Ejercicio práctico Análisis Discriminante

# 1. Planteamiento del problema

Para este ejercicio nos enfocaremos en un set de datos que representa la calidad de distintos tipos de tinto portugués. Dicha calidad comprende valores entre 3 y 8. En función de 11 atributos distintos que caracterizan cada tipo de vino debemos ser capaces de clasificar la calidad que tendrá dicho vino.

El dataset y el diccionario de datos podrás encontrarlo en la carpeta data.

Así pues, lo primero que haremos es cargar el dataset en R:

# cargamos depenendias necesarias  
require(MASS)  
require(caret)  
require(randomForest)  
require(e1071)  
require(dplyr)

# leemos los datos  
data <- read.csv("/Users/lgutierrez/Proyectos/master/M4/data/4.3\_AD\_ejercicio.csv", sep = ";")  
  
# creamos un dataframe tibble  
df <- as\_tibble(data)  
  
# creamos un auxiliar para no trabajar sobre nuestro dataframe original  
df\_aux <- df  
  
# previsualizamos datos  
head(df\_aux)

## # A tibble: 6 × 12  
## fixed.acidity volatile.acidity citric.acid residual.sugar chlorides  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 7.4 0.7 0 1.9 0.076  
## 2 7.8 0.88 0 2.6 0.098  
## 3 7.8 0.76 0.04 2.3 0.092  
## 4 11.2 0.28 0.56 1.9 0.075  
## 5 7.4 0.7 0 1.9 0.076  
## 6 7.4 0.66 0 1.8 0.075  
## # ℹ 7 more variables: free.sulfur.dioxide <dbl>, total.sulfur.dioxide <dbl>,  
## # density <dbl>, pH <dbl>, sulphates <dbl>, alcohol <dbl>, quality <int>

# vemos resumen  
summary(df\_aux)

## fixed.acidity volatile.acidity citric.acid residual.sugar   
## Min. : 4.60 Min. :0.1200 Min. :0.000 Min. : 0.900   
## 1st Qu.: 7.10 1st Qu.:0.3900 1st Qu.:0.090 1st Qu.: 1.900   
## Median : 7.90 Median :0.5200 Median :0.260 Median : 2.200   
## Mean : 8.32 Mean :0.5278 Mean :0.271 Mean : 2.539   
## 3rd Qu.: 9.20 3rd Qu.:0.6400 3rd Qu.:0.420 3rd Qu.: 2.600   
## Max. :15.90 Max. :1.5800 Max. :1.000 Max. :15.500   
## chlorides free.sulfur.dioxide total.sulfur.dioxide density   
## Min. :0.01200 Min. : 1.00 Min. : 6.00 Min. :0.9901   
## 1st Qu.:0.07000 1st Qu.: 7.00 1st Qu.: 22.00 1st Qu.:0.9956   
## Median :0.07900 Median :14.00 Median : 38.00 Median :0.9968   
## Mean :0.08747 Mean :15.87 Mean : 46.47 Mean :0.9967   
## 3rd Qu.:0.09000 3rd Qu.:21.00 3rd Qu.: 62.00 3rd Qu.:0.9978   
## Max. :0.61100 Max. :72.00 Max. :289.00 Max. :1.0037   
## pH sulphates alcohol quality   
## Min. :2.740 Min. :0.3300 Min. : 8.40 Min. :3.000   
## 1st Qu.:3.210 1st Qu.:0.5500 1st Qu.: 9.50 1st Qu.:5.000   
## Median :3.310 Median :0.6200 Median :10.20 Median :6.000   
## Mean :3.311 Mean :0.6581 Mean :10.42 Mean :5.636   
## 3rd Qu.:3.400 3rd Qu.:0.7300 3rd Qu.:11.10 3rd Qu.:6.000   
## Max. :4.010 Max. :2.0000 Max. :14.90 Max. :8.000

## 1.1 Preparación del dataset.

Tal y como podrás comprobar, el dataset tiene una dimensión de 1599 observaciones y 11 variables.

* **Ejercicio 1**: Modifica la variable quality, de tal modo que si la calidad se encuentra en los valores 3 o 4, pasará a categorizarse como “pobre”, si se encuentra en los valores 5 o 6 pasará a categorizarse como “aceptable” y si se encuentra en los valores 7 o 8, pasará a categorizarse como “bueno”. Posteriormente transforma la variable quality a factor.

# modifico la variable quality  
df\_aux$quality\_r <- cut(df\_aux$quality, breaks = c(-Inf, 4, 6, Inf))  
  
# observamos resultado de mapeo  
print(df\_aux[, c("quality","quality\_r")], n=20)

## # A tibble: 1,599 × 2  
## quality quality\_r  
## <int> <fct>   
## 1 5 (4,6]   
## 2 5 (4,6]   
## 3 5 (4,6]   
## 4 6 (4,6]   
## 5 5 (4,6]   
## 6 5 (4,6]   
## 7 5 (4,6]   
## 8 7 (6, Inf]   
## 9 7 (6, Inf]   
## 10 5 (4,6]   
## 11 5 (4,6]   
## 12 5 (4,6]   
## 13 5 (4,6]   
## 14 5 (4,6]   
## 15 5 (4,6]   
## 16 5 (4,6]   
## 17 7 (6, Inf]   
## 18 5 (4,6]   
## 19 4 (-Inf,4]   
## 20 6 (4,6]   
## # ℹ 1,579 more rows

# observamos los levels de nuestro factor  
levels(df\_aux$quality\_r)

## [1] "(-Inf,4]" "(4,6]" "(6, Inf]"

# convertimos los levels a etiquetas legibles  
levels(df\_aux$quality\_r) <- c("pobre", "aceptable", "bueno")  
  
# controlamos algunas ocurrencias de nuestro mapeo  
print(df\_aux[, c("quality","quality\_r")], n=50)

## # A tibble: 1,599 × 2  
## quality quality\_r  
## <int> <fct>   
## 1 5 aceptable  
## 2 5 aceptable  
## 3 5 aceptable  
## 4 6 aceptable  
## 5 5 aceptable  
## 6 5 aceptable  
## 7 5 aceptable  
## 8 7 bueno   
## 9 7 bueno   
## 10 5 aceptable  
## 11 5 aceptable  
## 12 5 aceptable  
## 13 5 aceptable  
## 14 5 aceptable  
## 15 5 aceptable  
## 16 5 aceptable  
## 17 7 bueno   
## 18 5 aceptable  
## 19 4 pobre   
## 20 6 aceptable  
## 21 6 aceptable  
## 22 5 aceptable  
## 23 5 aceptable  
## 24 5 aceptable  
## 25 6 aceptable  
## 26 5 aceptable  
## 27 5 aceptable  
## 28 5 aceptable  
## 29 5 aceptable  
## 30 6 aceptable  
## 31 5 aceptable  
## 32 6 aceptable  
## 33 5 aceptable  
## 34 6 aceptable  
## 35 5 aceptable  
## 36 6 aceptable  
## 37 6 aceptable  
## 38 7 bueno   
## 39 4 pobre   
## 40 5 aceptable  
## 41 5 aceptable  
## 42 4 pobre   
## 43 6 aceptable  
## 44 5 aceptable  
## 45 5 aceptable  
## 46 4 pobre   
## 47 5 aceptable  
## 48 5 aceptable  
## 49 5 aceptable  
## 50 5 aceptable  
## # ℹ 1,549 more rows

# checkeamos que no tengamos valores faltantes  
sum(is.na(df\_aux))

## [1] 0

* **Ejercicio 2**: Crea un nuevo dataset que contenga todas las variables explicativas normalizadas en rango 0-1 y la etiqueta a predecir (denominada quality en el conjunto de datos inicial).

# excluyo variable categoria  
dataset <- df\_aux[,-which(names(df\_aux) == "quality\_r" | names(df\_aux) == "quality")]  
  
# nnormalizo las variables del datset en rango 0-1  
maxs <- apply(dataset, 2, max)  
mins <- apply(dataset, 2, min)  
  
# creo nuevo dataset con las variables normalizadas y la etiqueta a predecir  
dataset <- as.data.frame(scale(dataset, center=mins, scale= maxs - mins))  
  
dataset$class <- df\_aux$quality\_r  
  
# visualizamos resumen de los datos  
head(dataset)

## fixed.acidity volatile.acidity citric.acid residual.sugar chlorides  
## 1 0.2477876 0.3972603 0.00 0.06849315 0.1068447  
## 2 0.2831858 0.5205479 0.00 0.11643836 0.1435726  
## 3 0.2831858 0.4383562 0.04 0.09589041 0.1335559  
## 4 0.5840708 0.1095890 0.56 0.06849315 0.1051753  
## 5 0.2477876 0.3972603 0.00 0.06849315 0.1068447  
## 6 0.2477876 0.3698630 0.00 0.06164384 0.1051753  
## free.sulfur.dioxide total.sulfur.dioxide density pH sulphates  
## 1 0.1408451 0.09893993 0.5675477 0.6062992 0.1377246  
## 2 0.3380282 0.21554770 0.4941263 0.3622047 0.2095808  
## 3 0.1971831 0.16961131 0.5088106 0.4094488 0.1916168  
## 4 0.2253521 0.19081272 0.5822320 0.3307087 0.1497006  
## 5 0.1408451 0.09893993 0.5675477 0.6062992 0.1377246  
## 6 0.1690141 0.12014134 0.5675477 0.6062992 0.1377246  
## alcohol class  
## 1 0.1538462 aceptable  
## 2 0.2153846 aceptable  
## 3 0.2153846 aceptable  
## 4 0.2153846 aceptable  
## 5 0.1538462 aceptable  
## 6 0.1538462 aceptable

summary(dataset)

## fixed.acidity volatile.acidity citric.acid residual.sugar   
## Min. :0.0000 Min. :0.0000 Min. :0.000 Min. :0.00000   
## 1st Qu.:0.2212 1st Qu.:0.1849 1st Qu.:0.090 1st Qu.:0.06849   
## Median :0.2920 Median :0.2740 Median :0.260 Median :0.08904   
## Mean :0.3292 Mean :0.2793 Mean :0.271 Mean :0.11225   
## 3rd Qu.:0.4071 3rd Qu.:0.3562 3rd Qu.:0.420 3rd Qu.:0.11644   
## Max. :1.0000 Max. :1.0000 Max. :1.000 Max. :1.00000   
## chlorides free.sulfur.dioxide total.sulfur.dioxide density   
## Min. :0.00000 Min. :0.00000 Min. :0.00000 Min. :0.0000   
## 1st Qu.:0.09683 1st Qu.:0.08451 1st Qu.:0.05654 1st Qu.:0.4060   
## Median :0.11185 Median :0.18310 Median :0.11307 Median :0.4905   
## Mean :0.12599 Mean :0.20951 Mean :0.14300 Mean :0.4902   
## 3rd Qu.:0.13022 3rd Qu.:0.28169 3rd Qu.:0.19788 3rd Qu.:0.5701   
## Max. :1.00000 Max. :1.00000 Max. :1.00000 Max. :1.0000   
## pH sulphates alcohol class   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 pobre : 63   
## 1st Qu.:0.3701 1st Qu.:0.1317 1st Qu.:0.1692 aceptable:1319   
## Median :0.4488 Median :0.1737 Median :0.2769 bueno : 217   
## Mean :0.4497 Mean :0.1965 Mean :0.3112   
## 3rd Qu.:0.5197 3rd Qu.:0.2395 3rd Qu.:0.4154   
## Max. :1.0000 Max. :1.0000 Max. :1.0000

* **Ejercicio 3**: Crea un subconjunto de entreno que represente el 70% del nuevo dataframe creado y un subconjunto de testing que represente el otro 30%.

# creo subconjunto de entreno (70% de las observaciones) y test (30%)  
set.seed(12345)  
  
index <- sample(1:nrow(dataset), round(nrow(dataset) \* 0.7), replace = FALSE)  
x\_train <- dataset[index,]  
  
# observamos tamaño y algunos valores  
glimpse(x\_train)

## Rows: 1,119  
## Columns: 12  
## $ fixed.acidity <dbl> 0.32743363, 0.37168142, 0.22123894, 0.15929204, 0…  
## $ volatile.acidity <dbl> 0.40753425, 0.36986301, 0.36986301, 0.51027397, 0…  
## $ citric.acid <dbl> 0.15, 0.26, 0.00, 0.03, 0.22, 0.46, 0.14, 0.13, 0…  
## $ residual.sugar <dbl> 0.06164384, 0.05479452, 0.20547945, 0.15753425, 0…  
## $ chlorides <dbl> 0.12854758, 0.10350584, 0.12353923, 0.09849750, 0…  
## $ free.sulfur.dioxide <dbl> 0.12676056, 0.04225352, 0.22535211, 0.36619718, 0…  
## $ total.sulfur.dioxide <dbl> 0.16254417, 0.06007067, 0.13780919, 0.18374558, 0…  
## $ density <dbl> 0.4941263, 0.5161527, 0.5528634, 0.3619677, 0.604…  
## $ pH <dbl> 0.3858268, 0.3228346, 0.5669291, 0.6850394, 0.496…  
## $ sulphates <dbl> 0.26347305, 0.24550898, 0.12574850, 0.09580838, 0…  
## $ alcohol <dbl> 0.1692308, 0.1230769, 0.1692308, 0.6615385, 0.169…  
## $ class <fct> aceptable, aceptable, aceptable, aceptable, acept…

# Creo subconjunto de testing (30% de las observaciones)  
test <- dataset[-index,]  
  
# observamos tamaño y algunos valores  
glimpse(test)

## Rows: 480  
## Columns: 12  
## $ fixed.acidity <dbl> 0.28318584, 0.24778761, 0.23893805, 0.18584071, 0…  
## $ volatile.acidity <dbl> 0.4383562, 0.3972603, 0.3630137, 0.3150685, 0.342…  
## $ citric.acid <dbl> 0.04, 0.00, 0.00, 0.08, 0.19, 0.56, 0.51, 0.31, 0…  
## $ residual.sugar <dbl> 0.09589041, 0.06849315, 0.02054795, 0.06164384, 0…  
## $ chlorides <dbl> 0.13355593, 0.10684474, 0.08848080, 0.14190317, 0…  
## $ free.sulfur.dioxide <dbl> 0.19718310, 0.14084507, 0.19718310, 0.19718310, 0…  
## $ total.sulfur.dioxide <dbl> 0.16961131, 0.09893993, 0.05300353, 0.20848057, 0…  
## $ density <dbl> 0.5088106, 0.5675477, 0.3325991, 0.4280470, 0.626…  
## $ pH <dbl> 0.4094488, 0.6062992, 0.5118110, 0.4251969, 0.338…  
## $ sulphates <dbl> 0.19161677, 0.13772455, 0.08383234, 0.12574850, 0…  
## $ alcohol <dbl> 0.2153846, 0.1538462, 0.2461538, 0.1230769, 0.123…  
## $ class <fct> aceptable, aceptable, bueno, aceptable, aceptable…

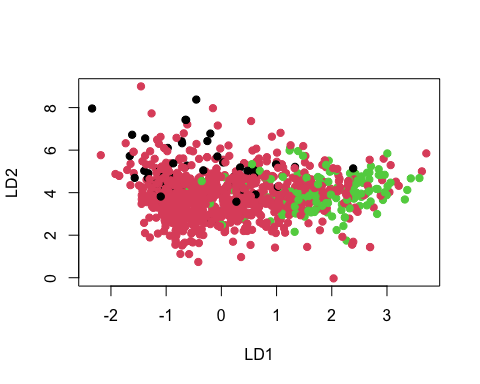
## 1.2 El LDA como predictor.

* **Ejercicio 4**: Crea un modelo LDA y grafica las 2 nuevas dimensiones creadas en un gráfico en el que se puedan visualizar las 3 categorías de la etiqueta a predecir por colores.¿Consideras que el LDA ha segmentado adecuadamente las observaciones en función de la clase a predecir? Justifica tu respuesta.

# creo el objeto con el modelo LDA llamado model  
set.seed(12345)  
model <- lda(class ~ ., data = x\_train)  
  
# visualizamos parametros del modelo  
model

## Call:  
## lda(class ~ ., data = x\_train)  
##   
## Prior probabilities of groups:  
## pobre aceptable bueno   
## 0.04468275 0.82394996 0.13136729   
##   
## Group means:  
## fixed.acidity volatile.acidity citric.acid residual.sugar chlorides  
## pobre 0.2922124 0.4086301 0.1760000 0.1131507 0.1357262  
## aceptable 0.3216363 0.2876192 0.2584165 0.1112754 0.1284896  
## bueno 0.3725844 0.2017752 0.3703401 0.1262464 0.1050617  
## free.sulfur.dioxide total.sulfur.dioxide density pH  
## pobre 0.1642254 0.10466431 0.4861233 0.5080315  
## aceptable 0.2199597 0.15365084 0.4978786 0.4506977  
## bueno 0.1828591 0.09944472 0.4425115 0.4380524  
## sulphates alcohol  
## pobre 0.1568862 0.2853846  
## aceptable 0.1915453 0.2857111  
## bueno 0.2459978 0.4840572  
##   
## Coefficients of linear discriminants:  
## LD1 LD2  
## fixed.acidity 1.0830962 6.6124626  
## volatile.acidity -2.7959736 7.7538052  
## citric.acid 0.3537256 1.5109893  
## residual.sugar 1.6700830 4.3235823  
## chlorides -3.1160322 1.3625196  
## free.sulfur.dioxide -0.1211744 -0.5416563  
## total.sulfur.dioxide -1.1144274 -3.3511436  
## density -1.0157856 -7.2154780  
## pH -1.1303057 6.2183206  
## sulphates 3.3154444 1.1721989  
## alcohol 4.4212734 -1.2621647  
##   
## Proportion of trace:  
## LD1 LD2   
## 0.8288 0.1712

# graficamos las dos nuevas dimensiones creadas por el modelo LDA  
proyected\_data <- as.matrix(x\_train[,-which(names(x\_train) == "class")]) %\*% model$scaling  
plot(proyected\_data, col = x\_train[,which(names(x\_train) == "class")], pch = 19)



* **Ejercicio 5**: Crea un modelo utilizando el LDA como clasificador, aplica las predicciones al subconjunto de testing y calcula la matriz de confusión. ¿Consideras que el modelo está acertando adecuadamente las observaciones cuya clase es minoritaria?

# creamos dataset de test excluyendo la variable a predecir  
x\_test <- test[, -which(names(test) == "class")]  
  
# aplicamos modelo para predecir valores de class  
model\_result <- predict(model, x\_test)  
  
# vemos conteo total de clases   
summary(test$class)

## pobre aceptable bueno   
## 13 397 70

# Creo la matriz de confusión  
t <- table(model\_result$class, test$class)  
  
print(confusionMatrix(t))

## Confusion Matrix and Statistics  
##   
##   
## pobre aceptable bueno  
## pobre 2 7 0  
## aceptable 10 370 38  
## bueno 1 20 32  
##   
## Overall Statistics  
##   
## Accuracy : 0.8417   
## 95% CI : (0.8059, 0.8732)  
## No Information Rate : 0.8271   
## P-Value [Acc > NIR] : 0.2179   
##   
## Kappa : 0.3983   
##   
## Mcnemar's Test P-Value : 0.0683   
##   
## Statistics by Class:  
##   
## Class: pobre Class: aceptable Class: bueno  
## Sensitivity 0.153846 0.9320 0.45714  
## Specificity 0.985011 0.4217 0.94878  
## Pos Pred Value 0.222222 0.8852 0.60377  
## Neg Pred Value 0.976645 0.5645 0.91101  
## Prevalence 0.027083 0.8271 0.14583  
## Detection Rate 0.004167 0.7708 0.06667  
## Detection Prevalence 0.018750 0.8708 0.11042  
## Balanced Accuracy 0.569428 0.6768 0.70296

### Respuesta

El modelo tiene una precisión de 0.8417, y muestra una tasa de error elevada para las clases minoritarias. Es así como para la clase catalogada como ‘pobre’ tenemos una tasa de error del 84% (11/13)y para la clase ‘bueno’, la misma métrica es del 54% (38/70). El modelo muestra buena precisión para clases mas voluminosas.

## 1.3 El LDA como reductor de dimensionalidad.

Una vez aplicado el LDA como clasificador, procederemos a aplicarlo como reductor de dimensionalidad para utilizar posteriormente un clasificador distinto.

* **Ejercicio 6**: Crea un nuevo dataset de entreno y otro de testing utilizando como variables explicativas las variables creadas por el modelo LDA que has creado anteriormente.

# creamos del nuevo dataset de entreno  
dim\_x\_train <- as.matrix(x\_train[-which(names(x\_train) == "class")]) %\*% model$scaling  
  
dim\_x\_train <- as.data.frame(dim\_x\_train)  
  
dim\_x\_train$class <- x\_train$class  
  
head(dim\_x\_train)

## LD1 LD2 class  
## 142 -0.5421487 4.309116 aceptable  
## 51 -0.3737467 4.287820 aceptable  
## 720 -1.0544402 4.273608 aceptable  
## 730 0.5639612 5.981774 aceptable  
## 1244 -0.7836617 3.136396 aceptable  
## 664 1.5782694 2.561829 aceptable

# creamos del nuevo dataset de testing  
dim\_x\_test <- as.matrix(test[-which(names(test) == "class")]) %\*% model$scaling  
  
dim\_x\_test <- as.data.frame(dim\_x\_test)  
  
head(dim\_x\_test)

## LD1 LD2  
## 3 -0.7657742 4.080824  
## 5 -1.3132183 4.394924  
## 8 -0.6306356 4.890001  
## 11 -1.2011338 2.994643  
## 16 -0.8849784 2.495887  
## 17 0.7667208 2.029179

* **Ejercicio 7**: Entrena un nuevo modelo utilizando el algoritmo del Random Forest sobre el nuevo dataset de entreno que has creado y aplica las predicciones al nuevo dataset de testing que has creado. Posteriormente, extrae la matriz de confusión. ¿Este modelo tiene mayor accuracy que el anterior? ¿Este modelo acierta más o menos en las clases minoritarias que el modelo anterior?

# entreno el modelo con random forest  
set.seed(12345)  
dim.model.rf <- randomForest(class ~ ., data=dim\_x\_train)  
  
# Predicciones con random forest  
dim.predictions.rf <- predict(dim.model.rf, as.data.frame(dim\_x\_test), type='class')  
  
# predictions.rf  
  
# Matriz de confusión  
dim.t <- table(dim.predictions.rf, test$class)  
  
print(confusionMatrix(dim.t))

## Confusion Matrix and Statistics  
##   
##   
## dim.predictions.rf pobre aceptable bueno  
## pobre 1 3 2  
## aceptable 12 372 28  
## bueno 0 22 40  
##   
## Overall Statistics  
##   
## Accuracy : 0.8604   
## 95% CI : (0.8262, 0.8902)  
## No Information Rate : 0.8271   
## P-Value [Acc > NIR] : 0.02824   
##   
## Kappa : 0.4848   
##   
## Mcnemar's Test P-Value : 0.04360   
##   
## Statistics by Class:  
##   
## Class: pobre Class: aceptable Class: bueno  
## Sensitivity 0.076923 0.9370 0.57143  
## Specificity 0.989293 0.5181 0.94634  
## Pos Pred Value 0.166667 0.9029 0.64516  
## Neg Pred Value 0.974684 0.6324 0.92823  
## Prevalence 0.027083 0.8271 0.14583  
## Detection Rate 0.002083 0.7750 0.08333  
## Detection Prevalence 0.012500 0.8583 0.12917  
## Balanced Accuracy 0.533108 0.7275 0.75889

* **Ejercicio 8**: Entrena un nuevo modelo utilizando el algoritmo del Random Forest sobre el dataset de entreno inicial que has utilizado para el modelo del LDA como clasificador y aplica las predicciones al dataset de testing que utilizaste para el modelo del LDA como clasificador. ¿Este modelo tiene mayor accuracy que los anteriores? ¿Este modelo acierta más o menos en las clases minoritarias que los modelos anteriores?

set.seed(12345)  
  
# entreno el modelo con random forest  
model.rf.1 <- randomForest(class ~ ., data=x\_train)  
  
# obtenemos predicciones con random forest  
predictions.rf.1 <- predict(model.rf.1, as.data.frame(x\_test), type='class')  
  
# vemos la matriz de confusión  
t.1 <- table(predictions.rf.1, test$class)  
  
print(confusionMatrix(t.1))

## Confusion Matrix and Statistics  
##   
##   
## predictions.rf.1 pobre aceptable bueno  
## pobre 0 0 0  
## aceptable 13 384 37  
## bueno 0 13 33  
##   
## Overall Statistics  
##   
## Accuracy : 0.8688   
## 95% CI : (0.8352, 0.8976)  
## No Information Rate : 0.8271   
## P-Value [Acc > NIR] : 0.00773   
##   
## Kappa : 0.449   
##   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: pobre Class: aceptable Class: bueno  
## Sensitivity 0.00000 0.9673 0.47143  
## Specificity 1.00000 0.3976 0.96829  
## Pos Pred Value NaN 0.8848 0.71739  
## Neg Pred Value 0.97292 0.7174 0.91475  
## Prevalence 0.02708 0.8271 0.14583  
## Detection Rate 0.00000 0.8000 0.06875  
## Detection Prevalence 0.00000 0.9042 0.09583  
## Balanced Accuracy 0.50000 0.6824 0.71986

### Respuesta

Para este modelo obtuvimos una precisión del 0.8604. La tasa de error para la clase ‘pobre’, es levemente mayor al primer modelo, con un 92% (12/13) de inexactitud. A su vez para la clase ‘bueno’ la tasa de error se redujo a 42% (30/70). Para la clase ‘aceptable’ acierta el 93% (370/397) de las veces.

* **Ejercicio 9**: Si tuvieras que presentar uno de estos 3 modelos, cuál elegirías? Justifica tu respuesta.

# Escribe tu respuesta  
print("Podemos notar en el primer modelo, donde utilizamos el Análisis de Discriminante Lineal (LDA) como clasificador, obtuvimos una precisión del 0.8417. Así mismo, se observa un tasa de error eleveada para las clases minoritarias, por ejemplo, para la clase que define la calidad de un vino como 'pobre' nuestro modelo falla el 84% de los casos (11/13), y para la clases 'bueno' la tasa de error ronda el 54% (38/70). Mientras que para la clase más voluminosa, 'aceptable', se tiene una precisión del 93% (370/397), dejando clara una mejor precisión para clases con mayor cantidad de exponentes.  
  
En el segundo modelo donde utilizamos primero LDA para reducir la dimensionalidad de nuestros datos y luego Random Forest como clasificador, obtuvimos una precisión del 0.8604, mejorando sensiblemente al primer modelo. La tasa de error sobre las clase mas pequeña, 'pobre', es levemente mayor al primer modelo, con un 92% (12/13) de inexactitud. A su vez para la clase 'bueno' la tasa de error se redujo a 42% (30/70). Al igual que el primer modelo, cuanto mayor el tamaño de la clase, mejor la precisión del modelo, acierta el 93% de las veces.  
  
El tercer modelo donde el algoritmo Random Forest funciona como predictor sobre el conjunto de datos original, obtuvo una precisión general del 0.8688, ligeramente superior a los dos modelos anteriores. Sin embargo, demostró peor sensibilidad ante clases minoritarias, con un 100% de ineficacia para el grupo 'pobre', y un 52% para 'bueno'. Como contrapartida demuestra gran eficiencia en la clasificación del grupo 'aceptable', el cual cuenta con 397 observaciones.  
  
Considerando los tres modelos y sus características parece adecuado seleccionar al segundo de ellos, donde se realizó la reducción de dimensiones con LDA y se utilizó el algoritmo de Random Forest para clasificar, como el mejor de ellos. Éste modelo parece balancear mejor su precision en clases minoritarias, siendo mas preciso que sus contrincantes al predecir ocurrencias de la clase 'bueno', y siendo levemente mas impreciso para detectar 'pobres' que el modelo primero.")

## [1] "Podemos notar en el primer modelo, donde utilizamos el Análisis de Discriminante Lineal (LDA) como clasificador, obtuvimos una precisión del 0.8417. Así mismo, se observa un tasa de error eleveada para las clases minoritarias, por ejemplo, para la clase que define la calidad de un vino como 'pobre' nuestro modelo falla el 84% de los casos (11/13), y para la clases 'bueno' la tasa de error ronda el 54% (38/70). Mientras que para la clase más voluminosa, 'aceptable', se tiene una precisión del 93% (370/397), dejando clara una mejor precisión para clases con mayor cantidad de exponentes.\n\nEn el segundo modelo donde utilizamos primero LDA para reducir la dimensionalidad de nuestros datos y luego Random Forest como clasificador, obtuvimos una precisión del 0.8604, mejorando sensiblemente al primer modelo. La tasa de error sobre las clase mas pequeña, 'pobre', es levemente mayor al primer modelo, con un 92% (12/13) de inexactitud. A su vez para la clase 'bueno' la tasa de error se redujo a 42% (30/70). Al igual que el primer modelo, cuanto mayor el tamaño de la clase, mejor la precisión del modelo, acierta el 93% de las veces.\n\nEl tercer modelo donde el algoritmo Random Forest funciona como predictor sobre el conjunto de datos original, obtuvo una precisión general del 0.8688, ligeramente superior a los dos modelos anteriores. Sin embargo, demostró peor sensibilidad ante clases minoritarias, con un 100% de ineficacia para el grupo 'pobre', y un 52% para 'bueno'. Como contrapartida demuestra gran eficiencia en la clasificación del grupo 'aceptable', el cual cuenta con 397 observaciones.\n\nConsiderando los tres modelos y sus características parece adecuado seleccionar al segundo de ellos, donde se realizó la reducción de dimensiones con LDA y se utilizó el algoritmo de Random Forest para clasificar, como el mejor de ellos. Éste modelo parece balancear mejor su precision en clases minoritarias, siendo mas preciso que sus contrincantes al predecir ocurrencias de la clase 'bueno', y siendo levemente mas impreciso para detectar 'pobres' que el modelo primero."

## 1.4 Puntuación del del ejercicio

Este ejercicio se puntuará con 10 puntos, siendo el mínimo necesario para superar la prueba de 5 puntos. La puntuación es la siguiente:

* Ejercicio 1: 1 punto
* Ejercicio 2: 1 punto
* Ejercicio 3: 1 punto
* Ejercicio 4: 1.5 puntos
* Ejercicio 5: 1.5 puntos
* Ejercicio 6: 1 punto
* Ejercicio 7: 1.5 puntos
* Ejercicio 8: 1 punto
* Ejercicio 9: 0.5 puntos