Regresión Lineal Múltiple y Análisis Discriminante

En primer lugar, cargaremos las librerías que nos hacen falta para poder realizar todo el análisis posterior, en concreto Regresión Lineal Múltiple y Análisis Discriminante.

```
library(knitr)
library(dplyr)
library(psych)
library(GGally)
library(corrplot)
library(ggplot2)
library(gridExtra)
library(caret)
library(MASS)
```

REGRESIÓN LINEAL MÚLTIPLE

A continuación, pasaremos a cargar los datos que utlizaremos para realizar estas técnicas que nos van a permetir predecir una variable. En este caso, la variable dependendiente será la popularidad, y el objetivo es intentar predecirla a partir de varias variables explicativas numéricas que recogen información sobre las características de la canción. Los datos seleccionados corresponden a todo el conjunto a excepción de las canciones clásicas, ya que este genero difiere significativamente del resto respecto a sus características y no nos interesa valorar esta opción ya que la popularidad de este tipo de canciones es baja y también buscamos encontrar las características que contribuyen más a tener una alta popularidad.

En primer lugar, cargamos los datos:

```
load("C:/Users/losaa/OneDrive/Escritorio/Estudios/Proyecto II/ncdata.RData")
load("C:/Users/losaa/OneDrive/Escritorio/Estudios/Proyecto II/desc_data2.RData")
```

Seguidamente, nos quedamos con las numéricas y quitamos la variable year del conjunto de datos, ya que el año en el que se sacó la canción no queremos que sea una de las características que nos ayuden a predecir la popularidad debido a que estamos buscando las características que nos ayuden a tener un alto valor de la variable respuesta independientemente del tiempo en el que salió la canción. Además, hemos decidido eliminar del conjunto de datos a todas las canciones con una popularidad 0 debido a que existen muchas canciones con esta popularidad y no sabemos si es debido a un fallo o¡en los datos o no y mantenerlo en nuestro conjunto de datos no nos va a ayudar a predecir valores altos de popularidad.

```
variablesnc = desc_data$variable[desc_data$type == 'numerical']
variablesnc = variablesnc[-15]
datos = ncdata[,variablesnc]
datos = datos[,-c(12)]
numero <- datos[datos$popularity == 0, ]
datos <- datos[datos$popularity != 0, ]
```

A continuación vamos a ver el número de filas que hemos eliminado por tener valor de popularidad 0:

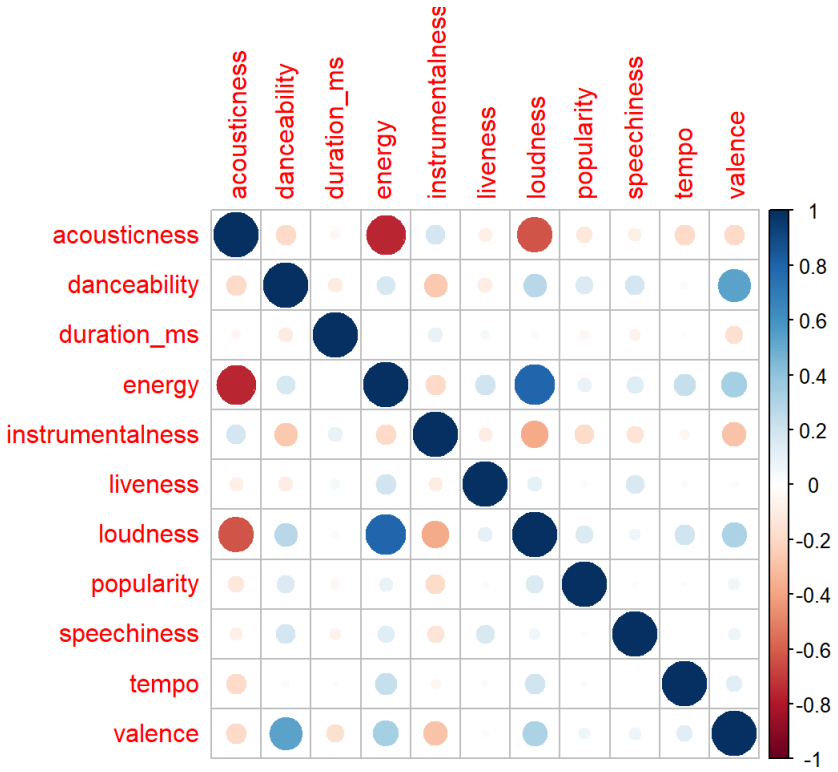
```
nrow(numero)
```

```
## [1] 470224
```

Como podemos ver casi medio millón de canciones tenían un valor 0 en la casilla de popolaridad por lo que cualquier modelo se podría ver influido a obtener valores bajos de popularidad ya que hay una gran cantidad de canciones con valor 0.

Vamos a hacer una ejecución de código que nos va a permitir saber que variables están más o menos correlacionadas. El siguiente gráfico nos permite obtener una idea visual rápida de que variables se ven más o menos correlacionadas.

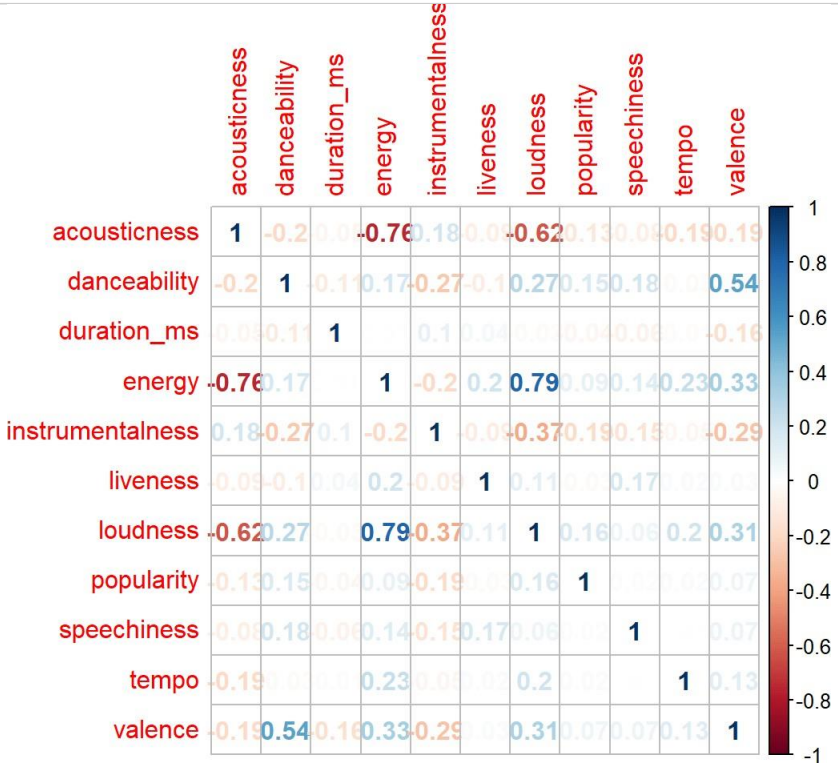
```
M <- round(cor(x = datos, method = "pearson"), 3)
corrplot(M, method = "circle")
```



Como podemos observar acousticness está relacionado negativamente con energy y loudness y estas dos últimas están correlacionadas positivamente, por lo que a mayor volumen tenga la canción(loudness) mayor será la energía de la misma. Además, simplificando lo dicho al principio, cuanto más acústica sea la canción menor volumen y menor energía tendrá. También, aunque el coeficiente de correlación es menor, podemos ver que a mayor danzabilidad tiene la canción mayor positividad transmite (valence).

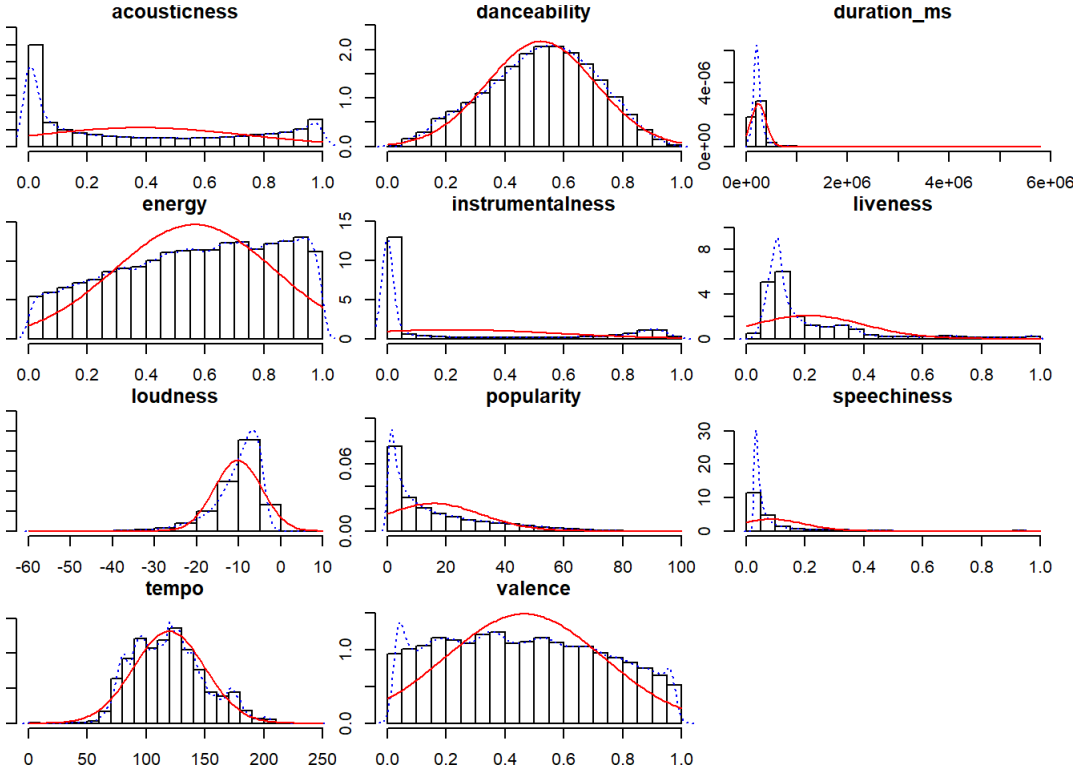
Como ya sabemos que variables están más correlacionadas, nos vamos a fijar en el valor que toma el coeficiente de correlación de estas correlaciones mediante el siguiente gráfico y de esta manera cuantificar dicha relación.

```
corrplot(M, method = "number")
```



Seguidamente, vamos a ver la distribución de cada variable:

```
multi.hist(x = datos, dcol = c("blue", "red"), dlty = c("dotted", "solid"), main = colnames(datos))
```



Podemos ver normalidad en la danzabilidad y en el tempo de la canción, largas colas en instrumentalness, speechness, liveness, popularity y loudness. Aquí destacamos que popularity tiene más valores bajos y por tanto vemos la cola a la parte derecha. Además, valence tiene una distribución más o menos uniforme y energía tiene más valores altos que bajos, no obstante la distribución se acerca más a una uniforme.

Pasamos a crear modelos de regresión. En primer lugar probaremos con todas las variables del conjunto de datos y nos haremos una idea de los resultados que esto nos proporciona. De todas maneras, como ya hemos hecho un Análisis de Componentes Principales ya sabemos que variables contribuyen más a explicar la variabilidad de los datos y también sabemos aquellas que están más correlacionadas entre ellas y crearemos un modelo sin el problema de multicolinealidad.

```
modelo <- lm(popularity ~ energy + tempo + danceability + instrumentalness+ acousticness + speechiness + liveness + va
lence, data = datos )
summary(modelo)
```

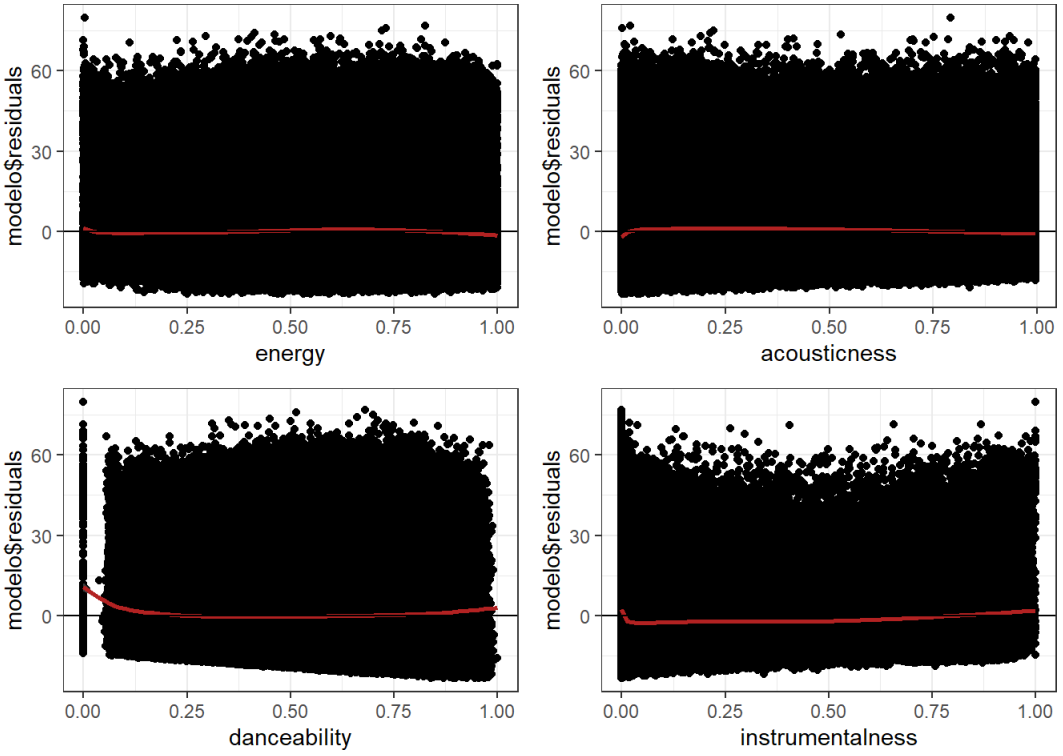
```
##
## Call:
## lm(formula = popularity ~ energy + tempo + danceability + instrumentalness +
##     acousticness + speechiness + liveness + valence, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.470 -11.963  -5.096   8.207  79.723
##
## Coefficients:
##                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)      15.044893   0.128596  116.993   <2e-16 ***
## energy            0.258996   0.110352    2.347   0.0189 *
## tempo             0.004333   0.000616    7.035    2e-12 ***
## danceability     11.349787   0.123040   92.244   <2e-16 ***
## instrumentalness -7.444198   0.055131 -135.027   <2e-16 ***
## acousticness     -3.832133   0.078974  -48.524   <2e-16 ***
## speechiness      -4.156239   0.166774  -24.921   <2e-16 ***
## liveness         -2.643745   0.098991  -26.707   <2e-16 ***
## valence          -3.949429   0.086328  -45.749   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.8 on 774187 degrees of freedom
## Multiple R-squared:  0.05651,    Adjusted R-squared:  0.05651
## F-statistic:  5797 on 8 and 774187 DF,  p-value: < 2.2e-16
```

Podemos observar que danceability, acousticness, instrumentalness y speechness tienen un valor absoluto del coeficiente obtenido más alto que el resto. De esto, podemos sacar que a mayor danzabilidad que tenga la canción mayor será su popularidad. Todas las variables resultan significativas pero somos conscientes de los posibles problemas de correlaciones entre las variables explicativas. Por último, cabe añadir que los datos tan sólo un 5'6% de la variabilidad en los datos por lo que concluimos que el modelo no es bueno.

Pasaremos a hacer un análisis de los residuos por si encontramos problemas de heterocedasticidad o por si vemos que los residuos tienen una determinada forma que no es captada por el modelo. Escogemos las variables que por lo visto en PCA y en los resultados de este modelo generado, tal vez son más influyentes y nos pueden ayudar en mayor proporción a predecir la popularidad.

```
plot1 <- ggplot(data = datos, aes(energy, modelo$residuals)) +
    geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
    theme_bw()
plot2 <- ggplot(data = datos, aes(acousticness, modelo$residuals)) +
    geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
    theme_bw()
plot3 <- ggplot(data = datos, aes(danceability, modelo$residuals)) +
    geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
    theme_bw()
plot4 <- ggplot(data = datos, aes(instrumentalness, modelo$residuals)) +
    geom_point() + geom_smooth(color = "firebrick") + geom_hline(yintercept = 0) +
    theme_bw()
grid.arrange(plot1, plot2, plot3, plot4)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Como podemos ver en los resultados los datos se distribuyen por todo el gráfico en todos los casos. Esto nos ayuda a saber que las diferencias entre los datos reales y predichos pueden variar mucho entre las diferentes canciones. Esto no nos ayuda mucho a encontrar un patrón ni a encontrar algún problema, pero si a saber que entre las canciones existe un gran variedad y que sacar información de los datos va a ser una tarea complicada, al menos realizando una regresión lineal múltiple.

A continuación, nos disponemos a crear 3 modelos más con las variables que consideramos que pueden ser las mejores. Además, después de ver los resultados, concluiremos si la regresión lineal múltiple es buena idea o no con este conjunto de datos.

```
modelo2 <- lm(popularity ~ energy + danceability + acousticness , data = datos )
summary(modelo2)
```

```
##
## Call:
## lm(formula = popularity ~ energy + danceability + acousticness,
##     data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.306 -11.951  -5.757   8.257  78.563
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  12.12172    0.10034 120.812   <2e-16 ***
## energy       -0.93081    0.10294  -9.042   <2e-16 ***
## danceability 11.67685    0.10077 115.875   <2e-16 ***
## acousticness -5.06890    0.07808 -64.922   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.99 on 774192 degrees of freedom
## Multiple R-squared:  0.03281,    Adjusted R-squared:  0.0328
## F-statistic:  8753 on 3 and 774192 DF,  p-value: < 2.2e-16
```

```
modelo3 <- lm(popularity ~ energy + danceability + acousticness + duration_ms, data = datos )
summary(modelo3)
```

```
##
## Call:
## lm(formula = popularity ~ energy + danceability + acousticness +
##     duration_ms, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.549 -11.960  -5.713   8.253  78.348
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.338e+01  1.083e-01  123.51   <2e-16 ***
## energy       -1.047e+00  1.030e-01  -10.17   <2e-16 ***
## danceability  1.131e+01  1.014e-01  111.45   <2e-16 ***
## acousticness -5.246e+00  7.824e-02  -67.05   <2e-16 ***
## duration_ms  -3.815e-06  1.244e-07  -30.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.98 on 774191 degrees of freedom
## Multiple R-squared:  0.03398,    Adjusted R-squared:  0.03397
## F-statistic:  6808 on 4 and 774191 DF,  p-value: < 2.2e-16
```

```
modelo4 <- lm(popularity ~ energy + danceability + duration_ms , data = datos )
summary(modelo4)
```

```
##
## Call:
## lm(formula = popularity ~ energy + danceability + duration_ms,
##     data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.530 -12.089  -5.804   8.266  79.508
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.904e+00  7.139e-02  110.71   <2e-16 ***
## energy        4.142e+00  6.810e-02   60.82   <2e-16 ***
## danceability  1.207e+01  1.011e-01  119.43   <2e-16 ***
## duration_ms  -3.200e-06  1.245e-07  -25.71   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.03 on 774192 degrees of freedom
## Multiple R-squared:  0.02837,    Adjusted R-squared:  0.02837
## F-statistic:  7535 on 3 and 774192 DF,  p-value: < 2.2e-16
```

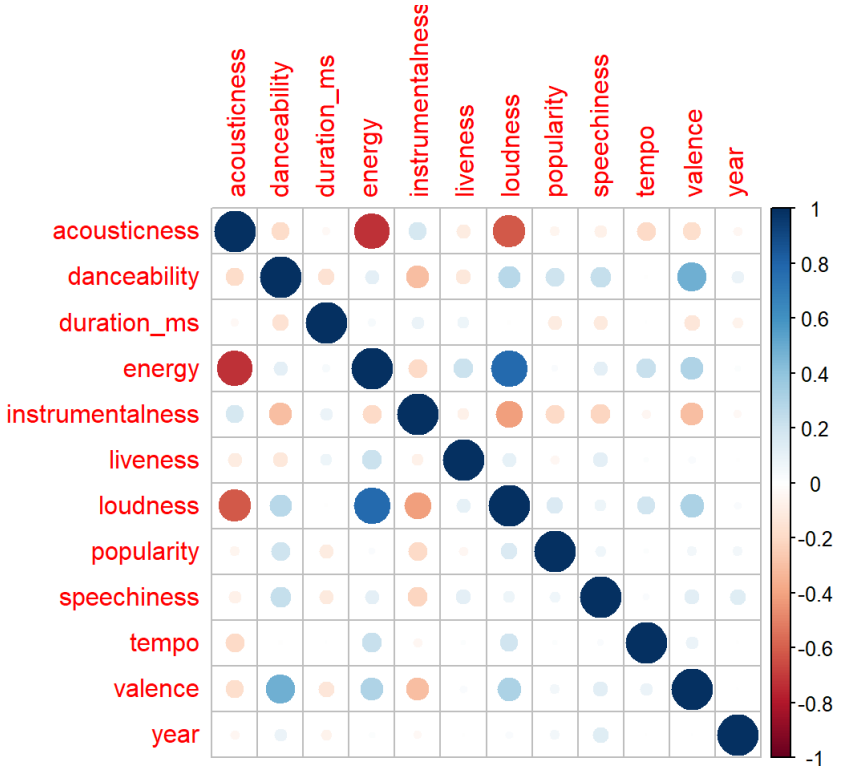
Como podemos ver en los resultados el R-squared explica un porcentaje de variabilidad muy bajo, inferior al 5% en todos los casos. Con esto concluimos que para este conjunto de datos no es una buena idea utilizar la regresión lineal múltiple. Es por este motivo, que vamos a reducir nuestros datos a un subconjunto muy pequeño en el que los datos son más actuales y pensamos que, tal vez, hay una menor varianza.

Ahora volveremos a realizar el proceso pero filtrando a datos más actuales, es decir, los datos de los últimos 5 años.

```
variablesnc = desc_data$variable[desc_data$type == 'numerical']
datos = ncdata[,variablesnc]
datos <- datos[datos$popularity != 0, ]
datos  = datos[datos$year <= 2021 & datos$year > 2016,]
```

Pasaremos a ver las correlaciones existentes entre las variables.

```
M <- round(cor(x = datos, method = "pearson"), 3)
corrplot(M, method = "circle")
```

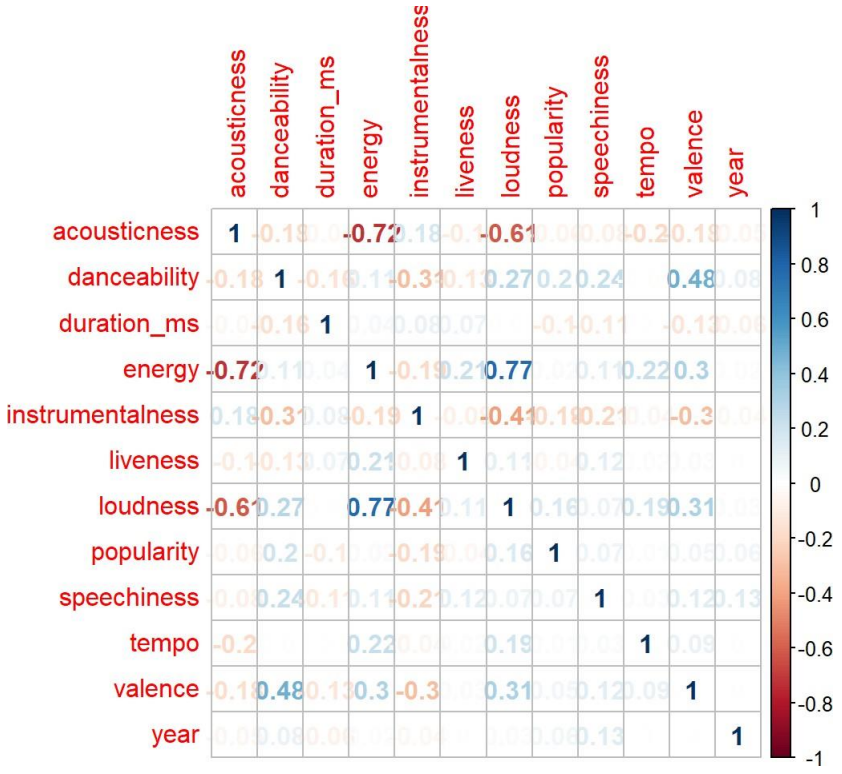


Como podemos observar, sucede lo mismo que con el conjunto de todos los datos que usamos anteriormente. Asi que las conclusiones son las mismas. No on¡bstante, las comentaremos igual, y por tanto, podemos decir que acousticness está relacionado negativamente con energy y loudness y estas dos últimas están correlacionadas positivamente, por lo que a mayor volumen tenga la canción(loudness) mayor

será la energía de la misma. Además, simplificando lo dicho al principio, cuanto más acústica sea la canción menor volumen y menor energía tendrá. También, aunque el coeficiente de correlación es menor, podemos ver que a mayor danzabilidad tiene la canción mayor positividad transmite (valence).

Pasaremos a ver los coeficientes de corrrelación de manera numérica de estos pares mencionados en el siguiente gráfico.

```
corrplot(M, method = "number")
```



Ahora vamos a crear los 4 modelos, igual que hemos hecho anteriormente y nos vamos a fijar si se consigue explicar un porcentaje de variavilidad más alto.

```
modelo <- lm(popularity ~ energy + tempo + danceability + instrumentalness+ acousticness + speechiness + liveness + va
lence, data = datos )
summary(modelo)
```

```
##
## Call:
## lm(formula = popularity ~ energy + tempo + danceability + instrumentalness +
##     acousticness + speechiness + liveness + valence, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.312 -13.166  -4.403   9.874  79.176
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      13.961886   0.298338  46.799  < 2e-16 ***
## energy           -0.865195   0.254757  -3.396 0.000684 ***
## tempo             0.008661   0.001389   6.235 4.54e-10 ***
## danceability     17.854323   0.263966  67.639  < 2e-16 ***
## instrumentalness -8.081209   0.127941 -63.163  < 2e-16 ***
## acousticness     -0.947868   0.183821  -5.156 2.52e-07 ***
## speechiness       0.144566   0.343164   0.421 0.673556
## liveness         -2.734929   0.237430 -11.519  < 2e-16 ***
## valence          -6.459330   0.200671 -32.189  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.93 on 171889 degrees of freedom
## Multiple R-squared:  0.06813,    Adjusted R-squared:  0.06809
## F-statistic:  1571 on 8 and 171889 DF,  p-value: < 2.2e-16
```

```
modelo2 <- lm(popularity ~ energy + danceability + acousticness , data = datos )
summary(modelo2)
```

```
##
## Call:
## lm(formula = popularity ~ energy + danceability + acousticness,
##     data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.132 -13.375  -4.835   9.935  77.266
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   10.7329     0.2348  45.708   <2e-16 ***
## energy        -2.0427     0.2399  -8.515   <2e-16 ***
## danceability  18.4742     0.2212  83.518   <2e-16 ***
## acousticness  -2.1825     0.1831 -11.917   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.16 on 171894 degrees of freedom
## Multiple R-squared:  0.04278,    Adjusted R-squared:  0.04277
## F-statistic:  2561 on 3 and 171894 DF,  p-value: < 2.2e-16
```

```
modelo3 <- lm(popularity ~ energy + danceability + acousticness + duration_ms, data = datos )
summary(modelo3)
```

```
##
## Call:
## lm(formula = popularity ~ energy + danceability + acousticness +
##     duration_ms, data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.468 -13.374  -4.774   9.942  76.880
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.333e+01  2.483e-01  53.658  < 2e-16 ***
## energy       -1.961e+00  2.392e-01  -8.196  2.5e-16 ***
## danceability 1.729e+01  2.238e-01  77.285  < 2e-16 ***
## acousticness -2.417e+00  1.828e-01 -13.225  < 2e-16 ***
## duration_ms  -8.296e-06  2.648e-07 -31.330  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.11 on 171893 degrees of freedom
## Multiple R-squared:  0.04822,    Adjusted R-squared:  0.0482
## F-statistic:  2177 on 4 and 171893 DF,  p-value: < 2.2e-16
```

```
modelo4 <- lm(popularity ~ energy + danceability + duration_ms , data = datos )
summary(modelo4)
```

```
##
## Call:
## lm(formula = popularity ~ energy + danceability + duration_ms,
##     data = datos)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.786 -13.403  -4.777   9.961  77.341
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.094e+01  1.707e-01  64.069   <2e-16 ***
## energy        3.112e-01  1.666e-01   1.868   0.0618 .
## danceability  1.775e+01  2.212e-01  80.249   <2e-16 ***
## duration_ms  -8.152e-06  2.647e-07 -30.798   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.12 on 171894 degrees of freedom
## Multiple R-squared:  0.04725,    Adjusted R-squared:  0.04723
## F-statistic:  2842 on 3 and 171894 DF,  p-value: < 2.2e-16
```

De los resultados obtenidos, llegamos a la conlusión que la máxima variabilidad explicada obtenido por uno de los modelos es de 6’8% por lo que finalmente, podemos decir que ajustar un modelo de regresión a estos datos no es muy buena idea. Esto se debe a que los datos son muy dispersos y que no se encuentra un patrón claro que ayude a predecir la popularidad.

Además de esto, también hemos estado probando otros modelos con datos sólo de 2020 y posteriormente con datos de 2020 filtrados por una popularidad mayor a 70. No obstante, en ambos casos los resultados obtenidos son los mismos, por lo que nos ahorraremos añadir este código.

ANÁLISIS DISCRIMINANTE

Vamos a proseguir con nuestra idea de predecir la popularidad, esta vez cambiando de técnica y utilizando un Análisis Discriminante. Como predecir exactamente un valor ha resultado ser complicado hemos decidido hacer un modelo de clasificación, de manera que vamos a recodificar la columna popularidad en una nueva columna con tres posibles valores, es decir, datos entre 0-30, 30-70 y >70. Por el mismo motivo que en el anterior análisis eliminaremos los datos con popularidad igual a 0 de nuestro conjunto de datos.

```
variablesnc = desc_data$variable[desc_data$type == 'numerical']
datos = ncdata[,variablesnc]
datos$recod[datos$popularity > 70] <- ">70"
```

```
## Warning: Unknown or uninitialised column: `recod`.
```

```
datos$recod[datos$popularity <= 70 & datos$popularity > 30] <- "30-70"
datos$recod[datos$popularity <= 30 & datos$popularity >= 0] <- "0-30"
datos <- datos[datos$popularity != 0, ]
```

Además utilizaremos una validación cruzada o cross-validation, dividiendo los datos en dos conjuntos, el de entrenamiento para crear el modelo y datos test para probarlo. Además, al igual que haremos con todos los modelos en los que utilicemos Análisis dDiscriminante, haremos 10 carpetas de datos y repetiremos esto 30 veces para asegurarnos de que los resultados que obtenemos son reales y no causa del azar.

```
set.seed(100)
trainFilas = createDataPartition(datos$recod, p=0.8, list=FALSE)
# trainFilas contiene los números de las filas que irán a Train
trainDatos = datos[trainFilas,]
testDatos = datos[-trainFilas,]
num = table(trainDatos$recod)
num
```

```
##
##    >70   0-30  30-70
##   2486 505914 110958
```

```
perc = 100*num/sum(num)
myTrainControl = trainControl(method = "repeatedcv",  # k-fold
                              number = 10,  # num folds
                              repeats = 30)
```

Como podemos ver, medio millón de filas comprenden entre 0 y 30 sus valores de popularidad, 110 mil filas entre 30 y 70 y unas 2.500 filas con popularidad mayor que 70. Esto nos puede dar problemas, ya que los datos están desbalanceados. No obstante, probaremos el modelo tanto en el conjunto de entrenamiento como en el de test.

```
set.seed(100)
trainDatosESC = trainDatos
trainDatos = trainDatos[, -8]
trainDatosESC = trainDatosESC[, -8]
#trainDatosESC[,-12] = scale(trainDatos[,-12], center = TRUE, scale =  TRUE)
modeloTR = train(recod ~ ., data = trainDatosESC, method='lda',
                 trControl = myTrainControl)  # preProcess = "scale"
modeloTR
```

```
## Linear Discriminant Analysis
##
## 619358 samples
##     11 predictor
##      3 classes: '>70', '0-30', '30-70'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 30 times)
## Summary of sample sizes: 557422, 557421, 557422, 557422, 557423, 557423, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8168316  -9.048426e-06
```

```
modeloTR$method
```

```
## [1] "lda"
```

```
modeloTR$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##         >70        0-30       30-70
## 0.004013834 0.816836143 0.179150023
##
## Group means:
##       acousticness danceability duration_ms    energy instrumentalness
## >70      0.2345609    0.6473230    217792.5 0.6360506       0.02496442
## 0-30     0.3985785    0.5115528    245007.1 0.5538752       0.24846495
## 30-70    0.3041661    0.5703983    234586.8 0.6049214       0.11174222
##        liveness   loudness speechiness    tempo   valence     year
## >70   0.1694485  -6.769097  0.09608749 120.5998 0.5056250 2010.409
## 0-30  0.2136872 -10.533622  0.08580814 119.2180 0.4569019 2004.436
## 30-70 0.1978441  -8.782169  0.08789337 120.5284 0.4993151 2002.587
##
## Coefficients of linear discriminants:
##                             LD1           LD2
## acousticness      1.084704e+00 -4.313385e-01
## danceability     -2.688076e+00 -1.771293e+00
## duration_ms       5.113124e-07  6.190175e-07
## energy            8.984961e-01  1.509458e+00
## instrumentalness  1.487249e+00 -2.937016e-01
## liveness          9.083757e-01  1.743345e-01
## loudness         -7.079723e-02 -7.967939e-02
## speechiness       7.288735e-01 -2.783491e-01
## tempo            -7.838344e-04  3.114668e-04
## valence           1.099898e+00  4.807227e-01
## year              2.018633e-02 -5.424138e-02
##
## Proportion of trace:
##    LD1    LD2
## 0.9676 0.0324
```

```
modeloTR$results
```

```
##   parameter  Accuracy         Kappa   AccuracySD      KappaSD
## 1      none 0.8168316 -9.048426e-06 1.152684e-05 1.598547e-05
```

Accuracy = 0.81; Kappa cercano a 0. Probabilidad a priori de obtener una popularidad entre 0 y 30 es de 0.81, lo que indica que hay muchos más datos en este intervalo que en el resto. El índice de Kappa se utiliza para cuando hay más de dos clases por lo que es un buen medidor del error de predicción. El índice de Kappa representa la proporción de acuerdos observados más allá del azar respecto del máximo acuerdo posible más allá del azar. En este caso el grado de acuerdo es insignificante. No obstante, la proporción de datos clasificados correctamente es muy elevada, esto es debido a que el modelo tiende a predecir practicamente todos los datos como en el intervalo 0-30 por lo que el modelo no tiene capacidad para predecir en otros intervalos y esto no nos interesa ya que nuestro objetivo es encontrar características que nos lleven a predecir datos altos de popularidad.

Valoraremos los resultados mediante una matriz de confusión y veremos los datos que se han clasificado correctamente y la cantidad de los que no lo han hecho.

```
ajusteTR = predict(modeloTR, type = "raw")
caret::confusionMatrix(ajusteTR,   factor(trainDatosESC$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    >70   0-30  30-70
##      >70        0      0      0
##      0-30    2486 505911 110958
##      30-70      0      3      0
##
## Overall Statistics
##
##                Accuracy : 0.8168
##                  95% CI : (0.8159, 0.8178)
##     No Information Rate : 0.8168
##     P-Value [Acc > NIR] : 0.5047
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: >70 Class: 0-30 Class: 30-70
## Sensitivity            0.000000      1.0000    0.000e+00
## Specificity            1.000000      0.0000    1.000e+00
## Pos Pred Value              NaN      0.8168    0.000e+00
## Neg Pred Value         0.995986      0.0000    8.208e-01
## Prevalence             0.004014      0.8168    1.792e-01
## Detection Rate         0.000000      0.8168    0.000e+00
## Detection Prevalence   0.000000      1.0000    4.844e-06
## Balanced Accuracy      0.500000      0.5000    5.000e-01
```

En estos resultados vemos lo comentado anteriormente, es decir, que el modelo tiende a predecir los datos con baja popularidad.

A continuación, probamos el modelo en los datos test:

```
testDatosESC = testDatos
testDatos = testDatos[, -8]
testDatosESC = testDatosESC[, -8]
#testDatosESC[,-12] = scale(testDatos[,-12], center = colMeans(trainDatos[,-12]), scale = apply(trainDatos[,-12], 2, s
d))
ajusteTest = predict(modeloTR, testDatosESC, type = "raw")
caret::confusionMatrix(ajusteTest,  factor(testDatos$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    >70   0-30  30-70
##      >70        0      0      0
##      0-30     621 126477  27739
##      30-70      0      1      0
##
## Overall Statistics
##
##                Accuracy : 0.8168
##                  95% CI : (0.8149, 0.8188)
##    No Information Rate : 0.8168
##    P-Value [Acc > NIR] : 0.5042
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: >70 Class: 0-30 Class: 30-70
## Sensitivity            0.000000      1.0000    0.000e+00
## Specificity            1.000000      0.0000    1.000e+00
## Pos Pred Value              NaN      0.8168    0.000e+00
## Neg Pred Value         0.995989      0.0000    8.209e-01
## Prevalence             0.004011      0.8168    1.791e-01
## Detection Rate         0.000000      0.8168    0.000e+00
## Detection Prevalence   0.000000      1.0000    6.458e-06
## Balanced Accuracy      0.500000      0.5000    5.000e-01
```

Con la curva ROC veremos el area debajo de la curva que nos permitirá hacer una representación de la proporción de verdaderos positivos frente a la proporción de falsos positivos.

```
library(pROC)
```

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

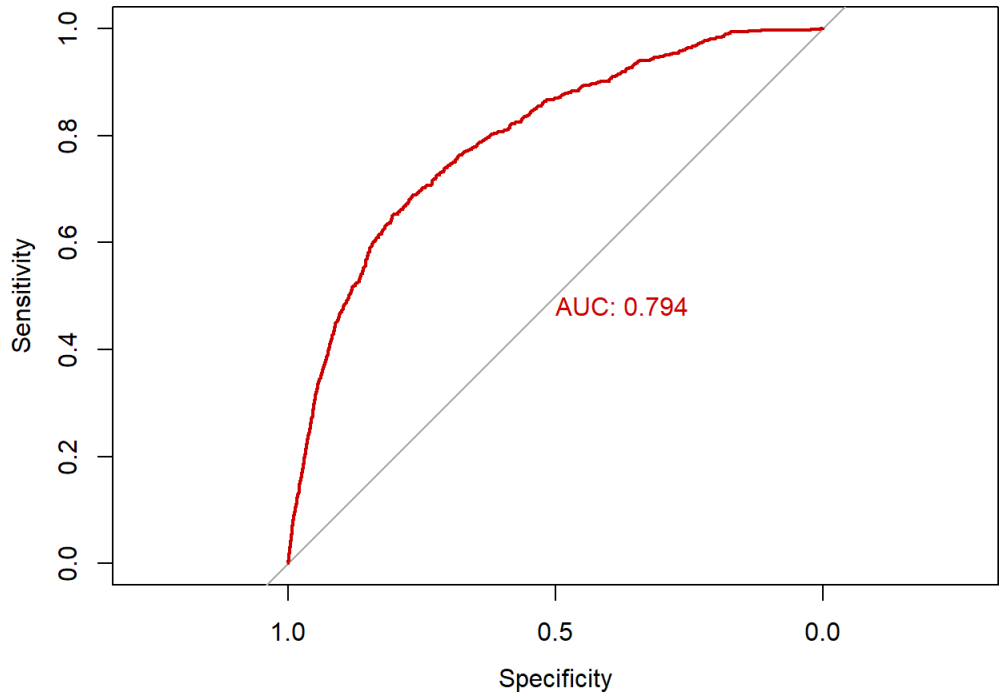
```
##
## Attaching package: 'pROC'
```

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

```
Y = 1*(testDatos$recod == ">70")
ajusteTestProb = predict(modeloTR, testDatosESC, type = "prob")
roc(Y ~ ajusteTestProb[,">70"], plot = TRUE, print.auc = TRUE, col = "red3")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.formula(formula = Y ~ ajusteTestProb[, ">70"], plot = TRUE,     print.auc = TRUE, col = "red3")
##
## Data: ajusteTestProb[, ">70"] in 154217 controls (Y 0) < 621 cases (Y 1).
## Area under the curve: 0.7941
```

El valor de AUC nos permite afirmar que el test es bueno, no obstante lo descartaremos por todo lo dicho anteriormente.

Debido a que hemos visto que los resultados se inclinan por predecir la mayoría de las veces una popularidad baja a causa del desbalanceo en los datos, probaremos recodificar la nueva columna creada de una manera que los datos estén balanceados. Por tanto, vamos a crear dos grupos, uno entre 0-10 de popularidad y otro con los datos restantes. El objetivo será hacer otro análisis discriminante si se predice un valor mayor de 10 y volver a hacer otra partición con este subconjunto de datos, por ejmplo de 10-30 y mayores de 30. No obstante, primero nos centraremos en ver si obtenemos unos resultados positivos.

```
variablesnc = desc_data$variable[desc_data$type == 'numerical']
datos = ncdata[,variablesnc]
datos$recod[datos$popularity > 10] <- ">10"
```

```
## Warning: Unknown or uninitialised column: `recod`.
```

```
datos$recod[datos$popularity <= 10 & datos$popularity >= 0] <- "0-10"
datos <- datos[datos$popularity != 0, ]
```

```
set.seed(100)
trainFilas = createDataPartition(datos$recod, p=0.8, list=FALSE)
# trainFilas contiene los números de las filas que irán a Train
trainDatos = datos[trainFilas,]
testDatos = datos[-trainFilas,]
num = table(trainDatos$recod)
num
```

```
##
##    >10   0-10
## 293580 325777
```

```
perc = 100*num/sum(num)
myTrainControl = trainControl(method = "repeatedcv",  # k-fold
                              number = 10,  # num folds
                              repeats = 30)
```

Ahora si que vemos que los datos están balanceados, por lo que vamos a pasar a la creación del modelo y a su posterior valoración.

```
set.seed(100)
trainDatosESC = trainDatos
trainDatos = trainDatos[, -8]
trainDatosESC = trainDatosESC[, -8]
#trainDatosESC[,-12] = scale(trainDatos[,-12], center = TRUE, scale =  TRUE)
modeloTR = train(recod ~ ., data = trainDatosESC, method='lda',
                 trControl = myTrainControl)  # preProcess = "scale"
modeloTR
```

```
## Linear Discriminant Analysis
##
## 619357 samples
##     11 predictor
##      2 classes: '>10', '0-10'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 30 times)
## Summary of sample sizes: 557421, 557422, 557421, 557422, 557421, 557422, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6072827  0.2111086
```

```
modeloTR$method
```

```
## [1] "lda"
```

```
modeloTR$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##       >10      0-10
## 0.4740077 0.5259923
##
## Group means:
##      acousticness danceability duration_ms    energy instrumentalness  liveness
## >10     0.3421828    0.5438653    236939.5 0.5849084        0.1682533 0.2098372
## 0-10    0.4157674    0.5032950    248708.8 0.5440657        0.2720156 0.2122605
##        loudness speechiness    tempo  valence     year
## >10   -9.489257  0.08794301 120.0083 0.477974 2002.612
## 0-10 -10.848751  0.08454411 118.8795 0.452946 2005.515
##
## Coefficients of linear discriminants:
##                            LD1
## acousticness      1.120773e+00
## danceability     -2.448799e+00
## duration_ms       9.603749e-07
## energy            7.501306e-01
## instrumentalness  1.248413e+00
## liveness          2.795637e-01
## loudness         -7.478006e-02
## speechiness       3.145847e-01
## tempo            -1.197541e-03
## valence           1.549929e+00
## year              3.738102e-02
```

```
modeloTR$results
```

```
##   parameter  Accuracy     Kappa AccuracySD     KappaSD
## 1      none 0.6072827 0.2111086 0.00176599 0.003533826
```

Accuracy 0’60; Kappa = 0’21.

Como podemos ver, las conclusiones cambian respecto a las del modelo anterior. Si que es verdad que la proporción de datos clasificados correctamente es menor, pero a un así clasifica el 60% correctamente y el índice de kappa muestra un grado de acuerdo mediano. No obstante, lo más importante de este modelo es que parece ser predice muchos más datos con popularidad elevada. Esto lo veremos en la sensibilidad.

Vamos a interpretar a continuación la matriz de confusión de los datos de entrenamiento:

```
ajusteTR = predict(modeloTR, type = "raw")
caret::confusionMatrix(ajusteTR,   factor(trainDatosESC$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    >10   0-10
##       >10  166967 116594
##       0-10 126613 209183
##
##                Accuracy : 0.6073
##                  95% CI : (0.6061, 0.6085)
##     No Information Rate : 0.526
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2112
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5687
##             Specificity : 0.6421
##          Pos Pred Value : 0.5888
##          Neg Pred Value : 0.6229
##              Prevalence : 0.4740
##          Detection Rate : 0.2696
##    Detection Prevalence : 0.4578
##       Balanced Accuracy : 0.6054
##
##        'Positive' Class : >10
```

Destacamos que el 60% de los datos son clasificados correctamente y sobretodo que la sensibilidad es de 0’56 por lo que vemos que nos ayuda predecir mucha más cantidad de datos con popularidad elevada.

Pasaremos a valorar los resultados del modelo en el conjunto de datos test.

```
testDatosESC = testDatos
testDatos = testDatos[, -8]
testDatosESC = testDatosESC[, -8]
#testDatosESC[,-12] = scale(testDatos[,-12], center = colMeans(trainDatos[,-12]), scale = apply(trainDatos[,-12], 2, s
d))
ajusteTest = predict(modeloTR, testDatosESC, type = "raw")
caret::confusionMatrix(ajusteTest,  factor(testDatos$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   >10  0-10
##       >10  41497 29146
##       0-10 31898 52298
##
##                Accuracy : 0.6058
##                  95% CI : (0.6033, 0.6082)
##     No Information Rate : 0.526
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2079
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5654
##             Specificity : 0.6421
##          Pos Pred Value : 0.5874
##          Neg Pred Value : 0.6211
##              Prevalence : 0.4740
##          Detection Rate : 0.2680
##    Detection Prevalence : 0.4562
##       Balanced Accuracy : 0.6038
##
##        'Positive' Class : >10
##
```

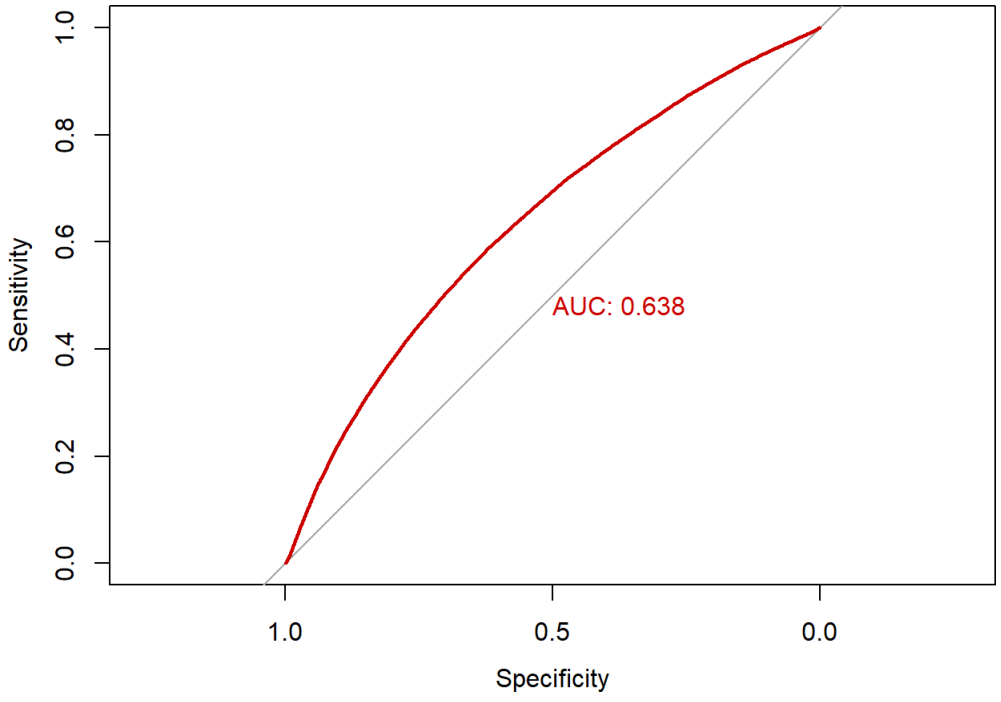
De la misma manera que en los datos de entrenamiento, podemos destacar que el 60% de los datos son clasificados correctamente.

A continuación, veremos las curva ROC y al area debajo de la curva que mide qué tan bien se clasifican las predicciones.

```
library(pROC)
Y = 1*(testDatos$recod == ">10")
ajusteTestProb = predict(modeloTR, testDatosESC, type = "prob")
roc(Y ~ ajusteTestProb[,">10"], plot = TRUE, print.auc = TRUE, col = "red3")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
##
## Call:
## roc.formula(formula = Y ~ ajusteTestProb[, ">10"], plot = TRUE,     print.auc = TRUE, col = "red3")
##
## Data: ajusteTestProb[, ">10"] in 81444 controls (Y 0) < 73395 cases (Y 1).
## Area under the curve: 0.6382
```

AUC = 0.63 – modelo regular (0’60-0’75).

A continuación, probaremos a crear un modelo creando una nueva variable explicada con tres valores balanceados por si los resultados obtenidos fueran mejores. En caso de obtener peores resultados que con el modelo anterior, seguiremos con la idea de hacer un Análisis Discriminante dentro del Análisis Discriminante, es decir, coger el conjunto de datos con popularidad mayor de 10 y volver a partir los datos en dos grupos.

```
variablesnc = desc_data$variable[desc_data$type == 'numerical']
datos = ncdata[,variablesnc]
datos$recod[datos$popularity > 20] <- ">20"
```

```
## Warning: Unknown or uninitialised column: `recod`.
```

```
datos$recod[datos$popularity <= 20 & datos$popularity > 5] <- "5-20"
datos$recod[datos$popularity <= 5 & datos$popularity >= 0] <- "0-5"
datos <- datos[datos$popularity != 0, ]
```

```
set.seed(100)
trainFilas = createDataPartition(datos$recod, p=0.8, list=FALSE)
# trainFilas contiene los números de las filas que irán a Train
trainDatos = datos[trainFilas,]
testDatos = datos[-trainFilas,]
num = table(trainDatos$recod)
num
```

```
##
##    >20    0-5   5-20
## 183786 232968 202604
```

```
perc = 100*num/sum(num)
myTrainControl = trainControl(method = "repeatedcv",  # k-fold
                              number = 10,  # num folds
                              repeats = 30)
```

Como vemos los datos están balanceados en los tres grupos.

```
set.seed(100)
trainDatosESC = trainDatos
trainDatos = trainDatos[, -8]
trainDatosESC = trainDatosESC[, -8]
trainDatosESC[,-12] = scale(trainDatos[,-12], center = TRUE, scale =  TRUE)
modeloTR = train(recod ~ ., data = trainDatosESC, method='lda',
                 trControl = myTrainControl)  # preProcess = "scale"
modeloTR
```

```
## Linear Discriminant Analysis
##
## 619358 samples
##     11 predictor
##      3 classes: '>20', '0-5', '5-20'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 30 times)
## Summary of sample sizes: 557422, 557421, 557422, 557423, 557422, 557423, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.4386537  0.1434981
```

```
modeloTR$method
```

```
## [1] "lda"
```

```
modeloTR$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##       >20       0-5      5-20
## 0.2967363 0.3761443 0.3271194
##
## Group means:
##      acousticness danceability duration_ms        energy instrumentalness
## >20  -0.162204407    0.1867889 -0.05259117  0.1156275356      -0.24676211
## 0-5   0.121008512   -0.1248644  0.05169912 -0.0916894689       0.16235143
## 5-20  0.007994847   -0.0258622 -0.01174074  0.0005428814       0.03715985
##          liveness     loudness  speechiness        tempo     valence
## >20  -0.028360054  0.186012669  0.016496749  0.027521588  0.09000885
## 0-5  -0.001484924 -0.145760172 -0.020086726 -0.020187749 -0.04922818
## 5-20  0.027433420 -0.001130622  0.008132579 -0.001752102 -0.02504282
##              year
## >20  -0.130331874
## 0-5   0.107276728
## 5-20 -0.005127593
##
## Coefficients of linear discriminants:
##                           LD1          LD2
## acousticness      0.41475507  0.036765592
## danceability     -0.46077995 -0.026426376
## duration_ms       0.13072428 -0.378631455
## energy            0.19984276 -0.169175075
## instrumentalness  0.45536248  0.643982801
## liveness          0.07185931  0.462328481
## loudness         -0.39845787  0.745809943
## speechiness       0.04925002  0.209816735
## tempo            -0.02955767 -0.009577793
## valence           0.39044703 -0.485613416
## year              0.57092027 -0.457098540
##
## Proportion of trace:
##    LD1    LD2
## 0.9766 0.0234
```

```
modeloTR$results
```

```
##   parameter  Accuracy     Kappa  AccuracySD     KappaSD
## 1      none 0.4386537 0.1434981 0.001602973 0.002436753
```

Podemos destacar que la accuracy del modelo es de 0’43 por lo que es menor que en el anterior modelo.

```
ajusteTR = predict(modeloTR, type = "raw")
caret::confusionMatrix(ajusteTR,   factor(trainDatosESC$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    >20    0-5   5-20
##       >20   94956  56227  68017
##       0-5   76188 161364 119181
##       5-20  12642  15377  15406
##
## Overall Statistics
##
##                Accuracy : 0.4387
##                  95% CI : (0.4375, 0.44)
##     No Information Rate : 0.3761
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1436
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: >20 Class: 0-5 Class: 5-20
## Sensitivity              0.5167     0.6926     0.07604
## Specificity              0.7148     0.4944     0.93277
## Pos Pred Value           0.4332     0.4523     0.35477
## Neg Pred Value           0.7780     0.7274     0.67497
## Prevalence               0.2967     0.3761     0.32712
## Detection Rate           0.1533     0.2605     0.02487
## Detection Prevalence     0.3539     0.5760     0.07011
## Balanced Accuracy        0.6157     0.5935     0.50440
```

Observamos que tanto Kappa como la proporción de datos clasificados correctamente es menor por lo que descartamos este modelo respecto del anterior, además de demostrar que tiene una menor sensibilidad.

Vemos que ocurre con los datos que nos hemos guardado para probar el modelo.

```
testDatosESC = testDatos
testDatos = testDatos[, -8]
testDatosESC = testDatosESC[, -8]
#testDatosESC[,-12] = scale(testDatos[,-12], center = colMeans(trainDatos[,-12]), scale = apply(trainDatos[,-12], 2, s
d))
ajusteTest = predict(modeloTR, testDatosESC, type = "raw")
caret::confusionMatrix(ajusteTest,  factor(testDatos$recod))
```

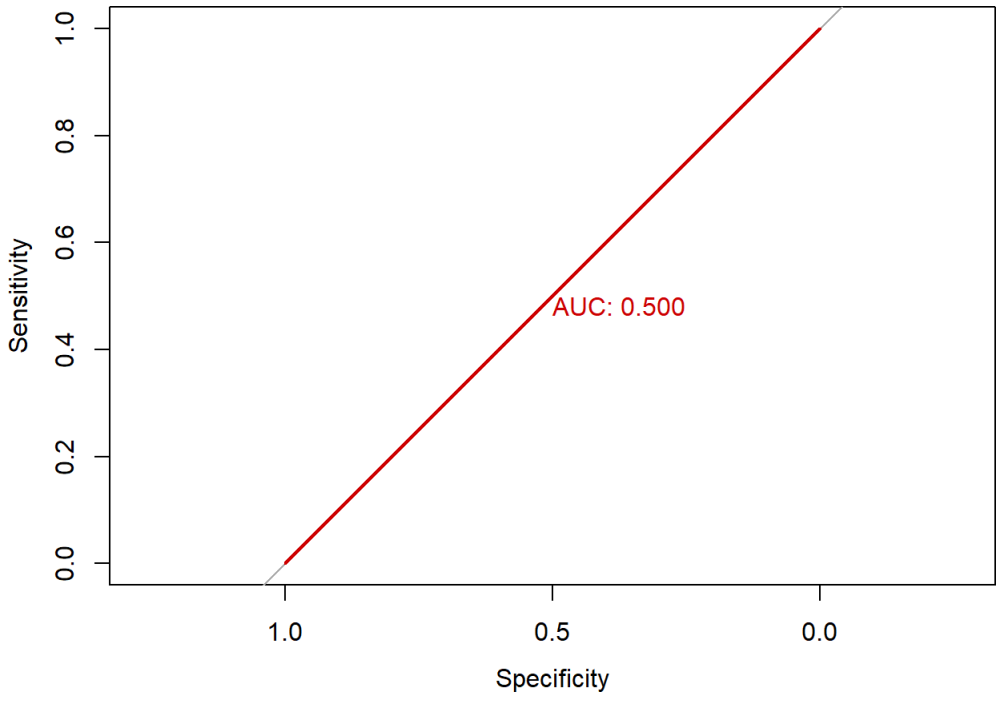
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   >20   0-5  5-20
##       >20      0     0     0
##       0-5  45946 58242 50650
##       5-20     0     0     0
##
## Overall Statistics
##
##                Accuracy : 0.3761
##                  95% CI : (0.3737, 0.3786)
##     No Information Rate : 0.3761
##     P-Value [Acc > NIR] : 0.501
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: >20 Class: 0-5 Class: 5-20
## Sensitivity              0.0000     1.0000      0.0000
## Specificity              1.0000     0.0000      1.0000
## Pos Pred Value              NaN     0.3761         NaN
## Neg Pred Value           0.7033        NaN      0.6729
## Prevalence               0.2967     0.3761      0.3271
## Detection Rate           0.0000     0.3761      0.0000
## Detection Prevalence     0.0000     1.0000      0.0000
## Balanced Accuracy        0.5000     0.5000      0.5000
```

Obtenemos una accuracy de 0’43 por lo que vemos que el modelo mencionado anteriormente dividido en dos grupos balanceados clasifica más datos correctamente.

```
library(pROC)
Y = 1*(testDatos$recod == ">20")
ajusteTestProb = predict(modeloTR, testDatosESC, type = "prob")
roc(Y ~ ajusteTestProb[,">20"], plot = TRUE, print.auc = TRUE, col = "red3")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```



```
##
## Call:
## roc.formula(formula = Y ~ ajusteTestProb[, ">20"], plot = TRUE,     print.auc = TRUE, col = "red3")
##
## Data: ajusteTestProb[, ">20"] in 108892 controls (Y 0) < 45946 cases (Y 1).
## Area under the curve: 0.5
```

Por tanto, vamos a guardarnos el conjunto de datos filtrados por popularidad mayor que 10 y crearemos una nueva columna con dos clases, es decir, datos con popularidad entre 10-30 y >30.

```
variablesnc = desc_data$variable[desc_data$type == 'numerical']
datos = ncdata[,variablesnc]
datos$recod[datos$popularity > 30] <- ">30"
```

```
## Warning: Unknown or uninitialised column: `recod`.
```

```
datos$recod[datos$popularity <= 30 & datos$popularity > 10] <- "10-30"
datos$recod[datos$popularity <= 10 & datos$popularity >= 0] <- "0-10"
datos <- datos[datos$popularity > 10, ]
```

```
set.seed(100)
trainFilas = createDataPartition(datos$recod, p=0.8, list=FALSE)
#trainFilas contiene los números de las filas que irán a Train
trainDatos = datos[trainFilas,]
testDatos = datos[-trainFilas,]
num = table(trainDatos$recod)
num
```

```
##
##    >30  10-30
## 113444 180137
```

```
perc = 100*num/sum(num)
myTrainControl = trainControl(method = "repeatedcv",  # k-fold
                              number = 10,  # num folds
                              repeats = 30)
```

Como podemos ver los datos se encuentran más o menos balanceados, así que procedemos a generar el modelo.

```
set.seed(100)
trainDatosESC = trainDatos
trainDatos = trainDatos[, -8]
trainDatosESC = trainDatosESC[, -8]
trainDatosESC[,-12] = scale(trainDatos[,-12], center = TRUE, scale =  TRUE)
modeloTR = train(recod ~ ., data = trainDatosESC, method='lda',
                 trControl = myTrainControl)  # preProcess = "scale"
modeloTR
```

```
## Linear Discriminant Analysis
##
## 293581 samples
##     11 predictor
##      2 classes: '>30', '10-30'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 30 times)
## Summary of sample sizes: 264222, 264223, 264223, 264223, 264223, 264223, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6215862  0.05779337
```

```
modeloTR$method
```

```
## [1] "lda"
```

```
modeloTR$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##       >30     10-30
## 0.3864147 0.6135853
##
## Group means:
##       acousticness danceability duration_ms      energy instrumentalness
## >30    -0.11746805    0.1567904 -0.02163080  0.08202622       -0.1866097
## 10-30   0.07397728   -0.0987411  0.01362232 -0.05165726        0.1175203
##          liveness    loudness   speechiness        tempo     valence
## >30   -0.06274507  0.15015159  0.0002958398  0.015101976  0.08245155
## 10-30  0.03951465 -0.09456023 -0.0001863096 -0.009510698 -0.05192511
##               year
## >30    0.009191205
## 10-30 -0.005788300
##
## Coefficients of linear discriminants:
##                           LD1
## acousticness      0.329893140
## danceability     -0.450316641
## duration_ms       0.003476167
## energy            0.258427487
## instrumentalness  0.559469948
## liveness          0.236075613
## loudness         -0.394256233
## speechiness       0.141456215
## tempo            -0.012456812
## valence           0.153073987
## year              0.097420875
```

```
modeloTR$results
```

```
##   parameter  Accuracy      Kappa  AccuracySD     KappaSD
## 1      none 0.6215862 0.05779337 0.001461303 0.003671871
```

Vemos que el valor de Accuracy es bueno también en este subconjunto y en este Análisis Discriminante.

Ahora pasamos a valorar la matriz de confusión de los datos de entrenamiento.

```
ajusteTR = predict(modeloTR, type = "raw")
caret::confusionMatrix(ajusteTR,   factor(trainDatosESC$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    >30  10-30
##      >30    11015   8640
##      10-30 102429 171497
##
##                Accuracy : 0.6217
##                  95% CI : (0.6199, 0.6234)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.058
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.09710
##             Specificity : 0.95204
##          Pos Pred Value : 0.56042
##          Neg Pred Value : 0.62607
##              Prevalence : 0.38641
##          Detection Rate : 0.03752
##    Detection Prevalence : 0.06695
##       Balanced Accuracy : 0.52457
##
##        'Positive' Class : >30
```

No obstante en este caso la sensibilidad es baja, concretamente 0’10, por lo que este segundo modelo no tiene tanta potencia como el anterior para predecir datos con popularidad elevada. De todas formas, nos estamos dando cuenta que este tipo de método supervisado es mucho mejor en este conjunto de datos que un modelo de regresión y que tenemos un primer modelo ya que nos ayuda a predecir valores con popularidad mayor a 10 con un nivel de sensibilidad relativamente alto.

Seguidamente, valoraremos los resultados del modelo para los datos test.

```
testDatosESC = testDatos
testDatos = testDatos[, -8]
testDatosESC = testDatosESC[, -8]
testDatosESC[,-12] = scale(testDatos[,-12], center = colMeans(trainDatos[,-12]),
                     scale = apply(trainDatos[,-12], 2, sd))
ajusteTest = predict(modeloTR, testDatosESC, type = "raw")
caret::confusionMatrix(ajusteTest,  factor(testDatos$recod))
```

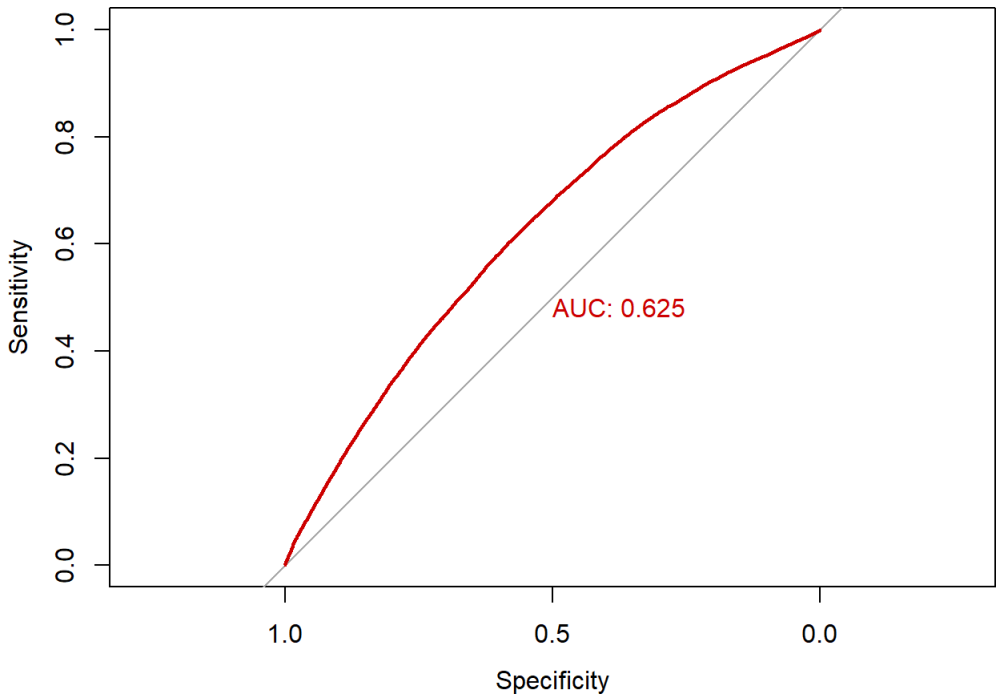
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   >30 10-30
##      >30    2722  2106
##      10-30 25638 42928
##
##                Accuracy : 0.622
##                  95% CI : (0.6185, 0.6255)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 1.491e-06
##
##                   Kappa : 0.0581
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.09598
##             Specificity : 0.95324
##          Pos Pred Value : 0.56379
##          Neg Pred Value : 0.62608
##              Prevalence : 0.38641
##          Detection Rate : 0.03709
##    Detection Prevalence : 0.06578
##       Balanced Accuracy : 0.52461
##
##        'Positive' Class : >30
```

A continuación, veremos las curva ROC y al area debajo de la curva que mide qué tan bien se clasifican las predicciones.

```
library(pROC)
Y = 1*(testDatos$recod == ">30")
ajusteTestProb = predict(modeloTR, testDatosESC, type = "prob")
roc(Y ~ ajusteTestProb[,">30"], plot = TRUE, print.auc = TRUE, col = "red3")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.formula(formula = Y ~ ajusteTestProb[, ">30"], plot = TRUE,     print.auc = TRUE, col = "red3")
##
## Data: ajusteTestProb[, ">30"] in 45034 controls (Y 0) < 28360 cases (Y 1).
## Area under the curve: 0.6249
```

En este proceso hemos creado modelos con todas las variables que tenemos en los datos como variables explicativas. No obstante, como sabemos, hay varias variables que están correlacionadas entre ellas, pudiendo generar problemas de multicolinealidad y además después de haber realizado PCA sabemos que hay varias variables que contribuyen a explicar una mayor variabilidad de los datos. Por tanto, crearemos un modelo juntando estas características y compararemos con el modelo creado anteriormente. Además, esto también lo hacemos porque aumentar la complejidad del modelo añadiendo muchas variables puede hacer que sobreajuste el modelo a los datos y que el modelo pierda capacidad predictora.

```
variablesnc = desc_data$variable[desc_data$type == 'numerical']
datos = ncdata[,variablesnc]
datos$recod[datos$popularity > 10] <- ">10"
```

```
## Warning: Unknown or uninitialised column: `recod`.
```

```
datos$recod[datos$popularity <= 10 & datos$popularity >= 0] <- "0-10"
datos <- datos[datos$popularity != 0, ]
```

```
set.seed(100)
trainFilas = createDataPartition(datos$recod, p=0.8, list=FALSE)
# trainFilas contiene los números de las filas que irán a Train
trainDatos = datos[trainFilas,]
testDatos = datos[-trainFilas,]
num = table(trainDatos$recod)
num
```

```
##
##    >10   0-10
## 293580 325777
```

```
perc = 100*num/sum(num)
myTrainControl = trainControl(method = "repeatedcv",  # k-fold
                              number = 10,  # num folds
                              repeats = 30)
```

Escogemos las variables energy, danceability, speechiness y instrumentalness para añadir al modelo ya que no están correlacionadas entre ellas y además son las que más variablidad de los datos explican. También pensamos que 4 como número de variables es un buen número ya que no genera un modelo complejo y así evitamos perder capacidad predictora.

```
set.seed(100)
trainDatosESC = trainDatos
trainDatos = trainDatos[, -8]
trainDatosESC = trainDatosESC[, -8]
#trainDatosESC[,-12] = scale(trainDatos[,-12], center = TRUE, scale =  TRUE)
modeloTR = train(recod ~ energy + danceability + speechiness + instrumentalness, data = trainDatosESC, method='lda',
                 trControl = myTrainControl)  # preProcess = "scale"
modeloTR
```

```
## Linear Discriminant Analysis
##
## 619357 samples
##      4 predictor
##      2 classes: '>10', '0-10'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 30 times)
## Summary of sample sizes: 557421, 557422, 557421, 557422, 557421, 557422, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5759172  0.1514728
```

```
modeloTR$method
```

```
## [1] "lda"
```

```
modeloTR$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##       >10      0-10
## 0.4740077 0.5259923
##
## Group means:
##         energy danceability speechiness instrumentalness
## >10  0.5849084    0.5438653  0.08794301        0.1682533
## 0-10 0.5440657    0.5032950  0.08454411        0.2720156
##
## Coefficients of linear discriminants:
##                           LD1
## energy           -0.8978962
## danceability     -2.3809555
## speechiness       1.1751710
## instrumentalness  2.1038989
```

```
modeloTR$results
```

```
##   parameter  Accuracy     Kappa AccuracySD     KappaSD
## 1      none 0.5759172 0.1514728  0.0019268 0.003832298
```

Accuracy = 0’57 ; Kappa = 0’15

Vemos que tanto Accuracy como Kappa son un poco menores que el modelo creado con todas las variables y esto puede ser debido a un sobreajuste de los datos, ya que tampoco varía mucho. No obstante, nos fijaremos en la potencia que tiene para predecir datos con popularidad alta mediante la sensibilidad y decidiremos que modelo nos puede ayudar más a cumplir nuestro objetivo.

```
ajusteTR = predict(modeloTR, type = "raw")
caret::confusionMatrix(ajusteTR,   factor(trainDatosESC$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    >10   0-10
##       >10  169046 138104
##       0-10 124534 187673
##
##                Accuracy : 0.576
##                  95% CI : (0.5747, 0.5772)
##     No Information Rate : 0.526
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1515
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5758
##             Specificity : 0.5761
##          Pos Pred Value : 0.5504
##          Neg Pred Value : 0.6011
##              Prevalence : 0.4740
##          Detection Rate : 0.2729
##    Detection Prevalence : 0.4959
##       Balanced Accuracy : 0.5759
##
##        'Positive' Class : >10
##
```

Como podemos ver en este resultado, la sensibilidad nos indica que este primer modelo con escasas variables sigue teniendo potencia para predecir valores mayores a 10 de popularidad. No obstante, validaremos esta afirmación con los datos que nos hemos guardado a modo test para probar el modelo.

```
testDatosESC = testDatos
testDatos = testDatos[, -8]
testDatosESC = testDatosESC[, -8]
#testDatosESC[,-12] = scale(testDatos[,-12], center = colMeans(trainDatos[,-12]), scale = apply(trainDatos[,-12], 2, s
d))
ajusteTest = predict(modeloTR, testDatosESC, type = "raw")
caret::confusionMatrix(ajusteTest,  factor(testDatos$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   >10  0-10
##       >10  42111 34454
##       0-10 31284 46990
##
##                Accuracy : 0.5754
##                  95% CI : (0.573, 0.5779)
##     No Information Rate : 0.526
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.1504
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.5738
##             Specificity : 0.5770
##          Pos Pred Value : 0.5500
##          Neg Pred Value : 0.6003
##              Prevalence : 0.4740
##          Detection Rate : 0.2720
##    Detection Prevalence : 0.4945
##       Balanced Accuracy : 0.5754
##
##        'Positive' Class : >10
##
```

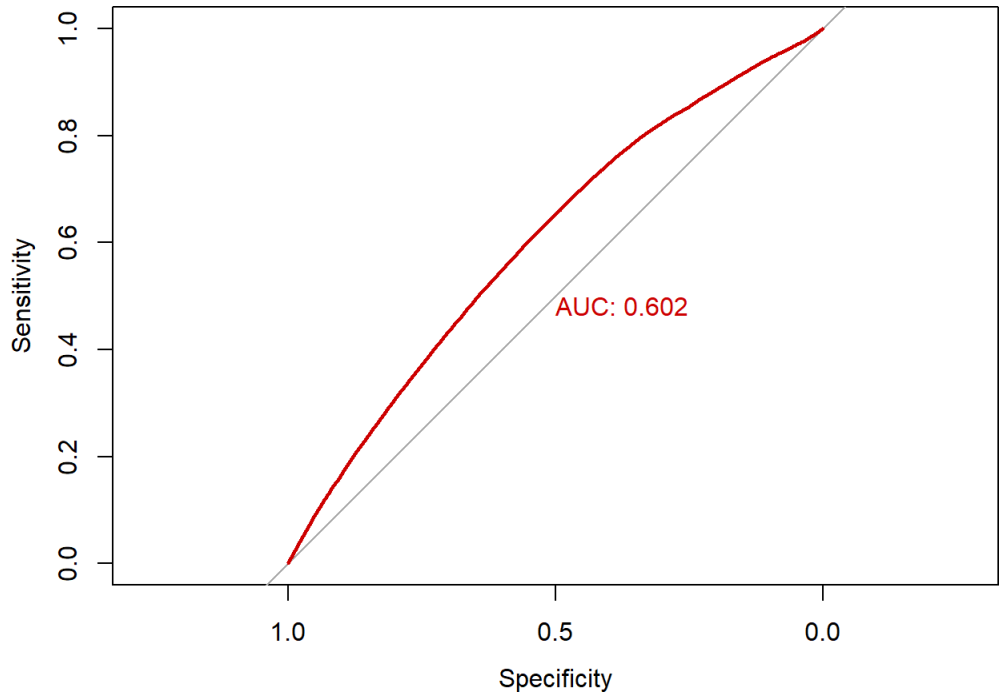
Probando el modelo con el conjunto de datos test. Podemos destacar que el 57% de los datos son clasificados correctamente y que la sensibilidad toma un valor también alto de 0’57 por lo que este modelo es tan útil como el generado con todas las variables, y en este caso con menor complejidad. Además nos ayuda en nuestro objetivo principal.

A continuación, veremos las curva ROC y al area debajo de la curva que mide qué tan bien se clasifican las predicciones.

```
library(pROC)
Y = 1*(testDatos$recod == ">10")
ajusteTestProb = predict(modeloTR, testDatosESC, type = "prob")
roc(Y ~ ajusteTestProb[,">10"], plot = TRUE, print.auc = TRUE, col = "red3")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.formula(formula = Y ~ ajusteTestProb[, ">10"], plot = TRUE,     print.auc = TRUE, col = "red3")
##
## Data: ajusteTestProb[, ">10"] in 81444 controls (Y 0) < 73395 cases (Y 1).
## Area under the curve: 0.6022
```

AUC = 0'602 — test regular (0'60-0'75)

Ahora vamos a proceder a hacer la segunda parte, es decir, vamos a hacer un Análisis Discriminante con el conjunto de datos mayores que 10 de popularidad.

```
variablesnc = desc_data$variable[desc_data$type == 'numerical']
datos = ncdata[,variablesnc]
datos$recod[datos$popularity > 30] <- ">30"
```

```
## Warning: Unknown or uninitialised column: `recod`.
```

```
datos$recod[datos$popularity <= 30 & datos$popularity > 10] <- "10-30"
datos$recod[datos$popularity <= 10 & datos$popularity >= 0] <- "0-10"
datos <- datos[datos$popularity > 10, ]
```

Creamos nuestros datos de entrenamiento y test:

```
set.seed(100)
trainFilas = createDataPartition(datos$recod, p=0.8, list=FALSE)
trainDatos = datos[trainFilas,]
testDatos = datos[-trainFilas,]
num = table(trainDatos$recod)
num
```

```
##
##    >30  10-30
## 113444 180137
```

```
perc = 100*num/sum(num)
myTrainControl = trainControl(method = "repeatedcv",  # k-fold
                              number = 10,  # num folds
                              repeats = 30)
```

Las clases están balanceadas por lo que evitamos tener este problema en el modelo. Pasamos a entrenar el modelo con las pocas variables que hemos escogido.

```
set.seed(100)
trainDatosESC = trainDatos
trainDatos = trainDatos[, -8]
trainDatosESC = trainDatosESC[, -8]
#trainDatosESC[,-12] = scale(trainDatos[,-12], center = TRUE, scale =  TRUE)
modeloTR = train(recod ~ energy + danceability + speechiness + instrumentalness, data = trainDatosESC, method='lda',
                 trControl = myTrainControl)  # preProcess = "scale"
modeloTR
```

```
## Linear Discriminant Analysis
##
## 293581 samples
##      4 predictor
##      2 classes: '>30', '10-30'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 30 times)
## Summary of sample sizes: 264222, 264223, 264223, 264223, 264223, 264223, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.6153431  0.020709
```

```
modeloTR$method
```

```
## [1] "lda"
```

```
modeloTR$finalModel
```

```
## Call:
## lda(x, grouping = y)
##
## Prior probabilities of groups:
##       >30     10-30
## 0.3864147 0.6135853
##
## Group means:
##          energy danceability speechiness instrumentalness
## >30   0.6055842    0.5724929  0.08806766        0.1096079
## 10-30 0.5712247    0.5262340  0.08801432        0.2052152
##
## Coefficients of linear discriminants:
##                         LD1
## energy           -0.7425176
## danceability     -2.9620567
## speechiness       2.0843726
## instrumentalness  2.1989115
```

```
modeloTR$results
```

```
##   parameter  Accuracy    Kappa   AccuracySD     KappaSD
## 1      none 0.6153431 0.020709 0.0009545985 0.002415289
```

Accuracy = 0’61 ; Kappa 0’02.

Como podemos ver los resultados en este segundo caso son muy parecidos al segundo caso con el modelo que contiene todas las variables.

```
ajusteTR = predict(modeloTR, type = "raw")
caret::confusionMatrix(ajusteTR,   factor(trainDatosESC$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    >30  10-30
##      >30     4361   3860
##      10-30 109083 176277
##
##                Accuracy : 0.6153
##                  95% CI : (0.6135, 0.6171)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 0.02889
##
##                   Kappa : 0.0205
##
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.03844
##             Specificity : 0.97857
##          Pos Pred Value : 0.53047
##          Neg Pred Value : 0.61774
##              Prevalence : 0.38641
##          Detection Rate : 0.01485
##    Detection Prevalence : 0.02800
##       Balanced Accuracy : 0.50851
##
##        'Positive' Class : >30
##
```

Vemos las mismas conclusiones en la matriz de conclusión. Así que pasamos a probar el modelo en los datos test. No obstante, por la experiencia adquirida los resultados no deberían variar mucho o practicamente nada con los datos de entrenamiento. Esto se ha podido deber a que ambos conjuntos son muy parecidos y se obtienen resultados muy similares. No obstante, creemos que cualquier dato nuevo introducido será similar a todos los que estamos utilizando para entrenar y testar el modelo.

```
testDatosESC = testDatos
testDatos = testDatos[, -8]
testDatosESC = testDatosESC[, -8]
ajusteTest = predict(modeloTR, testDatosESC, type = "raw")
caret::confusionMatrix(ajusteTest,  factor(testDatos$recod))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   >30 10-30
##      >30    1069   917
##      10-30 27291 44117
##
##                Accuracy : 0.6157
##                  95% CI : (0.6121, 0.6192)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 0.1254
##
##                   Kappa : 0.0209
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.03769
##             Specificity : 0.97964
##          Pos Pred Value : 0.53827
##          Neg Pred Value : 0.61782
##              Prevalence : 0.38641
##          Detection Rate : 0.01457
##    Detection Prevalence : 0.02706
##       Balanced Accuracy : 0.50867
##
##        'Positive' Class : >30
```

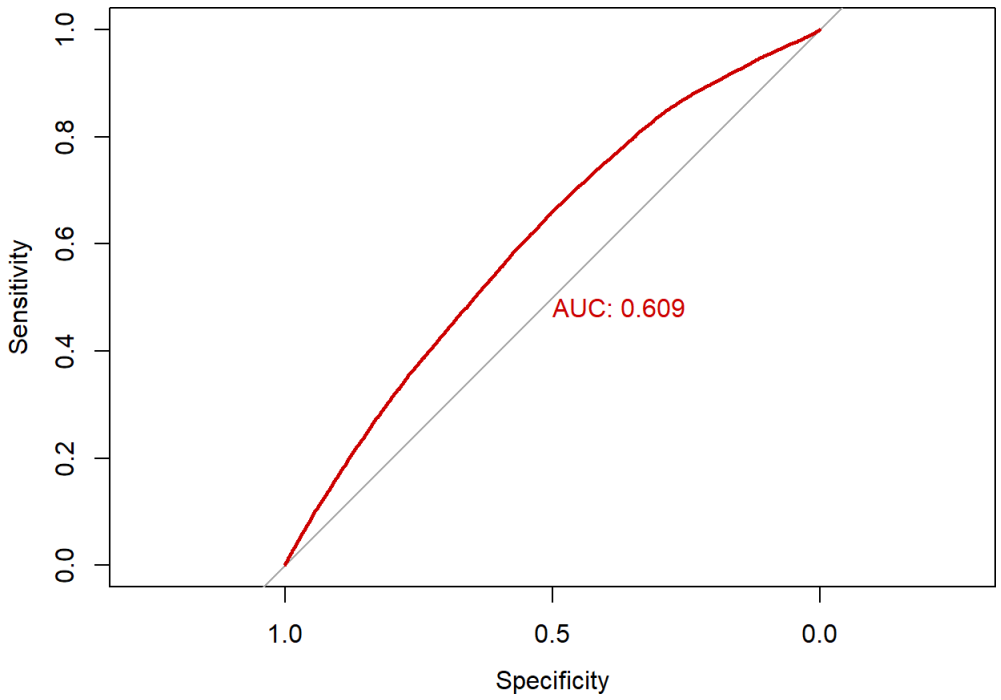
Podemos destacar que el 0’61% de los datos son clasificados correctamente. No obstante kappa es 0’02, es decir prácticamente 0. La sensubilidad es baja al igual que pasa cuando volvemos a hacer el segundo modelo con todas las variables. Esto nos indica que es muy dificil predecir datos con mucha popularidad.

A continuación, veremos las curva ROC y al area debajo de la curva que mide qué tan bien se clasifican las predicciones.

```
library(pROC)
Y = 1*(testDatos$recod == ">30")
ajusteTestProb = predict(modeloTR, testDatosESC, type = "prob")
roc(Y ~ ajusteTestProb[,">30"], plot = TRUE, print.auc = TRUE, col = "red3")
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```




```
##
## Call:
## roc.formula(formula = Y ~ ajusteTestProb[, ">30"], plot = TRUE,     print.auc = TRUE, col = "red3")
##
## Data: ajusteTestProb[, ">30"] in 45034 controls (Y 0) < 28360 cases (Y 1).
## Area under the curve: 0.6086
```

AUC = 0’60 – test Regular (0’6-0’75)

CONCLUSIÓN

La conclusión más importante que logramos extraer de este proceso es que los datos se encuentran muy dispersos y es muy dificil encontrar un modelo que nos ayude a predecir la popularidad, siendo todavía más complicado llegar a predecir valores muy altos de esta variable.

No obstante, extraemos información de nuestro conjunto de datos y de las variables que se encuentran más relacionadas entre ellas como por ejemplo hemos explicado en los gráficos generados con corrplot, donde vemos que existe una correlación positiva entre energy y loudness y negativa entre acousticness y las dos mencionadas anteriormente por separado. También, aunque el coeficiente de correlación es menor, podemos ver que a mayor danzabilidad tiene la canción mayor positividad transmite (valence).

Por último, nos queda claro que sería mejor utilizar un modelo de clasificación recodificando datos y creando clases balanceadas, que generando un modelo de regresión. No obstante, del modelo de regresión nos quedamos con que en todos los modelos se obtiene que a mayor danzabilidad tiene la canción mayor es su popularidad, por lo que esta variable nos ayuda a tener valores altos de popularidad, y tal vez, convendría componer canciones que se puedan bailar si se quiere tener éxito en el mercado.

Si nos tuvieramos que quedar con un modelo sería con el modelo de 4 variables que tiene una menor complejidad y se obtienen prácticamente unos resultados idénticos al modelo creado con todas las variables. Sabremos que en el primer Análisis Discriminante que hagamos tendremos una mayor potencia para predecir datos con popularidad mayor que 10, pero que en el segundo modelo que generamos la sensibilidad se reduce considerablemente y no nos ayuda a predecir datos con popularidad mayor que 30 de manera

correcta. Así pues proponemos estos modelos, teniendo en cuenta todo lo mencionado anteriormente. Estas conclusiones son realmente significativas y validadas debido a que hemos utilizado cross-validation creando 10 carpetas y repitiendo el proceso 30 veces, por lo que evitamos cualquier error debido a la aleatoriedad y mostramos conclusiones objetivas y validadas correctamente.