# Spotify *Hits* Popularity

Statistical Learning

Ian Solagna, Giovanna Pilot, Michele Zanatta

2022

## 1 Introduction

Thanks to the *Digital Revolution* music has become more fluid, cheap and portable, so overall more accessible. Therefore, nowadays, music is an essential part of most people's life since we can have access to it through our phones, computers, watches, cars, home assistants, literally from everywhere. One of the main reasons for the spread of accessible music in the last fifteen years is the rise of music streaming providers such as *Spotify*. By paying an affordable monthly subscription to *Spotify*, you are granted access to a vast collection of music where users can find most of the songs ever published. *Spotify* keeps track of its users' music preferences, listening habits and consequentially it can list charts of the most popular artists, songs and genres. The aim of this paper is to establish whether it is possible to predict the popularity of a song based on some of its technical features that we will further present. So our research question could be summarised as: is there a specific algorithm that one can follow to create a so-called *hit* ?

### 1.1 The Dataset

The dataset used for the analysis is publicly available on Kaggle under the name **Top Hits Spotify from 2000-2019** and it contains 2000 rows, each of them is one popular song released between 1998 and 2020. For every observation there are 18 variables which some are quantitative and a few are qualitative:

- **artist:** the name of the artist that released the song.
- **song:** the title of the song.
- **duration_ms:** the length of the song in milliseconds.
- **explicit:** binary variable indicating whether the song contains curse words
- **year:** the year the song was released.
- **popularity:** discrete variable (from 0 to 100) indicating how popular a song is.

- **danceability:** continuous variable ranging from 0 to 1, describing how suitable a track is for dancing (low values least danceable, high values most danceable).
- **energy:** continuous variable ranging from 0 to 1 that represents a perceptual measure of intensity and activity.
- **key:** the key the track is in. Integers map to pitches using standard Pitch Class notation (0 = C, 1 = C/D, 2 = D, and so on. If no key was detected, the value is -1).
- **loudness:** the loudness of a track misured in decibel.
- **mode:** the binary modality of a track.
- **speechines:** continuous variable ranging from 0 to 1 that detects the presence of spoken words in a track (0 is music with no lyrics).
- **acousticness:** a confidence measure from (0 to 1) of whether the track is acoustic.
- **instrumentalness:** continuous variable ranging from 0 to 1, that detects whether a track contains no vocals.
- **liveness:** continuous variable ranging from 0 to 1 that detects the presence of an audience in the track (1 is a live track).
- **valence:** continuous variable ranging from 0 to 1, describing the musical positiveness conveyed by a track (0 negative, 1 positive).
- **tempo:** the tempo of a track measured in beats per minute.
- **genre:** genre of the song.

As a first step of our work, we load the data in R and conduct exploratory data analysis.

```
spotify <- read.csv("spotify.csv", header=T)

sum(is.na(spotify))

## [1] 0

sum(duplicated(spotify$song))

## [1] 121

spotify <- unique(spotify) #remove duplicates

head(spotify[c("artist", "song", "year", "popularity")], 7)
```

```
##           artist                    song year popularity
## 1 Britney Spears Oops!...I Did It Again 2000         77
## 2      blink-182   All The Small Things 1999         79
## 3     Faith Hill                Breathe 1999         66
## 4       Bon Jovi           It's My Life 2000         78
## 5         *NSYNC            Bye Bye Bye 2000         65
## 6          Sisqo             Thong Song 1999         69
## 7         Eminem    The Real Slim Shady 2000         86
```

The `popularity` variable ranges from 0 to 100 and only takes discrete values. Here is how it is distributed:

```
spotify%>%ggplot(aes(x = popularity)) +
  geom_bar(stat="count")+labs( x = "\nPopularity", y = "Frequency\n") +
  theme_classic() + geom_vline(xintercept = 70, color = 'green')
```

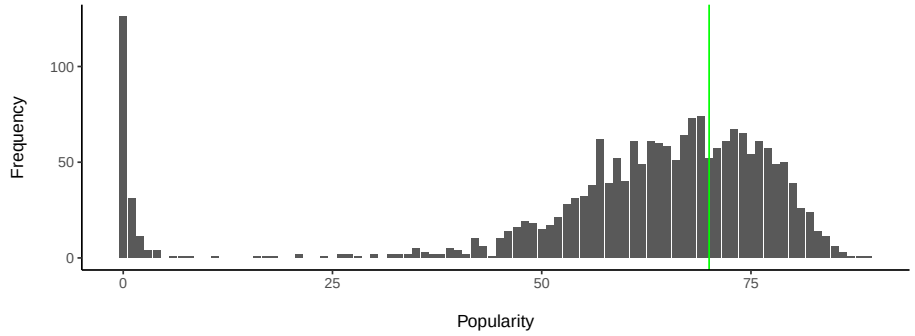



Figure 1: Distribution of Popularity

We decided to change the encoding of `popularity` as follow: if a song has a popularity greater or equal than 70, it is considered as a real *hit* and therefore labeled as **1**, otherwise it is considered just a good song and it is labeled **0**. Doing so, we transform the problem into a binary classification task which should distinguish songs that had a huge success from the others.

```
popularity <- ifelse(spotify$popularity>=70,1,0)
sum(popularity) #We have 699 hits

## [1] 699

spotify$popularity <- popularity
spotify$popularity[1:10] #Check if it worked

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

As a result, the new variable considers 699 out of the 1940 songs as true *hits*, hence the dataset is almost balanced which is a helpful property for the analysis. Before proceeding with the exploratory data analysis, we make sure that **R** reads our features in the right way.

```
str(spotify)

## 'data.frame': 1941 obs. of  18 variables:
##  $ artist          : Factor w/ 835 levels "*NSYNC","112",..: 123 105 258 115 1 696 243 64
##  $ song            : Factor w/ 1879 levels "...Ready For It?",..: 1188 73 230 825 250 165
##  $ duration_ms     : int  211160 167066 250546 224493 200560 253733 284200 258560 271333
##  $ explicit        : Factor w/ 2 levels "False","True": 1 1 1 1 1 2 2 1 1 1 ...
##  $ year            : int  2000 1999 1999 2000 2000 1999 2000 2000 1999 2001 ...
##  $ popularity      : num  1 1 0 1 0 0 1 0 1 1 ...
##  $ danceability    : num  0.751 0.434 0.529 0.551 0.614 0.706 0.949 0.708 0.713 0.72 ...
##  $ energy          : num  0.834 0.897 0.496 0.913 0.928 0.888 0.661 0.772 0.678 0.808 ...
##  $ key             : int  1 0 7 0 8 2 5 7 5 6 ...
##  $ loudness        : num  -5.44 -4.92 -9.01 -4.06 -4.81 ...
##  $ mode            : int  0 1 1 0 0 1 0 1 0 1 ...
##  $ speechiness     : num  0.0437 0.0488 0.029 0.0466 0.0516 0.0654 0.0572 0.0322 0.102 0.
##  $ acousticness    : num  0.3 0.0103 0.173 0.0263 0.0408 0.119 0.0302 0.0267 0.273 0.0079
##  $ instrumentalness: num  1.77e-05 0.00 0.00 1.35e-05 1.04e-03 9.64e-05 0.00 0.00 0.00 2.
##  $ liveness        : num  0.355 0.612 0.251 0.347 0.0845 0.07 0.0454 0.467 0.149 0.0634 .
##  $ valence         : num  0.894 0.684 0.278 0.544 0.879 0.714 0.76 0.861 0.734 0.869 ...
##  $ tempo           : num  95.1 148.7 136.9 120 172.7 ...
##  $ genre           : Factor w/ 59 levels "country","country, latin",..: 24 48 25 47 24 16

# Popularity, key and mode needs to be changed
spotify$popularity = as.factor(spotify$popularity)
spotify$key = as.factor(spotify$key)
spotify$mode = as.factor(spotify$mode)
```

## 2 Exploratory Data Analysis

Even though we will not use `artist` and `year` as regressors for the classification task, to fully understand the data, it is of interest to see who the top artists are and in which year most of the songs were released.

```
#Frequencies for artist and year
table_artist <- data.frame(table(spotify$artist))
table_year <- data.frame(table(spotify$year))

#Plot for Top 10 artists
plot1 <- table_artist %>% top_n(10) %>% ggplot(aes(x = Var1, y = Freq,)) +
```

```
            geom_bar(stat = "identity") +
            labs( x = "\nArtist", y = "Frequency\n") +
            coord_flip()+
            theme_classic()

#Plot for the years
plot2 <- table_year %>% ggplot(aes(x = Var1, y = Freq)) +
            geom_bar(stat = "identity") +
            labs( x = "\nYear", y = "Frequency\n") +
            coord_flip() +
            theme_classic()

grid.arrange(plot1, plot2, ncol=2)
```

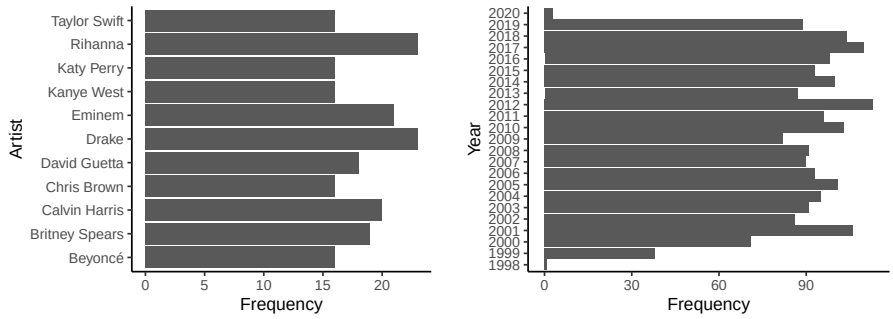



Figure 2: TOP artists and TOP years

The top artist is Rihanna and the year with the most songs is 2012.

Now, we look at the distribution of the numerical variables of the dataset:

```
#danceability
plot1 <- ggplot(spotify, aes(danceability)) +
  geom_histogram(aes(y = ..density..), binwidth=0.01,
  fill="#69b3a2", color="#e9ecef") +
  theme_classic() +geom_density()+
  theme(
    plot.title = element_text(size=15)
  )

# Same for the others
```

In figure 3, distributions are unimodal besides the `tempo` feature which seems to be bimodal. `Speechness`, `acousticness` and `liveness` are left skewed

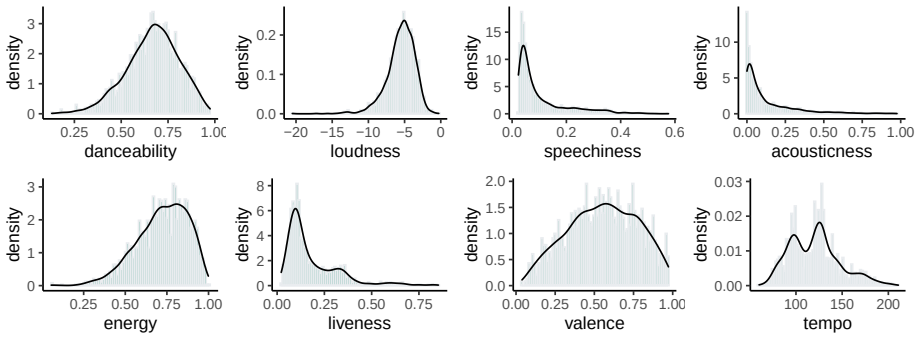



Figure 3: Density of quantitative features

meanining that most of the songs are non-acustic, not performed live and do not have much spoken words. On the contrary, `dancability`, `loudness` and `energy` are right skewed showing that the majority of the songs are very loud, intense and danceable.
Next we analyse how the covariates are related to the target variable thanks to the use of boxplot for numerical variables, and barplots for factor variables.
We first look at the engaging features of a song, such as `energy`, `loudness`, `danceability`, `tempo`:

```
#energy
plot1 <- spotify%>%
  ggplot(aes(x=popularity, y=energy, col=popularity)) +
  geom_boxplot() + theme_classic()
#Same for the other variables
```

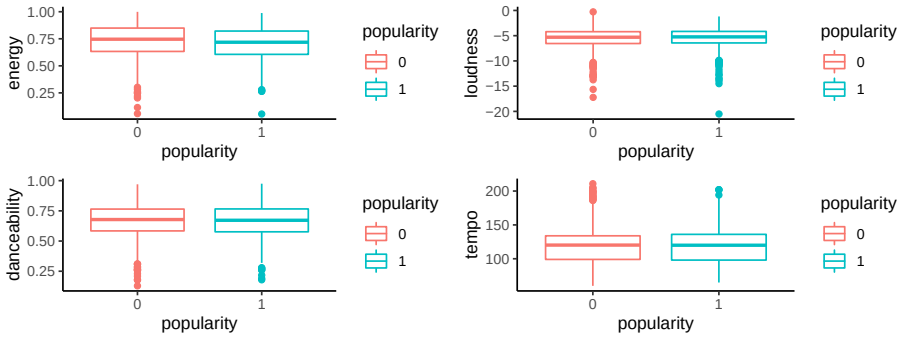



Figure 4: Engaging Features and their influence on the target variable

The boxplots in this case do not show any substantial influence of the features onto the response variable. We can see a slightly difference in the distributions

of the *hits* and *non-hits* for the variable `loudness` and `danceability`. What the plots show could mean that the most famous songs may be a little bit more danceable and louder than the less popular ones, but when it comes to `energy` and `tempo` they are basically the same.

Other interesting variables are `explicit`, `valence` and `duration_ms` which reflect features about the contents of a song: the first establishes how aggressive a track is, the second one how positive it is and the last one how long it lasts.

```
#explicit
plot1 <- ggplot(spotify, aes(explicit, fill=explicit)) +
  geom_bar(aes(y = ..count.. / sapply(PANEL,
  FUN=function(x) sum(count[PANEL == x])))) +
  facet_wrap(~popularity, nrow=2, ncol=1) +
  labs( x = "\nExplicit", y = "Relative frequency\n")

#valence
plot2 <- spotify%>%
  ggplot(aes(x=popularity, y=valence, col=popularity)) +
  geom_boxplot() +
  theme_classic()

#Same for other variables
```

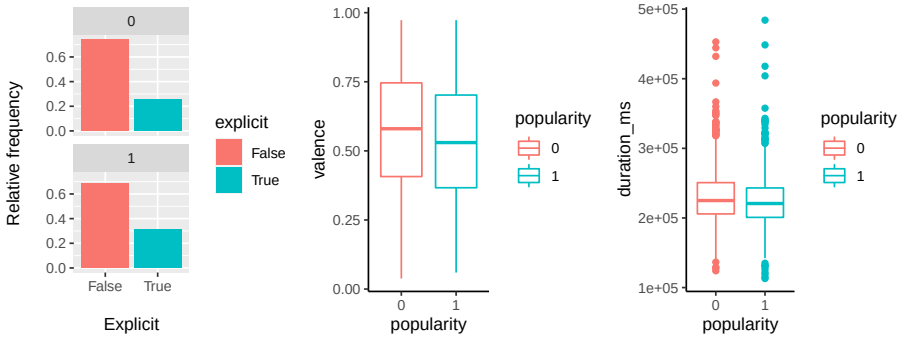



Figure 5: Contents and duration of the songs

Figure 5 shows that for popular songs the fraction of explicit and negative tracks is larger, conveying the fact that people may like more aggressive and gloomy songs. This behaviour could be explained by the fact that music is a known way to express and relieve from negative emotions. Furthermore, it seems that famous songs tend to be a little shorter.

Next we look at the variables that reflect the instrumental and the vocal features of a track: `speechines`, `acousticness`, `instrumentalness` and `liveness`.

```
#speechiness
plot1 <- spotify%>%
  ggplot(aes(x=popularity, y=speechiness, col=popularity)) +
  geom_boxplot() + theme_classic()
#Same for other variables
```

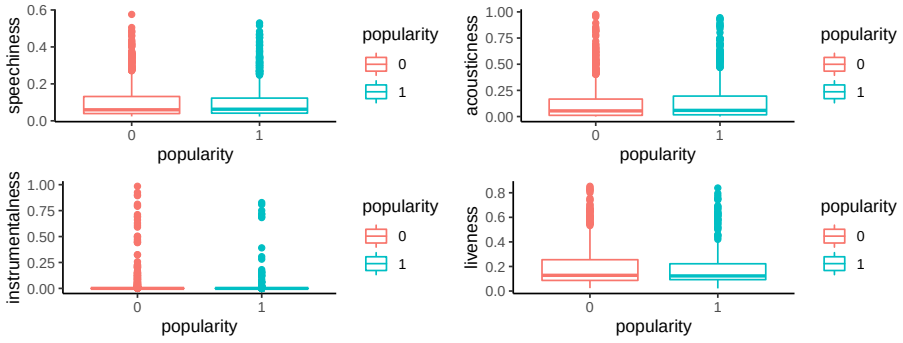



Figure 6: Features related with the instrumental

Overall, the set of variables displayed in figure 6 does not seem to have a relevant impact on the popularity. Only `acousticness` and `liveness` may have some influence over the popularity of a song but the boxplots still look very similar. Moreover, most songs have the value of `instrumentalness` set to 0, meaning that all tracks contain some vocals, therefore we decide to remove this variable since it does not convey any information.

```
spotify <- spotify[-14]
```

Finally, we take a look at the last two characteristics: `key` and `mode` which are related to the melody of the song.

```
plot1 <- spotify%>%
  ggplot(aes(x=key, y=popularity)) +
  geom_count(aes(size=..prop.., group=key),
  colour="cornflowerblue") +
  coord_fixed() +theme_minimal()

plot2 <- spotify %>%
  ggplot(aes(mode, fill=mode)) +
  geom_bar(aes(y = ..count.. / sapply(PANEL,
  FUN=function(x) sum(count[PANEL == x])))) +
  facet_wrap(~popularity, nrow=2, ncol=1) +
  labs( x = "\nMode", y = "Relative frequency\n")
```

```
grid.arrange(plot1,plot2, ncol=2)
```

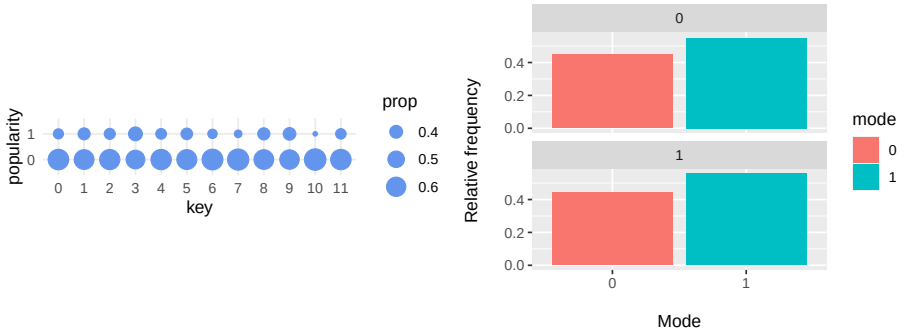



Figure 7: Features related with the instrumental

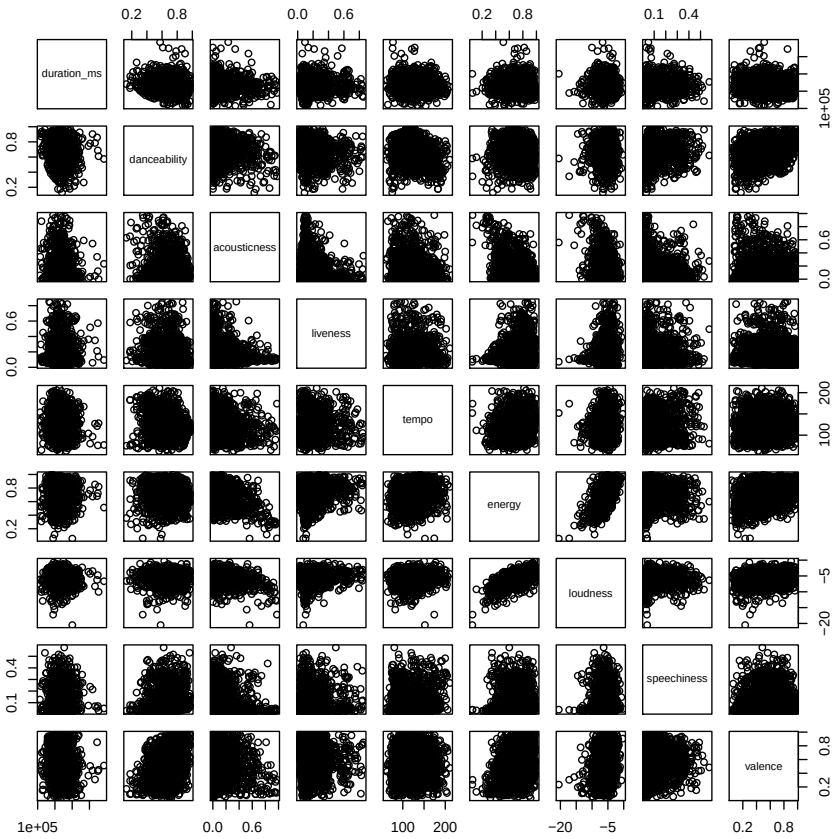
Looking at figure 7, the plot about the `mode`, the relative frequencies are almost the same for both the more popular and less popular songs. On the other hand the `key` variable shows some differences: Key 10 and Key 7 are dominated by the non-popular songs, while the others seem balanced.

In the end, we decide to remove the `genre` variable for two main reasons: first, one song may belong to more than one genre and second, we want to focus only on the characteristics of a song that are "measureable".

At this point of the analysis, we have identified the features that might influence the target variable. We can then start modelling and test if those variables are actually significant and allow us to create a model to predict whether a song has the right characteristics to become a *hit* or not.

While building a model, we could face the so-called multicollinearity due to coefficients that are strongly correlated. So before we deal with the selection of the regressors in the model, we analyse how they correlate with each other:

```
spotify %>%
    dplyr::select(duration_ms, danceability, acousticness, liveness, tempo, energy,
        loudness, speechiness, valence) %>%
    pairs()
```




From the correlation matrix, we can see that `energy` and `loudness` have a correlation of 0.65 which it is not ideal value but it should not be high enough to interfere with the estimate of the model. In case that happens, we will take actions in order to solve the issue such as removing one of the two correlated variables.

# 3 Models

As a first step of the modeling phase, we remove the variables that we want to exclude from the model, such as `artist`, `song`, `year` and `genre`.
We also create a Traning Set, a Validation Set and a Test Set.

```
variables_to_remove <- c("artist", "song", "year", "genre")

data <- spotify %>% dplyr::select(-variables_to_remove)
```

It might be convenient to standardize the quantitative variables because the model could consider variables with a higher scale as more relevant. The standardization ensures that all variables are treated evenly.

```
data%>%dplyr::select(duration_ms, danceability, acousticness,
                                liveness, tempo, energy, loudness,
                                speechiness,
                                valence) %>% cov() %>% diag()

##  duration_ms danceability acousticness     liveness        tempo       energy
## 1.540546e+09 1.977069e-02 2.978518e-02 1.985550e-02 7.284857e+02 2.336973e-02
##     loudness  speechiness      valence
## 3.759529e+00 9.244407e-03 4.877268e-02

#We need to standardize, we also create the training,
#validation and test set

train_split_pct = 0.70
val_split_pct = 0.20
test_split_pct = 0.20


#Number of rows for each set
train_size <- ceiling(nrow(data)*train_split_pct)
val_size <- ceiling(nrow(data)*val_split_pct)
test_size <- nrow(data) - train_size -val_size

#Shuffle the data
set.seed(10)
ix <-  sample(1:nrow(data))

#Creation of Train, Validation and Test
x_new = data[ix, ]
Train  = x_new[1:train_size, ]
Val  = x_new[(train_size+1):(train_size+val_size), ]
Test   = x_new[(train_size+val_size+1):(train_size+test_size+val_size), ]

#Standardize
mean_sd <- Train %>%
  summarise(across(where(is.numeric),  ~ c(mean(., na.rm = TRUE),
                                           sd(., na.rm = TRUE))))
f1 <- function(x, y) (x -y[1])/y[2]

list2env(map(lst(Train, Val, Test), ~  {
  .x[names(mean_sd)] <- map2(dplyr::select(.x, names(mean_sd)), mean_sd, f1)
  .x}), .GlobalEnv)
```

```
## <environment: R_GlobalEnv>
```

### 3.1 Linear Model

Even though in this case the target variable is a factor, we firstly model a linear regression because it provides a general overview on the significance of the coefficients and it can be used to choose the variables that we will be using in the Linear Discriminant Analysis.
We start with the full model considering all the variables.

```
Train$popularity <- as.numeric(Train$popularity)-1
lm_full <- lm(popularity~., data = Train)
summary(lm_full)

##
## Call:
## lm(formula = popularity ~ ., data = Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6604 -0.3745 -0.2747  0.5597  0.9150
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.3645114  0.0473189   7.703 2.57e-14 ***
## duration_ms  -0.0494102  0.0131994  -3.743 0.000189 ***
## explicitTrue  0.1112001  0.0337216   3.298 0.001001 **
## danceability  0.0076881  0.0158338   0.486 0.627367
## energy       -0.0410240  0.0195726  -2.096 0.036271 *
## key1          0.0097673  0.0557948   0.175 0.861060
## key2         -0.0232075  0.0627456  -0.370 0.711540
## key3          0.0437519  0.0823177   0.532 0.595161
## key4         -0.0440003  0.0667986  -0.659 0.510201
## key5          0.0294433  0.0613608   0.480 0.631421
## key6         -0.0767059  0.0643636  -1.192 0.233567
## key7         -0.0780579  0.0589103  -1.325 0.185388
## key8         -0.0387623  0.0607594  -0.638 0.523607
## key9          0.0008442  0.0636837   0.013 0.989425
## key10        -0.0883365  0.0657972  -1.343 0.179644
## key11        -0.0388949  0.0593594  -0.655 0.512423
## loudness      0.0411908  0.0171050   2.408 0.016170 *
## mode1        -0.0022767  0.0275434  -0.083 0.934136
## speechiness  -0.0162595  0.0144815  -1.123 0.261733
## acousticness  0.0139843  0.0147629   0.947 0.343679
## liveness      0.0041832  0.0132416   0.316 0.752115
```

```
## valence      -0.0568262  0.0157420  -3.610 0.000318 ***
## tempo        -0.0079778  0.0134238  -0.594 0.552410
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4747 on 1336 degrees of freedom
## Multiple R-squared:  0.04654,Adjusted R-squared:  0.03084
## F-statistic: 2.964 on 22 and 1336 DF,  p-value: 5.333e-06
```

As it can be seen from the summary, the significant variables are: `duration_ms`, `explicit`, `energy`, `loudness` and `valence`. The R-squared statistic is very low due to the fact that we are modelling a factor treating it as a continuous normal variable. Because of this, any further analysis, such as study of the residuals, is useless.
To choose the significant variables we use the `step` function that can perform a stepwise selection, the method chosen in this case is `backward`.

```
lm_backward <- step(lm_full, method="backward")
```

```
summary(lm_backward)
```

```
##
## Call:
## lm(formula = popularity ~ duration_ms + explicit + energy + loudness +
##     valence, data = Train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6082 -0.3760 -0.2815  0.5628  0.8902
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.34119    0.01517  22.499  < 2e-16 ***
## duration_ms  -0.04969    0.01310  -3.794 0.000155 ***
## explicitTrue  0.09571    0.02964   3.229 0.001271 **
## energy       -0.04995    0.01752  -2.851 0.004418 **
## loudness      0.04428    0.01689   2.622 0.008852 **
## valence      -0.05526    0.01367  -4.044 5.56e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4738 on 1353 degrees of freedom
## Multiple R-squared:  0.03799,Adjusted R-squared:  0.03443
## F-statistic: 10.69 on 5 and 1353 DF,  p-value: 4.348e-10
```

We look for a reduced model using the `stepwise` selection which confirms that `duration_ms`, `explicit`, `energy`, `loudness` and `valence` are the only five significant variables. Then, we try an `ANOVA` test to prove if the two models are the same.

We also look for significant interactions between coefficients to add to the model.

```
anova(lm_full, lm_backward)

## Analysis of Variance Table
##
## Model 1: popularity ~ duration_ms + explicit + danceability + energy +
##     key + loudness + mode + speechiness + acousticness + liveness +
##     valence + tempo
## Model 2: popularity ~ duration_ms + explicit + energy + loudness + valence
##   Res.Df    RSS  Df Sum of Sq     F Pr(>F)
## 1   1336 301.08
## 2   1353 303.78 -17   -2.7008 0.705 0.8003

add1(lm_backward,.~(.)^2, test="F")

## Single term additions
##
## Model:
## popularity ~ duration_ms + explicit + energy + loudness + valence
##                       Df Sum of Sq    RSS     AIC F value    Pr(>F)
## <none>                             303.78 -2024.0
## duration_ms:explicit  1   0.04771 303.73 -2022.3  0.2124 0.6449882
## duration_ms:energy    1   3.02111 300.76 -2035.6 13.5808 0.0002375 ***
## duration_ms:loudness  1   1.04488 302.74 -2026.7  4.6664 0.0309345 *
## duration_ms:valence   1   0.40383 303.38 -2023.9  1.7997 0.1799790
## explicit:energy       1   0.00846 303.77 -2022.1  0.0377 0.8461653
## explicit:loudness     1   0.12546 303.65 -2022.6  0.5586 0.4549564
## explicit:valence      1   0.02432 303.76 -2022.2  0.1082 0.7421996
## energy:loudness       1   0.16178 303.62 -2022.8  0.7204 0.3961695
## energy:valence        1   0.11968 303.66 -2022.6  0.5329 0.4655292
## loudness:valence      1   1.09674 302.68 -2027.0  4.8988 0.0270415 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

lm_backward2 = update(lm_backward, . ~ . +duration_ms:energy)

anova(lm_backward2, lm_backward)

## Analysis of Variance Table
##
## Model 1: popularity ~ duration_ms + explicit + energy + loudness + valence +
```

```
##     duration_ms:energy
## Model 2: popularity ~ duration_ms + explicit + energy + loudness + valence
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1   1352 300.76
## 2   1353 303.78 -1   -3.0211 13.581 0.0002375 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
add1(lm_backward2,.~(.)^2, test="F")
```

```
## Single term additions
##
## Model:
## popularity ~ duration_ms + explicit + energy + loudness + valence +
##     duration_ms:energy
##                       Df Sum of Sq    RSS     AIC F value  Pr(>F)
## <none>                             300.76 -2035.6
## duration_ms:explicit   1   0.00708 300.75 -2033.7  0.0318 0.85846
## duration_ms:loudness   1   0.00059 300.76 -2033.6  0.0027 0.95887
## duration_ms:valence    1   0.05114 300.71 -2033.9  0.2298 0.63177
## explicit:energy        1   0.00214 300.76 -2033.6  0.0096 0.92190
## explicit:loudness      1   0.09259 300.67 -2034.0  0.4160 0.51903
## explicit:valence       1   0.02833 300.73 -2033.8  0.1273 0.72134
## energy:loudness        1   0.31993 300.44 -2035.1  1.4386 0.23057
## energy:valence         1   0.30117 300.46 -2035.0  1.3542 0.24475
## loudness:valence       1   1.36580 299.39 -2039.8  6.1631 0.01316 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
lm_backward3 = update(lm_backward2, . ~ . +loudness:valence)
```

```
anova(lm_backward3, lm_backward2)
```

```
## Analysis of Variance Table
##
## Model 1: popularity ~ duration_ms + explicit + energy + loudness + valence +
##     duration_ms:energy + loudness:valence
## Model 2: popularity ~ duration_ms + explicit + energy + loudness + valence +
##     duration_ms:energy
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1   1351 299.39
## 2   1352 300.76 -1   -1.3658 6.1631 0.01316 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
add1(lm_backward3,.~(.)^2, test="F")
```

```
## Single term additions
##
## Model:
## popularity ~ duration_ms + explicit + energy + loudness + valence +
##     duration_ms:energy + loudness:valence
##                       Df Sum of Sq    RSS     AIC F value Pr(>F)
## <none>                             299.39 -2039.8
## duration_ms:explicit   1  0.015553 299.38 -2037.9  0.0701 0.7912
## duration_ms:loudness   1  0.004408 299.39 -2037.8  0.0199 0.8879
## duration_ms:valence    1  0.082049 299.31 -2038.2  0.3701 0.5431
## explicit:energy        1  0.000118 299.39 -2037.8  0.0005 0.9816
## explicit:loudness      1  0.076301 299.32 -2038.2  0.3441 0.5575
## explicit:valence       1  0.006963 299.39 -2037.8  0.0314 0.8594
## energy:loudness        1  0.001647 299.39 -2037.8  0.0074 0.9313
## energy:valence         1  0.046223 299.35 -2038.0  0.2085 0.6481
```

We add two interactions between covariates in our model: `duration_ms:energy` and `loudness:valence`. We consider this model to be the final one, so we look at the coefficients to interpret them:

```
coef(lm_backward3)

##        (Intercept)        duration_ms       explicitTrue             energy
##         0.33885227        -0.04542758         0.09247715        -0.04463562
##           loudness            valence duration_ms:energy   loudness:valence
##         0.05073823        -0.05426503         0.04819312         0.03058717
```

There is a positive correlation between `population` and `explicit`, `loudness`, this means that if they increase, then it is more likely that the song is going to be popular. Viceversa for the negative ones. It is interesting to see that both `duration_ms` and `energy` are negative correlated but their interaction is positive: it means that when these two characteristics are both in a song then the song is likely to become more popular, same thing for `valence` in combination with `loudness`. In this case as threshold popular and non popular song, we use the value 0.5 of the response variable.

```
pred_train_lm <- predict(lm_backward3, newdata = Train)
pred_test_lm <- predict(lm_backward3, newdata= Test)

table(as.factor(Train$popularity), round(pred_train_lm))

##
##       0   1
##   0 809  51
##   1 421  78

mean(Train$popularity!=round(pred_train_lm)) #Error on Train
```

```
## [1] 0.3473142

table(as.factor(Test$popularity), round(pred_test_lm))

##
##       0   1
##   0 122  13
##   1  51   7

mean(Test$popularity!=round(pred_test_lm)) #Error on Test

## [1] 0.3316062
```

We observe that the model does not perform very well. Even though for a task like this, the level of accuracy we obtained is not considerate low, the number of true positives is quite little. The problem might be in the choice of the threshold, therefore we create a function that computes the accuracy, the sensitivity and the specificity for different values of the threshold. Then using the data contained in the Validation Set we pick the value that balances the previous metrics the most.

```
check= function(predicted, truth){
  threshold = seq(0.3,0.5, by=0.025)
  check = data.frame(matrix(ncol=3, nrow=9))
  row.names(check) <- threshold
  colnames(check) <- c("Accuracy", "Sensitivity", "Specificity")
  for (i in 1:length(threshold)){
    newprev <- predicted>threshold[i]
    t<-table(newprev, truth)
    n = sum(t)
    accuracy = (t[1,1]+t[2,2])/n
    sensitivity = t[2,2]/(t[1,2]+t[2,2])
    specificity = t[1,1]/(t[1,1]+t[2,1])
    check[i,] = c(accuracy, sensitivity, specificity)
    }
    return(check)
}
```

Applying the function to the linear model we obtain:

```
check(predict(lm_backward3, newdata = Val), Val$popularity)

##        Accuracy Sensitivity Specificity
## 0.3   0.4704370   0.8169014   0.2712551
## 0.325 0.5141388   0.7323944   0.3886640
## 0.35  0.5398458   0.6267606   0.4898785
```

```
## 0.375 0.5449871   0.5070423   0.5668016
## 0.4   0.5809769   0.3802817   0.6963563
## 0.425 0.6143959   0.3309859   0.7773279
## 0.45  0.6221080   0.2605634   0.8299595
## 0.475 0.6195373   0.1830986   0.8704453
## 0.5   0.6246787   0.1126761   0.9190283
```

The best choice of the threshold seems to be 0.425. We plot the confusion matrices related to the Train set and Test set

```
table(as.factor(Train$popularity), round(pred_train_lm+0.075))

##
##       0   1
##   0 702 158
##   1 325 174

mean(Train$popularity!=round(pred_train_lm+0.075)) #Error on Train

## [1] 0.3554084

table(as.factor(Test$popularity), round(pred_test_lm+0.075))

##
##      0  1
##   0 99 36
##   1 37 21

mean(Test$popularity!=round(pred_test_lm+0.075)) #Error on Test

## [1] 0.3782383
```

From the confusion matrix, we can notice that the accuracy lowers with respect to the previous threshold but the number of true positives doubled on the training set and tripled in the test set.

## 3.2 Linear Discriminant Analysis

As mentioned earlier, we can use the variable selection of the linear model for the linear discriminant analysis. Therefore we add the same features and interactions to the LDA model.

```
Train$popularity <- as.factor(Train$popularity)
lda_spotify <- lda(Train$popularity ~ duration_ms+explicit +energy+
loudness+valence+duration_ms:energy+loudness:valence, data= Train )

check(predict(lda_spotify, newdata = Val)$posterior[,2], Val$popularity)
```

```
##        Accuracy Sensitivity Specificity
## 0.3   0.4730077   0.7957746   0.2874494
## 0.325 0.5347044   0.7183099   0.4291498
## 0.35  0.5321337   0.5915493   0.4979757
## 0.375 0.5655527   0.4647887   0.6234818
## 0.4   0.5886889   0.3591549   0.7206478
## 0.425 0.6169666   0.3309859   0.7813765
## 0.45  0.6221080   0.2605634   0.8299595
## 0.475 0.6195373   0.1830986   0.8704453
## 0.5   0.6246787   0.1267606   0.9109312
```

The best threshold in the case of the LDA seems to be 0.425, we can have a look at the model performance:

```
pred_train_lda <- predict(lda_spotify, newdata = Train)$posterior[,2]
pred_test_lda <- predict(lda_spotify, newdata =Test)$posterior[,2]

table(Train$popularity, round(pred_train_lda+0.075))

##
##       0   1
##   0 705 155
##   1 326 173

mean(Train$popularity!=round(pred_train_lda+0.075)) #Error on Train

## [1] 0.3539367

table(Test$popularity, round(pred_test_lda+0.075))

##
##      0  1
##   0 99 36
##   1 38 20

mean(Train$popularity!=round(pred_train_lda+0.075)) #Error on Test

## [1] 0.3539367
```

The performance looks almost identical to the linear model, but the accuracy on the test set is slightly higher.

## 3.3 Quadratic Discriminant Analysis

We proceed with the QDA to see if it can outperform the previous models.

```
qda_spotify <- qda(Train$popularity ~ duration_ms+explicit +energy+
loudness+valence+duration_ms:energy+loudness:valence, data= Train )

check(predict(qda_spotify, newdata = Val)$posterior[,2], Val$popularity)

##        Accuracy Sensitivity Specificity
## 0.3   0.5321337   0.5845070   0.5020243
## 0.325 0.5526992   0.5281690   0.5668016
## 0.35  0.5655527   0.4507042   0.6315789
## 0.375 0.5784062   0.4014085   0.6801619
## 0.4   0.5989717   0.3450704   0.7449393
## 0.425 0.6015424   0.3239437   0.7611336
## 0.45  0.6092545   0.3028169   0.7854251
## 0.475 0.6066838   0.2464789   0.8137652
## 0.5   0.6118252   0.1901408   0.8542510
```

In this case we pick 0.4 as threshold.

```
pred_train_qda <- predict(qda_spotify, newdata = Train)$posterior[,2]
pred_test_qda <- predict(qda_spotify, newdata =Test)$posterior[,2]

table(Train$popularity, round(pred_train_qda+0.1))

##
##       0   1
##   0 638 222
##   1 291 208

mean(Train$popularity!=round(pred_train_qda+0.10)) #Error on Train

## [1] 0.3774834

table(Test$popularity, round(pred_test_qda+0.10))

##
##      0  1
##   0 88 47
##   1 36 22

mean(Test$popularity!=round(pred_test_qda+0.10)) #Error on Test

## [1] 0.4300518
```

The QDA performs the worst compared to the two previous models, it has the highest error rate on both the training and test set.

### 3.4 Logistic Regression

Since we are dealing with a binary classification problem, the type of model that best suits this kind of task is the Logistic Regression. As we did with the linear model, we start with all the covariates and from that we reduce the number of variables contained in the model to get a smaller but as effective model.

```
glm_full <- glm(popularity ~., family = "binomial", data=Train)
summary(glm_full)

##
## Call:
## glm(formula = popularity ~ ., family = "binomial", data = Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5043  -0.9618  -0.7966   1.2824   2.0150
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.574698   0.207595  -2.768 0.005634 **
## duration_ms  -0.229590   0.061820  -3.714 0.000204 ***
## explicitTrue  0.490757   0.149369   3.286 0.001018 **
## danceability  0.030144   0.070932   0.425 0.670858
## energy       -0.189084   0.088110  -2.146 0.031873 *
## key1          0.041000   0.243739   0.168 0.866414
## key2         -0.100590   0.276829  -0.363 0.716332
## key3          0.178018   0.361629   0.492 0.622530
## key4         -0.202967   0.296962  -0.683 0.494304
## key5          0.126103   0.267886   0.471 0.637831
## key6         -0.354380   0.290524  -1.220 0.222542
## key7         -0.363505   0.265530  -1.369 0.171005
## key8         -0.176719   0.269074  -0.657 0.511332
## key9         -0.003362   0.279812  -0.012 0.990413
## key10        -0.414496   0.299256  -1.385 0.166025
## key11        -0.174175   0.262859  -0.663 0.507577
## loudness      0.194177   0.078417   2.476 0.013278 *
## mode1        -0.013094   0.123111  -0.106 0.915300
## speechiness  -0.070852   0.065240  -1.086 0.277469
## acousticness  0.060474   0.064823   0.933 0.350866
## liveness      0.017280   0.059788   0.289 0.772570
## valence      -0.253989   0.070754  -3.590 0.000331 ***
## tempo        -0.037168   0.060118  -0.618 0.536410
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1786.9  on 1358  degrees of freedom
## Residual deviance: 1722.1  on 1336  degrees of freedom
## AIC: 1768.1
##
## Number of Fisher Scoring iterations: 4
```

Like we did in the previous section, we try a stepwise backward selection but also a forward selection to see if they both reach the same set of covariates for the model.

```
glm_backward <- step(glm_full, method="backward")
glm_empty <- glm(popularity~1, family="binomial", data=Train)
glm_forward <- step(glm_empty, direction="forward", scope = formula(glm_full))
```

```
summary(glm_backward)
```

```
##
## Call:
## glm(formula = popularity ~ duration_ms + explicit + energy +
##     loudness + valence, family = "binomial", data = Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3956  -0.9640  -0.8093   1.2864   1.9555
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.68067    0.06888  -9.882  < 2e-16 ***
## duration_ms  -0.22921    0.06121  -3.745 0.000181 ***
## explicitTrue  0.41899    0.13021   3.218 0.001292 **
## energy       -0.22682    0.07891  -2.874 0.004049 **
## loudness      0.20635    0.07735   2.668 0.007638 **
## valence      -0.24590    0.06140  -4.005  6.2e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1786.9  on 1358  degrees of freedom
## Residual deviance: 1734.2  on 1353  degrees of freedom
## AIC: 1746.2
##
## Number of Fisher Scoring iterations: 4
```

```
summary(glm_forward)
```

```
##
## Call:
## glm(formula = popularity ~ valence + duration_ms + explicit +
##     energy + loudness, family = "binomial", data = Train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3956  -0.9640  -0.8093   1.2864   1.9555
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.68067    0.06888  -9.882  < 2e-16 ***
## valence      -0.24590    0.06140  -4.005  6.2e-05 ***
## duration_ms  -0.22921    0.06121  -3.745 0.000181 ***
## explicitTrue  0.41899    0.13021   3.218 0.001292 **
## energy       -0.22682    0.07891  -2.874 0.004049 **
## loudness      0.20635    0.07735   2.668 0.007638 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1786.9  on 1358  degrees of freedom
## Residual deviance: 1734.2  on 1353  degrees of freedom
## AIC: 1746.2
##
## Number of Fisher Scoring iterations: 4
```

The two methods reach the same model leading to the same set of significant variables of the linear model.
At this point, we indifferently pick one of the two, for example `glm_backward`, and use `ANOVA` to see if there is any difference with the `glm_full`.

Then we check for any interaction between the variables that we have picked.

```
anova(glm_full, glm_backward, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: popularity ~ duration_ms + explicit + danceability + energy +
##     key + loudness + mode + speechiness + acousticness + liveness +
##     valence + tempo
## Model 2: popularity ~ duration_ms + explicit + energy + loudness + valence
##   Resid. Df Resid. Dev  Df Deviance Pr(>Chi)
## 1      1336     1722.1
```

```
## 2      1353     1734.2 -17  -12.068    0.796

add1(glm_backward,.~(.)^2, test="Chisq")

## Single term additions
##
## Model:
## popularity ~ duration_ms + explicit + energy + loudness + valence
##                       Df Deviance    AIC     LRT  Pr(>Chi)
## <none>                     1734.2 1746.2
## duration_ms:explicit   1   1734.1 1748.1  0.0735 0.7863432
## duration_ms:energy     1   1721.0 1735.0 13.1563 0.0002865 ***
## duration_ms:loudness   1   1728.0 1742.0  6.1714 0.0129832 *
## duration_ms:valence    1   1732.7 1746.7  1.4820 0.2234542
## explicit:energy        1   1734.1 1748.1  0.0844 0.7714198
## explicit:loudness      1   1733.7 1747.7  0.4966 0.4810075
## explicit:valence       1   1734.2 1748.2  0.0318 0.8584566
## energy:loudness        1   1733.3 1747.3  0.8616 0.3532968
## energy:valence         1   1733.8 1747.8  0.3733 0.5412254
## loudness:valence       1   1728.8 1742.8  5.3570 0.0206390 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

glm_backward2 <- update(glm_backward, . ~ . +duration_ms:energy)

add1(glm_backward2,.~(.)^2, test="Chisq")

## Single term additions
##
## Model:
## popularity ~ duration_ms + explicit + energy + loudness + valence +
##     duration_ms:energy
##                       Df Deviance    AIC    LRT Pr(>Chi)
## <none>                     1721.0 1735.0
## duration_ms:explicit   1   1721.0 1737.0 0.0725  0.78776
## duration_ms:loudness   1   1720.9 1736.9 0.1029  0.74841
## duration_ms:valence    1   1720.9 1736.9 0.1155  0.73393
## explicit:energy        1   1721.0 1737.0 0.0063  0.93675
## explicit:loudness      1   1720.8 1736.8 0.2730  0.60132
## explicit:valence       1   1721.0 1737.0 0.0560  0.81291
## energy:loudness        1   1719.9 1735.9 1.1632  0.28080
## energy:valence         1   1719.9 1735.9 1.1635  0.28075
## loudness:valence       1   1714.4 1730.4 6.6045  0.01017 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

glm_backward3 <- update(glm_backward2, . ~ . +loudness:valence)
```

```
add1(glm_backward3,.~(.)^2, test="Chisq")

## Single term additions
##
## Model:
## popularity ~ duration_ms + explicit + energy + loudness + valence +
##     duration_ms:energy + loudness:valence
##                       Df Deviance    AIC     LRT Pr(>Chi)
## <none>                     1714.4 1730.4
## duration_ms:explicit  1   1714.3 1732.3 0.10808   0.7423
## duration_ms:loudness  1   1714.3 1732.3 0.10512   0.7458
## duration_ms:valence   1   1714.3 1732.3 0.15514   0.6937
## explicit:energy       1   1714.4 1732.4 0.00023   0.9878
## explicit:loudness     1   1714.2 1732.2 0.20408   0.6514
## explicit:valence      1   1714.4 1732.4 0.00190   0.9652
## energy:loudness       1   1714.4 1732.4 0.04312   0.8355
## energy:valence        1   1714.0 1732.0 0.46961   0.4932

anova(glm_backward,glm_backward3, test="Chisq")

## Analysis of Deviance Table
##
## Model 1: popularity ~ duration_ms + explicit + energy + loudness + valence
## Model 2: popularity ~ duration_ms + explicit + energy + loudness + valence +
##     duration_ms:energy + loudness:valence
##   Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
## 1      1353     1734.2
## 2      1351     1714.4  2   19.761 5.117e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Even for the logistic we insert the interaction between `duration_ms:energy` and `loudness:valence`. We now look at the coefficients of this last configuration of the model and try to give them an interpretation.

```
coef(glm_backward3)

##        (Intercept)        duration_ms       explicitTrue             energy
##         -0.6999080         -0.2098276          0.4093706         -0.1963621
##           loudness            valence duration_ms:energy   loudness:valence
##          0.2520440         -0.2515797          0.2246537          0.1495850
```

The values change from the Linear Regression but the way the regressors influenced the Y is still the same. For the Logistic Regression, the contribution of the coefficients to the response variable is to be interpreted in a multiplicative way. For example if a song is explicit, then it is $e^{0.409} = 1.5$ times more likely

to be popular than a non-explicit one, given that the other features keep the same values. We can check how well the model performs.

```
check(predict(glm_backward3, newdata = Val, type="response"),Val$popularity)
```

```
##         Accuracy Sensitivity Specificity
## 0.3    0.4678663   0.8028169   0.2753036
## 0.325 0.5218509   0.7112676   0.4129555
## 0.35   0.5295630   0.5915493   0.4939271
## 0.375 0.5604113   0.4929577   0.5991903
## 0.4    0.5732648   0.3521127   0.7004049
## 0.425 0.6195373   0.3309859   0.7854251
## 0.45   0.6272494   0.2676056   0.8340081
## 0.475 0.6143959   0.1830986   0.8623482
## 0.5    0.6272494   0.1267606   0.9149798
```

The threshold value we pick is the same as in the linear model: 0.425.

```
glm_train_pred <- predict(glm_backward3, newdata=Train, type="response")
glm_test_pred <- predict(glm_backward3, newdata=Test, type="response")

table(Train$popularity, round(glm_train_pred+0.075))
```

```
##
##       0   1
##   0 703 157
##   1 323 176
```

```
mean(Train$popularity!=round(glm_train_pred+0.075)) #Error on Train
```

```
## [1] 0.3532009
```

```
table(Test$popularity, round(glm_test_pred+0.075))
```

```
##
##      0  1
##   0 99 36
##   1 38 20
```

```
mean(Test$popularity!=round(glm_test_pred+0.075)) #Error on Test
```

```
## [1] 0.3834197
```

The results are very similar to the Linear Regression model, the error is slightly lower on the training set but a bit larger in the test set. In the next few sections we will try a shrinkage method to see if the model improves.

## 3.5 Ridge Regression

We try with the Ridge Regression. First of all, we need to arrange the data in a way that the `glmnet` function can read. We also define a function that for each $\lambda$, calculates the error of the model computed with said $\lambda$. This way we can choose a model that works well among all the ones that the `glmnet` function estimates.

```
lambda.grid <- 10^seq(-5,1,length=1000)
X <- model.matrix(popularity~., data=Train)
X <- X[,-1]
y <- Train$popularity

score <- function(pred, truth){
  errori <- matrix(NA, nrow = ncol(pred), ncol = 2)
  errori[,1] <- 1:ncol(pred)
  for (i in 1:ncol(pred)){
    tab = table(pred[,i]>0.5,truth)
    errori[i,2] <- 1 - sum(diag(tab))/sum(tab)
  }
  errori
}
```

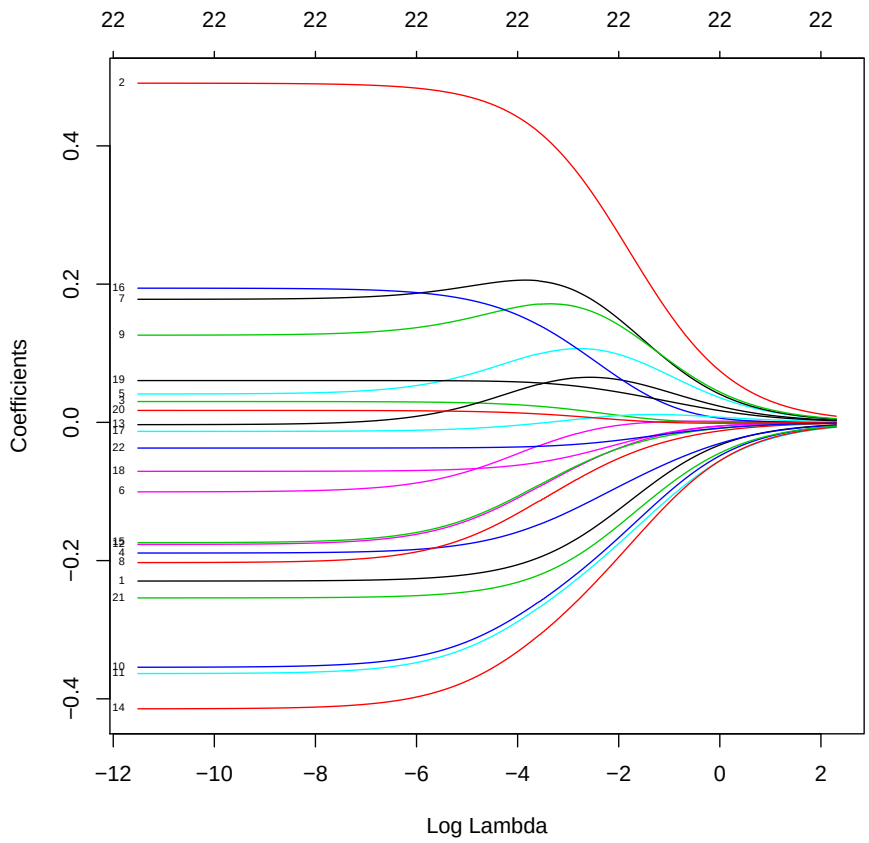
Now we can train our model and calculate the error on the Validation set.

```
spotify.ridge <- glmnet(X,y, family = "binomial", alpha = 0,
lambda.min.ratio = 0.00000001,

                        lambda=lambda.grid)  #Ridge
pred.r <- predict(spotify.ridge,
newx= model.matrix(popularity~., data=Val)[,-1],
type="response")
```

We can plot the values of the coefficients based on the lambda parameter to see the shrinkage in action.

```
plot(spotify.ridge, xvar="lambda", label=T)
```




Then we calculate the error on all models and pick the minimum.

```
errors.r <- score(pred.r, Val$popularity)
best.r <- which.min(errors.r[,2])
best_lambda_ridge <- spotify.ridge$lambda[best.r]
best_lambda_ridge

## [1] 0.2024447
```

The best $\lambda$ is 0.202447. We now look for the best threshold.

```
check(predict(spotify.ridge, newx=model.matrix(popularity~.,
data=Val)[,-1],
                              s=best_lambda_ridge, type="response"),
                              Val$popularity)

##        Accuracy Sensitivity Specificity
```

```
## 0.3   0.3958869  0.93661972  0.08502024
## 0.325 0.4575835  0.80281690  0.25910931
## 0.35  0.5089974  0.66197183  0.42105263
## 0.375 0.5372751  0.45070423  0.58704453
## 0.4   0.5629820  0.24647887  0.74493927
## 0.425 0.5861183  0.14788732  0.83805668
## 0.45  0.6143959  0.10563380  0.90688259
## 0.475 0.6375321  0.07042254  0.96356275
## 0.5   0.6452442  0.04225352  0.99190283
```

The best value is between 0.375 and 0.4.

```
pred_train_ridge <- predict(spotify.ridge,
                            newx=model.matrix(popularity~., data=Train)[,-1],
                            s=best_lambda_ridge, type="response")

table(Train$popularity, round(pred_train_ridge+0.11))

##
##       0   1
##   0 654 206
##   1 287 212

mean(Train$popularity!=round(pred_train_ridge+0.11)) #Error on Train

## [1] 0.3627667

pred_test_ridge <- predict(spotify.ridge,
                           newx=model.matrix(popularity~., data=Test)[,-1],
                           s=best_lambda_ridge, type="response")
table(Test$popularity, round(pred_test_ridge+0.11))

##
##      0  1
##   0 89 46
##   1 38 20

mean(Test$popularity!=round(pred_test_ridge+0.11)) #Error on Test

## [1] 0.4352332
```

In this case the shrinkage makes the model perform worse than the standard `glm` and is the worst alongside the QDA on the test set.
Finally we can plot the ROC curves of all the models to see which one performs the best.

```r
prediction <- tibble(truth = factor(Train$popularity)) %>%
  mutate(pred = glm_train_pred) %>%
  mutate(model = "GLM") %>%
  add_row(truth = factor(Train$popularity),
  pred = as.vector(pred_train_ridge), model = "RIDGE") %>%
  add_row(truth = factor(Train$popularity),
  pred = pred_train_lda, model = "LDA")      %>%
  add_row(truth = factor(Train$popularity),
  pred = pred_train_qda, model = "QDA")     %>%
  add_row(truth = factor(Train$popularity),
  pred = pred_train_lm, model = "LM")

 #Knitr on overleaf does not support the package yardstick with which
 #we have created and plotted the roc curves.
 #We uploaded it as an image
```

# 4 Conclusion

The aim of this project was to test whether it is possible to predict if a song will become viral by simply analysing some of its key characteristics. Thanks to the **Exploratory Data Analysis** and later the modelling, we have identified some of the features that the more popular songs have in common, such as `duration_ms`, `explicit`, `energy` and `valence`. It looks like that people prefer shorter songs that are explicit, very energic and negative even though this behaviour may change due to the interaction between variables. This is not a causal interpretation but this is what the data that was gathered tell us. We have tested several models: **Linear Regression** with stepwise selection, the **Linear and Quadratic Discriminant Analysis**, **Logistic Regression** with stepwise selection and lastly **Ridge Regression**. Unfortunately, all models perform quite poorly meaning that these characteristics are not enough to predict how popular a song is. As it can be seen from 8 all models are very similar to each other. However, the worst one seems to be the QDA while the best is one between the LDA and GLM.

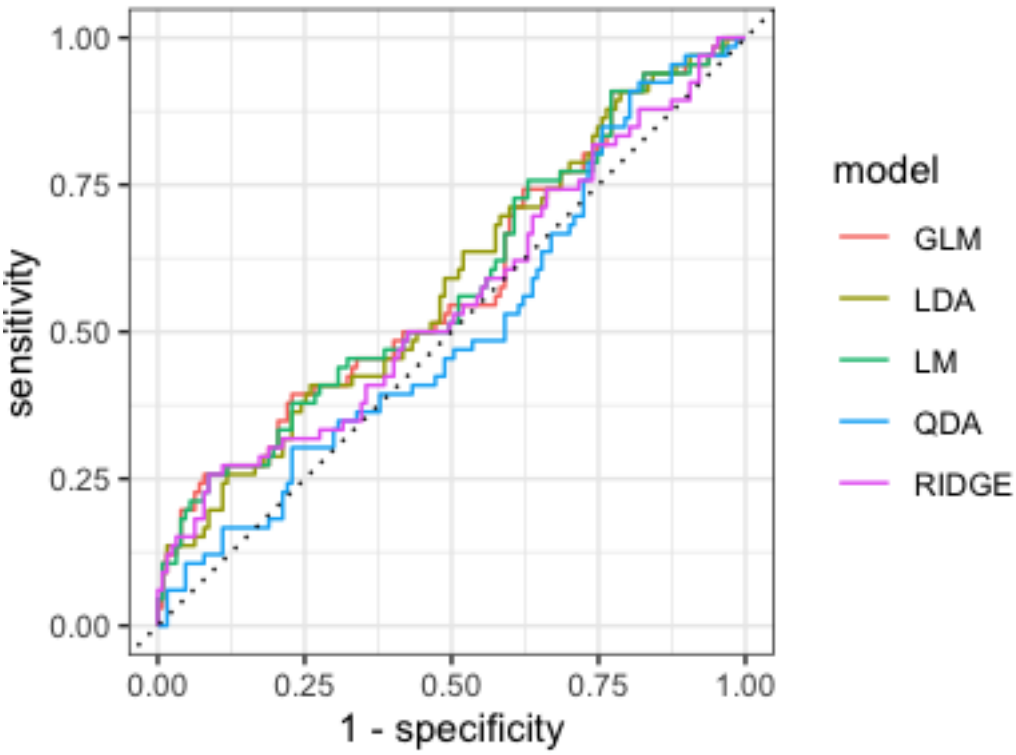



Figure 8: Roc Curves fitted on the Test Set for all Models