo è allontanarsi dalla bisettrice, nel nostro caso la curva è quasi prossima

ai cateti del triangolo, quindi il classificatore sembra performare bene. Probabilmente, nel caso sopra riportato sarebbe più conveniente scegliere una soglia più bassa, ad esempio in corrispondenza del colore verde-azzurro. Ovviamente abbassando la soglia è vero che aumenteranno i TP, ma aumenteranno anche i FP.

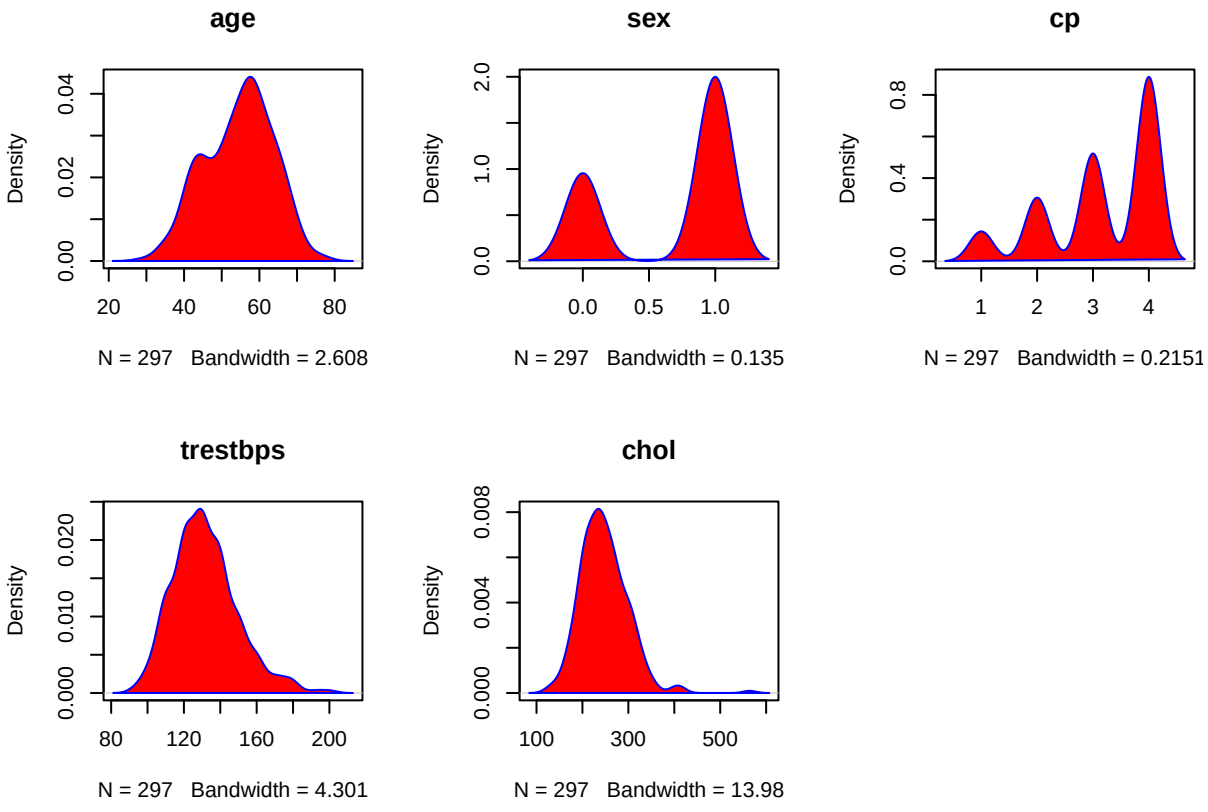
```
perf.2<-ROCR::performance(pred,measure="auc",fpr.stop=0.4)
auc_ROCR <- ROCR::performance(pred, measure = "auc")
auc_ROCR <- auc_ROCR@y.values[[1]]
auc_ROCR
```

```
## [1] 0.9366359
```

In questo modo calcolo l'AUC, ponendo una threashold pari a 0.4 ho una combinazione di fpr e tpr che mi da un'area sotto la curva pari a 0.94 circa. Ricordando che un modello le cui previsioni sono tutte errate avrà un AUC di 0, viceversa se le previsioni sono tutte corrette l'AUC sarà pari a 1.

In realtà nell'LDA il vettore x (delle variabili indipendenti) si ipotizza che si distribuisca come una normalità multivariata. E quindi le variabili indipendenti provengono da una distribuzione normale. Per cercare di visualizzare la distribuzione di una variabile utilizziamo i grafici di densità.

```
par(mfrow=c(2, 3))
colnames <- dimnames(df)[[2]]
for (i in 1:5) {
    d <- density(df[,i])
    plot(d, type="n", main=colnames[i])
    polygon(d, col="red", border="blue")
}
```



Sappiamo che i grafici di densità sono utili per le variabili continue, questo possiamo notarlo osservando la variabile sex o cp, che sappiamo essere discrete.

Adesso, stimiamo il modello LDA sul training set e lo testiamo sul test set utilizzando però come predittori solamente le variabili continue.

```
heart_lda1<-lda(Class ~ trestbps+chol+thalach+oldpeak, data= train.data)
heart_lda1
```

```
## Call:
## lda(Class ~ trestbps + chol + thalach + oldpeak, data = train.data)
##
## Prior probabilities of groups:
##         0         1
## 0.5420168 0.4579832
##
## Group means:
##   trestbps     chol  thalach  oldpeak
## 0 130.0388 241.2248 158.3488 0.620155
## 1 135.6239 252.8991 139.0367 1.591743
##
## Coefficients of linear discriminants:
##                   LD1
## trestbps  0.011464234
## chol      0.004160562
## thalach  -0.032086637
## oldpeak   0.517383632
```

Anche in questo caso abbiamo solo una variabile canonica, poichè il numero di gruppi è sempre due. Quindi avremo G-1=1.

```
prev_test_lda1<-predict(heart_lda1, test.data,type = "p")
head(prev_test_lda1$class)
```

```
## [1] 1 1 0 0 0 0
## Levels: 0 1
```

Le due prime unità del test set sono classificate come 1.

```
head(prev_test_lda1$posterior)
```

```
##             0         1
## 2  0.09275215 0.9072479
## 12 0.46816550 0.5318345
## 17 0.80490792 0.1950921
## 18 0.62160215 0.3783978
## 21 0.51189343 0.4881066
## 22 0.58315552 0.4168445
```

La prima con una probabilità a posteriori pari a 0.9 e la seconda con una probabilità pari a 0.53. Questo è un caso che ci permette di capire l'importanza della threashold, poichè essendo di default posta a 0.5, in questo caso è vero che il paziente è stato classificato con la label 1, ma l'incertezza di previsione è molto alta,

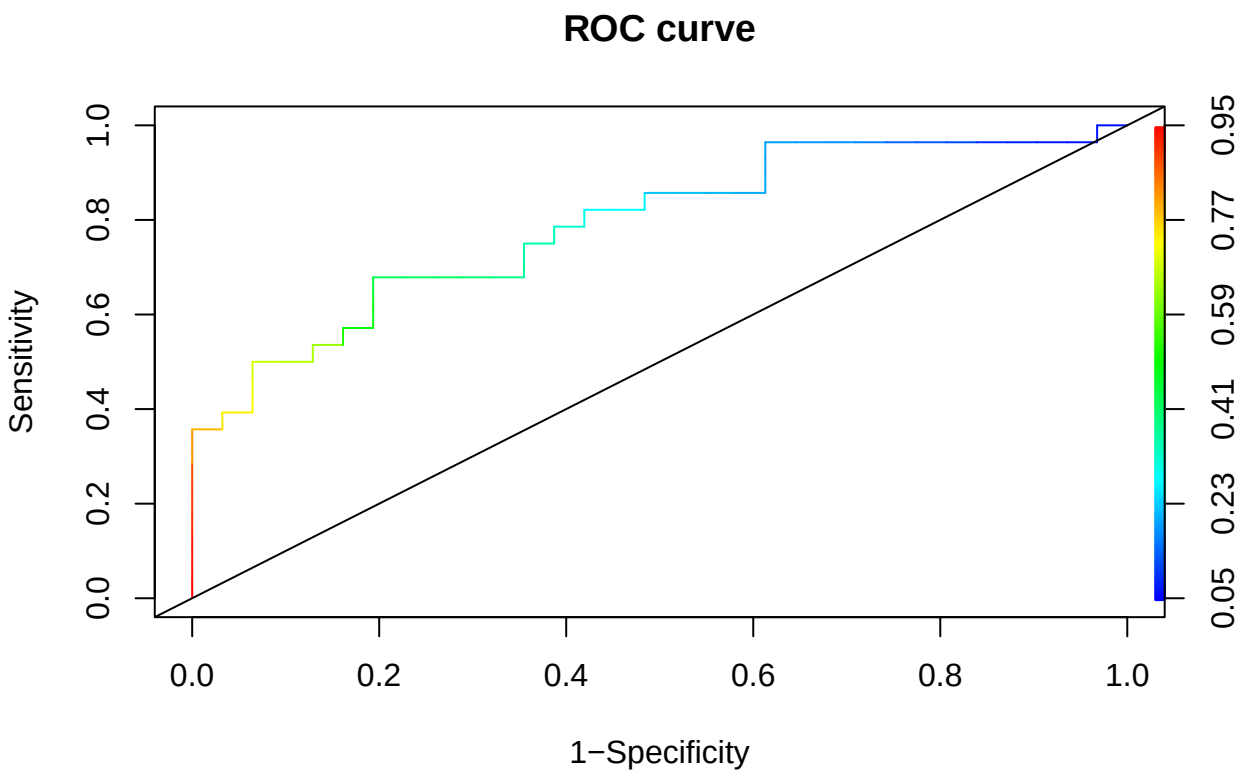
perchè essendo la threashold= 0.5 si tratta quasi di un caso limite. Un discorso analogo può essere fatto con il 5 paziente, questa volta classificato come 0 con una probabilità del 51.2% circa; probabilmente in un caso del genere sarebbe stato più opportuno effettuare ulteriori accertamenti e valutare se effettivamente si tratta di un paziente con malattie cardiache o meno.

```
confusionMatrix( data = prev_test_lda1$class, reference = x, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 26 13
##          1  5 15
##
##                Accuracy : 0.6949
##                  95% CI : (0.5613, 0.8081)
##     No Information Rate : 0.5254
##     P-Value [Acc > NIR] : 0.006077
##
##                   Kappa : 0.3797
##
##  Mcnemar's Test P-Value : 0.098960
##
##             Sensitivity : 0.5357
##             Specificity : 0.8387
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.6667
##              Prevalence : 0.4746
##          Detection Rate : 0.2542
##    Detection Prevalence : 0.3390
##       Balanced Accuracy : 0.6872
##
##        'Positive' Class : 1
##
```

A differenza di prima, l'accuratezza del modello diminuisce, sicuramente per l'aumento dei falsi negativi che ammonta a 13 (prima FP=6). Questo, ovviamente si ripercuote in un minor valore della sensitività. Probabilmente in questo caso sarebbe opportuno abbassare il valore della soglia, in modo tale da diminuire il numero di falsi negativi. Visualizziamo per tale ragione la curva ROC:

```
pred1 <- prediction(prev_test_lda1$posterior[,"1"], test.data$Class)
roc_ROCR1 <- ROCR::performance(pred1, "tpr", "fpr")
plot(roc_ROCR1, main = "ROC curve", colorize = T, xlab="1-Specificity", ylab="Sensitivity" )
abline(a = 0, b = 1)
```



Anche dalla curva ROC appare evidente che il modello performa in modo peggiore rispetto al precedente, infatti si è allontanata dai cateti del triangolo superiore per avvicinarsi alla bisettrice. Questo è dovuto sicuramente all'assenza di qualche variabile esplicativa strettamente legata con la variabile di interesse, anche osservando il valore del P-Value [Acc > NIR] nelle statistiche della matrice di confusione, ci rendiamo conto che il valore è decisamente più alto rispetto a prima. Dalla curva ci rendiamo conto che per ottenere un numero inferiore di FN si potrebbe abbassare la soglia portandola fino a circa 0.3 (parte azzurra). Ovviamente, aumenterà anche il numero di falsi positivi.

Abbassiamo la soglia e visualizziamo la nuova matrice di confusione.

```
pdata <- as.data.frame(prev_test_lda1)
pdata$my_custom_predicted_class <- ifelse(pdata$posterior.1 > .30, 1, 0)
a<-as.factor(pdata$my_custom_predicted_class)
confusionMatrix(data = a, reference = x, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 19  6
##          1 12 22
##
##                Accuracy : 0.6949
##                  95% CI : (0.5613, 0.8081)
##     No Information Rate : 0.5254
##     P-Value [Acc > NIR] : 0.006077
##
##                   Kappa : 0.3945
##
##  Mcnemar's Test P-Value : 0.238593
```

```
##
##             Sensitivity : 0.7857
##             Specificity : 0.6129
##          Pos Pred Value : 0.6471
##          Neg Pred Value : 0.7600
##              Prevalence : 0.4746
##          Detection Rate : 0.3729
##    Detection Prevalence : 0.5763
##       Balanced Accuracy : 0.6993
##
##        'Positive' Class : 1
##
```

Come ci aspettavamo il numero di falsi negativi si è ridotto (FN=6) ed è aumentato il numero di FP=12. L'accuratezza del modello è identica alla precedente, ma il valore della sensitività è aumentato (ovviamente quello della specificità è diminuito). Tuttavia, per il ragionamento sostenuto in precedenza sarebbe preferibile una soglia più bassa poichè è preferibile minimizzare il numero di falsi negativi piuttosto che i falsi positivi.

LDA MULTICLASSE

In questo caso non considereremo come variabile di classificazione "Class" che assume valore 0,1; ma prenderemo come riferimento la variabile *num*, che può assumere 5 classi (da 0 a 5 in base alla gravità della malattia cardiaca presente, 0 indica l'assenza).

Dividiamo in training e test set il dataset denominato df (contenente la variabile *num*).

```
set.seed(1998)
training.samples1 = createDataPartition(df$num, p = .8, list = FALSE)
train.data1 <- df[training.samples, ]
test.data1 <- df[-training.samples, ]
```

Eseguiamo l'LDA:

```
heart_lda_num<-lda(num ~ ., data= train.data1)
heart_lda_num
```

```
## Call:
## lda(num ~ ., data = train.data1)
##
## Prior probabilities of groups:
##          0          1          2          3          4
## 0.54201681 0.18067227 0.11764706 0.12184874 0.03781513
##
## Group means:
##        age       sex       cp trestbps     chol        fbs   restecg  thalach
## 0 52.82946 0.5813953 2.829457 130.0388 241.2248 0.15503876 0.8372093 158.3488
## 1 56.06977 0.8139535 3.255814 134.0465 253.6512 0.04651163 1.2093023 148.8140
## 2 57.85714 0.7857143 3.785714 134.3929 259.0000 0.21428571 0.8214286 133.2857
## 3 56.82759 0.7931034 3.724138 137.3793 249.2414 0.20689655 1.2068966 131.1034
## 4 56.44444 0.8888889 3.666667 141.3333 242.1111 0.11111111 1.6666667 135.7778
##       exang  oldpeak    slope        ca     thal
## 0 0.1472868 0.620155 1.457364 0.3100775 3.821705
## 1 0.4186047 1.004651 1.581395 0.6744186 5.441860
```

```
## 2 0.5714286 1.803571 1.928571 1.1428571 6.214286
## 3 0.6551724 2.006897 2.000000 1.4827586 6.275862
## 4 0.4444444 2.400000 2.111111 1.6666667 6.333333
##
## Coefficients of linear discriminants:
##                   LD1          LD2          LD3          LD4
## age      -0.020531376 -0.051017691 -0.042918564  0.036247474
## sex       0.422030954 -0.810078144  0.026289675  0.603236957
## cp        0.332057126  0.018857212 -0.201170235  0.462411442
## trestbps  0.012186601 -0.002732442  0.015403666 -0.001391941
## chol      0.002693724 -0.003308797 -0.007859533  0.004675041
## fbs      -0.292388259  1.681346544 -0.940913079 -0.295368803
## restecg   0.118669654 -0.353450109  0.613893389 -0.184486625
## thalach  -0.018279454 -0.015074930  0.005176966  0.012757640
## exang     0.386314483 -0.340587351 -0.480871898 -1.945945439
## oldpeak   0.222730528  0.295688078  0.173359702  0.377536010
## slope     0.220414628  0.180006671  0.091832042  0.378625539
## ca        0.633146157  0.278659294  0.536237062 -0.288504590
## thal      0.264134711 -0.190655080 -0.135697397  0.041115031
##
## Proportion of trace:
##    LD1    LD2    LD3    LD4
## 0.9017 0.0540 0.0370 0.0073
```

In questo caso per ogni gruppo avrò la probabilità a priori di appartenere ad esso. In seguito ci sarà la media di ogni variabile per ogni gruppo. In questo caso avremo 4 variabili canoniche, poiche G-1=4 (il numero di p=13). Effettuiamo la previsione sul test set.

```
prev_test_lda_num<-predict(heart_lda_num, test.data1)
prev_test_lda_num$class[1:11]
```

```
##  [1] 3 0 0 0 0 0 3 2 3 0 1
## Levels: 0 1 2 3 4
```

La prima unità del test set è stata assegnata al gruppo 3 (si presume abbia una malattia cardiaca abbastanza grave). La decima unità è stata, invece, assegnata al gruppo 1. Visualizziamo con quale probabilità sono state assegnate (le probabilità sono state arrotondate alla 5 cifra decimale):

```
round(prev_test_lda_num$posterior[1:11,], digits = 5)
```

```
##          0       1       2       3       4
## 2  0.00122 0.03391 0.10167 0.63462 0.22857
## 12 0.92947 0.06632 0.00268 0.00126 0.00026
## 17 0.76506 0.19649 0.03091 0.00646 0.00107
## 18 0.86919 0.11422 0.01374 0.00242 0.00042
## 21 0.86582 0.13050 0.00214 0.00144 0.00010
## 22 0.99183 0.00779 0.00028 0.00009 0.00000
## 30 0.03676 0.15082 0.20124 0.53990 0.07127
## 32 0.11928 0.14474 0.49660 0.23462 0.00477
## 37 0.02994 0.15363 0.20865 0.51444 0.09333
## 42 0.73503 0.24121 0.01144 0.01171 0.00061
## 46 0.25790 0.50332 0.11035 0.07015 0.05828
```

Ad esempio, la prima unità è stata classificata come 3 con una probabilità a posteriori del 63.5%; la seconda è stata classificata come 0 con una probablità del 93%. Visualizziamo la matrice di confusione:

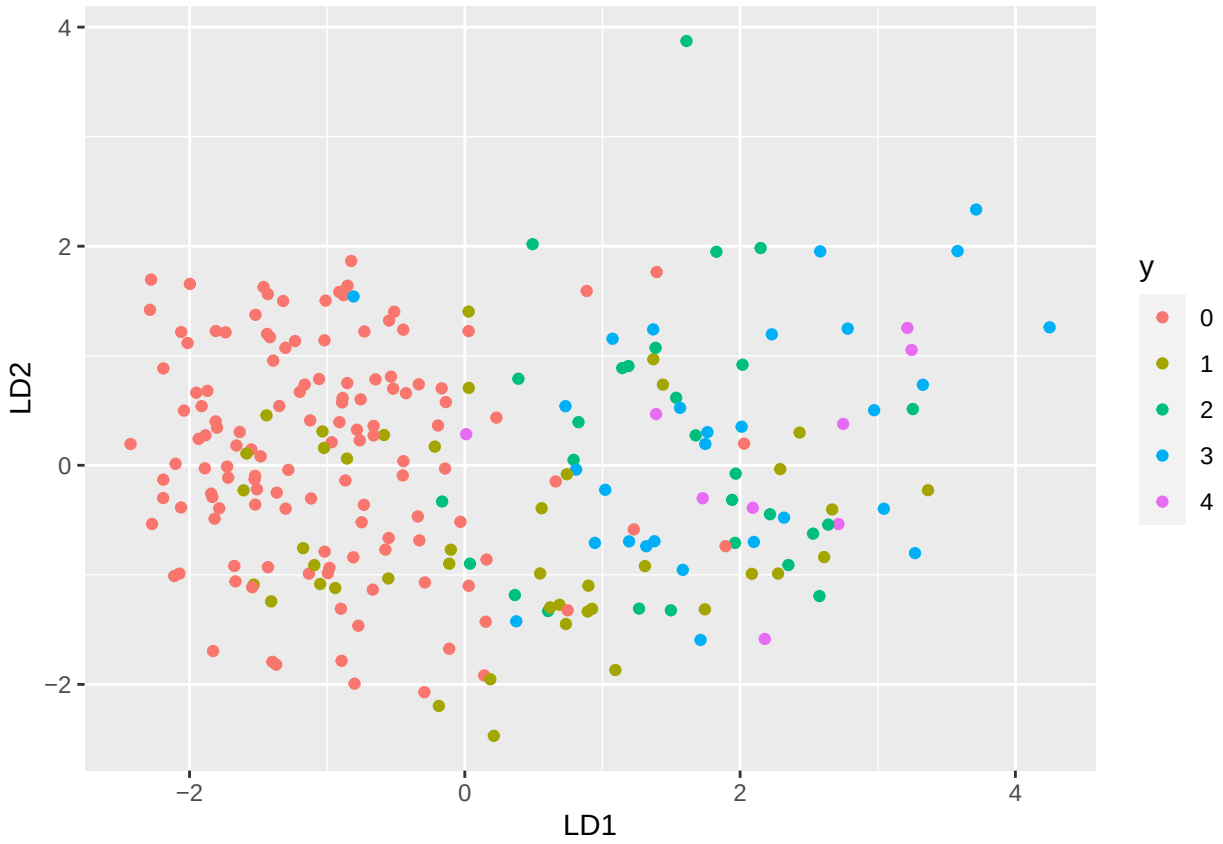
```
x1<-as.factor(test.data1$num)
confusionMatrix( data = prev_test_lda_num$class, reference = x1, positive = "...")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1  2  3  4
##          0 27  5  1  0  0
##          1  4  1  2  0  2
##          2  0  1  2  1  1
##          3  0  4  2  4  0
##          4  0  0  0  1  1
##
## Overall Statistics
##
##                Accuracy : 0.5932
##                  95% CI : (0.4575, 0.7193)
##     No Information Rate : 0.5254
##     P-Value [Acc > NIR] : 0.1809
##
##                   Kappa : 0.3723
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
## Sensitivity            0.8710  0.09091  0.28571   0.6667  0.25000
## Specificity            0.7857  0.83333  0.94231   0.8868  0.98182
## Pos Pred Value         0.8182  0.11111  0.40000   0.4000  0.50000
## Neg Pred Value         0.8462  0.80000  0.90741   0.9592  0.94737
## Prevalence             0.5254  0.18644  0.11864   0.1017  0.06780
## Detection Rate         0.4576  0.01695  0.03390   0.0678  0.01695
## Detection Prevalence   0.5593  0.15254  0.08475   0.1695  0.03390
## Balanced Accuracy      0.8283  0.46212  0.61401   0.7767  0.61591
```

Notiamo che l'accuratezza del modello è inferiore rispetto ai casi precedenti. In linea generale si nota che tutte le statistiche assumono valori abbastanza bassi, ad eccezione delle statistiche calcolate per la classe 0 che risultano abbastanza elevate.

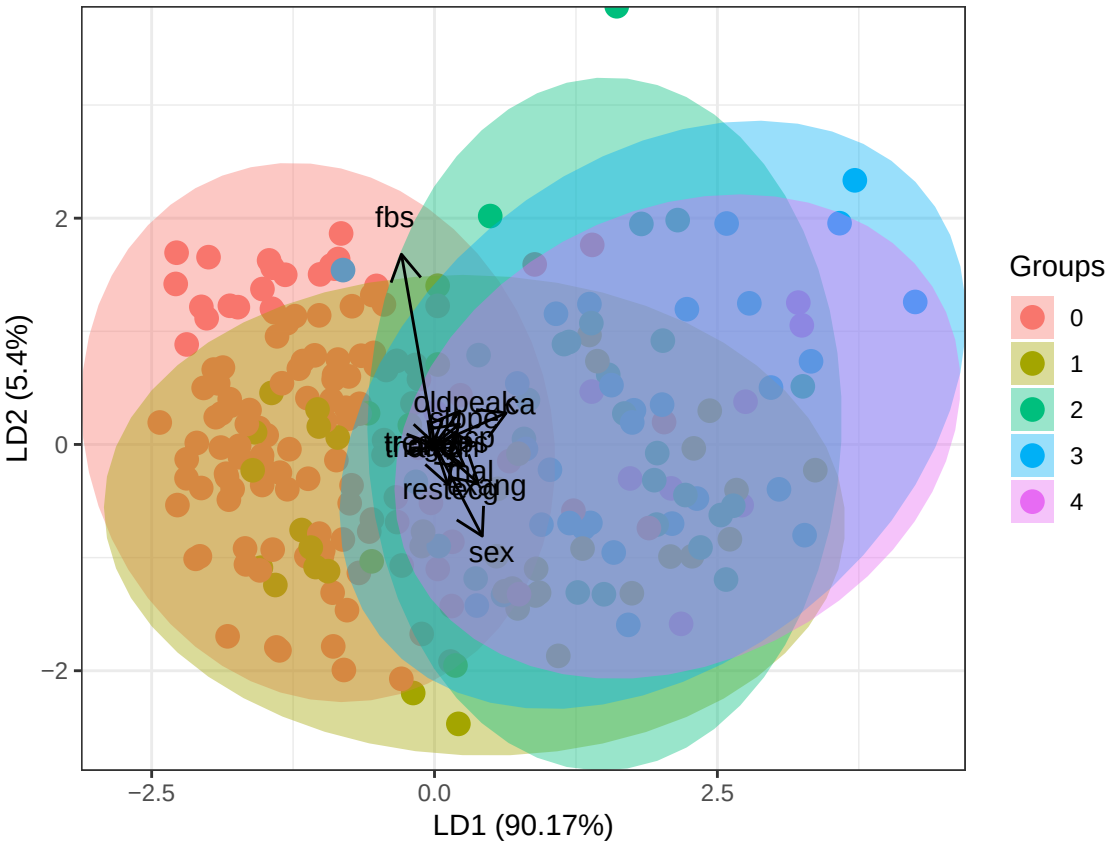
Come detto in precedenza in questo caso è stato possibile ottenere 4 variabili canoniche. Attraverso il pacchetto ggplot è possibile ottenere un LDA PLOT, dove sugli assi abbiamo le prime due variabili canoniche.

```
y<-as.factor(train.data1$num)
lda.data_num <- cbind(train.data1, predict(heart_lda_num)$x)
ggplot(lda.data_num, aes(LD1, LD2))+
  geom_point(aes(color = y))
```



Dal grafico ci rendiamo conto che solo la classe 0 è abbastanza separata dalle altre, le classi 1,2,3 e 4 risultano abbastanza sovrapposte tra loro. Per tale ragione le statistiche più alte sono registrate sulla classe 0.

```
ggord(heart_lda_num, y)
```



Anche con questa rappresentazione è abbastanza evidente che i gruppi tendono a sovrapporsi.

QDA

Nell'analisi discriminante lineare, assumiamo che le matrici di varianza e covarianza delle classi siano uguali. Nell'analisi discriminante quadratica invece supponiamo che le matrici di varianza e covarianza siano diverse. In questo caso la funzione che utilizzeremo sarà qda e gli argomenti sono analoghi a quelli dell'lda. L'analisi verrà svolta sul df_10 dataset:

```
heart_qda <- qda(Class ~ ., data = train.data)
heart_qda
```

```
## Call:
## qda(Class ~ ., data = train.data)
##
## Prior probabilities of groups:
##         0         1
## 0.5420168 0.4579832
##
## Group means:
##        age       sex       cp trestbps     chol       fbs   restecg  thalach
## 0 52.82946 0.5813953 2.829457 130.0388 241.2248 0.1550388 0.8372093 158.3488
## 1 56.76147 0.8073394 3.550459 135.6239 252.8991 0.1376147 1.1467890 139.0367
##       exang  oldpeak    slope        ca     thal
## 0 0.1472868 0.620155 1.457364 0.3100775 3.821705
## 1 0.5229358 1.591743 1.825688 1.0917431 5.935780
```

Adesso, possiamo testare il modello sul test set:

```
prev_test_qda<-predict(heart_qda, test.data, type="prob")
head(prev_test_qda$class)
```

```
## [1] 1 0 0 0 0 0
## Levels: 0 1
```

La prima unità è stata classificata come 1, con una probabilità del 99%.

```
head(prev_test_qda$posterior)
```

```
##               0           1
## 2  2.876821e-06 0.999997123
## 12 9.915567e-01 0.008443295
## 17 9.933919e-01 0.006608142
## 18 9.761055e-01 0.023894475
## 21 9.228777e-01 0.077122344
## 22 9.989572e-01 0.001042804
```

Visualizziamo la matrice di confusione e le statistiche:

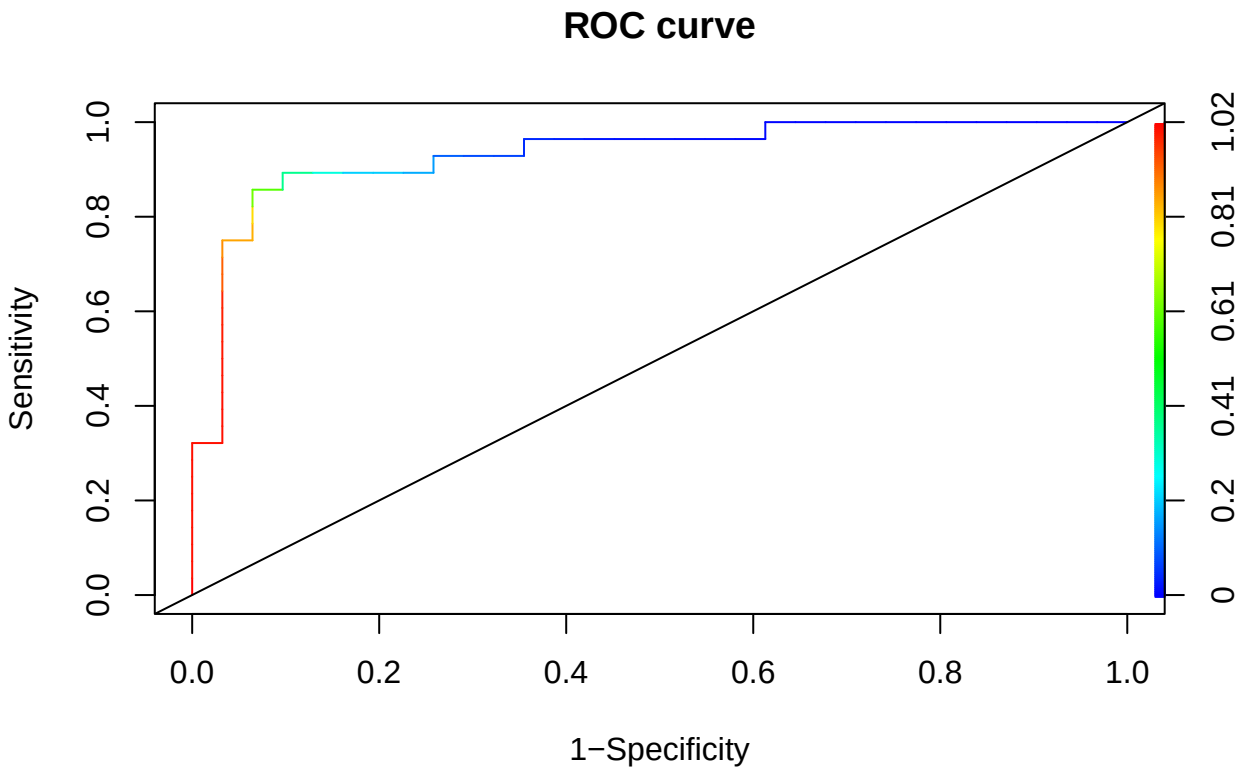
```
x<-as.factor(test.data$Class)
confusionMatrix( data = prev_test_qda$class, reference = x, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 29  4
##          1  2 24
##
##                Accuracy : 0.8983
##                  95% CI : (0.7917, 0.9618)
##     No Information Rate : 0.5254
##     P-Value [Acc > NIR] : 8.998e-10
##
##                   Kappa : 0.7954
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.8571
##             Specificity : 0.9355
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.8788
##              Prevalence : 0.4746
##          Detection Rate : 0.4068
##    Detection Prevalence : 0.4407
##       Balanced Accuracy : 0.8963
##
##        'Positive' Class : 1
##
```

Il modello ha un'accuratezza dell'89.83%, anche i valori di sensitività e specificità risultano abbastanza alti, infatti solo 6 osservazioni sono erroneamente classificate. Se lo confrontiamo con il risultato ottenuto sul medesimo dataset (train.data e test.data), notiamo che il classificatore quadratico performa meglio rispetto al classificatore lineare.

Visualizziamo la ROC curve:

```
pred <- prediction(prev_test_qda$posterior[,"1"], test.data$Class)
roc_ROCR <- ROCR::performance(pred, "tpr", "fpr")
plot(roc_ROCR, main = "ROC curve", colorize = T, xlab="1-Specificity", ylab="Sensitivity" )
abline(a = 0, b = 1)
```



Anche dalla ROC curve si può osservare che il modello risulta performante, poichè la curva è molto vicina ai cateti del triangolo superiore.

FONTE DATASET https://archive.ics.uci.edu/ml/datasets/heart+disease