PH125.9x - Capstone: Hit Song Science Project

Ian Espejo-Campos

Abstract

This project explores Spotify's song popularity prediction testing five machine learning techniques. The dataset comes from Kaggle's 'Data on Songs from Billboard 1999-2019', which contains the Spotify for Developers' audio features for 154,931 songs. Ten of this variables are used for the predictive model, with a binary outcome variable for Hit/No-Hit classification, according to a cut made in the discrete popularity variable. The highest accuracy was achieved by random forest (86.8%), with high sensitivity (0.99) but low specificity (0.08) in the model.

Keywords: hit song science, Spotify, machine learning, capstone, r markdown

---

1 Introduction

In Pachet and Roy's paper "Hit Song Science Is Not Yet a Science", the authors argue that a recent formulation known as "Hit Science", aims that a cultural item's popularity can be predicted prior to its distribution:

> "More precisely, the claim is that cultural items would have specific, technical features that make them preferred by a majority of people, explaining the non uniform distribution of preferences. These features could be extracted by algorithms to entirely automate the prediction process from a given, arbitrary new item (a song or a movie scenario)" (Pachet & Roy, 2008, p. 355).

Hit Song Science (HSS) could then predict if a song could get into the Billboard Hit 100 Chart, receiving a Recording Industry Association of America (RIAA) award, or winning a Grammy, based on audio features that increases the chance of rising popularity and becoming a hit. These features are the same that companies like Spotify use for their recommendation systems: track suitability for dancing, vocals content, positiveness, among others.

Based on the data's richness, Spotify for Developers has become a powerful option for music lovers, allowing users to download datasets with the following characteristics of their playlists' songs:

- Mood
  - *Danceability*: Describes how suitable a track is for dancing based on a combination of musical elements.

  - *Energy*: Represents a perceptual measure of intensity and activity (fast, loud, and noisy).
  - *Valence*: Describes the musical positiveness conveyed by a track.
  - *Tempo*: Overall estimated tempo of a track in beats per minute (BPM).

- Properties
  - *Loudness*: Quality of a sound that is the primary psychological correlate of physical strength (amplitude).
  - *Speechiness*: Detects the presence of spoken words in a track.
  - *Instrumentalness*: Predicts whether a track contains no vocals.
  - *Duration*: Track's time length in milliseconds.

- Context
  - *Acousticness*: A confidence measure of whether the track is acoustic.
  - *Liveness*: Detects the presence of an audience in the recording (Spotify for Developers, 2020).

The Spotify Web API can be used for simulating recommendations based on an own modelling: Garg (2020) calculates the variation of random songs in comparison to his personal favorite songs in order to build a "DJ Python" music playlist with a random forest regression. From the HSS view, Middlebrook and Sheik (2019) test four models on a 1.8 million hit and no-hit songs database and get an 88% accuracy to predict Billboard success using a random forest model.

The project approach is to build a predictive model that maximizes overall accuracy by considering this ten audio features, and a calculated binary outcome that indicates `Hit = 1` if the Spotify's Popularity score (0-100) is higher or equal to `40`, based on a `154,931` song dataset from 1999 to 2019. The algorithms performed were guessing with `10%` odds, k-Nearest Neighbors (`KNN`), Generalized Linear Model (`GLM`), Linear Discriminant Analysis (`LDA`) and Random Forest (`RF`). The latest achieved the highest accuracy, but with low specificity.

2 Methodology

As the first step, we downloaded the "Data on Songs from Billboard 1999-2019" available in Kaggle (https://www.kaggle.com/danield2255/data-on-songs-from-billboard-19992019), which contains eight CSV files with information on artists and awards. For this project, we will be using the 'songAttributes_1999-2019.csv' file (stored in the Documents folder).

2.1 Data extraction

```
# Import libraries
rm(list=ls())

if(!require(tidyverse))
  install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret))
```

```
  install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(dplyr))
  install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(ggplot2))
  install.packages("ggplot2", repos = "http://cran.us.r-project.org")
if(!require(broom))
  install.packages("broom", repos = "http://cran.us.r-project.org")
if(!require(knitr))
  install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(randomForest))
  install.packages("randomForest", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(data.table)
library(ggplot2)
library(broom)
library(knitr)
library(randomForest)

# Import CSV file from Documents
songs_dataset <- read.csv("~/songAttributes_1999-2019.csv", header = T)
```

2.2 Data description

The dataset contains a list of 154,931 songs with 17 variables, including the song's name, artist and album, the audio features, and a popularity scale with values from 0 to 100.

```
dim(songs_dataset)
```

```
## [1] 154931     18
```

```
head(songs_dataset) %>% as.tibble()
```

```
## # A tibble: 6 x 18
##       X Acousticness Album Artist Danceability Duration Energy Explicit
##   <int>        <dbl> <chr> <chr>         <dbl>    <int>  <dbl> <chr>
## 1     0    0.000728  Coll~ Colle~        0.52    234947  0.904 False
## 2     1    0.0182    Coll~ Colle~        0.581   239573  0.709 False
## 3     2    0.000473  Coll~ Colle~        0.572   198400  0.918 False
## 4     3    0.00097   Coll~ Colle~        0.596   231453  0.661 False
## 5     4    0.0000358 Coll~ Colle~        0.52    222520  0.808 False
## 6     5    0.0106    Coll~ Colle~        0.353   263880  0.754 False
## # ... with 10 more variables: Instrumentalness <dbl>, Liveness <dbl>,
## #   Loudness <dbl>, Mode <int>, Name <chr>, Popularity <int>,
## #   Speechiness <dbl>, Tempo <dbl>, TimeSignature <int>, Valence <dbl>
```

```
summary(songs_dataset)
```

```
##        X          Acousticness       Album              Artist
##  Min.   :   0   Min.   :0.0000   Length:154931      Length:154931
##  1st Qu.:  39   1st Qu.:0.0202   Class :character   Class :character
##  Median :  91   Median :0.1280   Mode  :character   Mode  :character
##  Mean   : 169   Mean   :0.2663
##  3rd Qu.: 197   3rd Qu.:0.4530
##  Max.   :1796   Max.   :0.9960
##   Danceability       Duration            Energy         Explicit
##  Min.   :0.0000   Min.   :   1731   Min.   :0.0000   Length:154931
##  1st Qu.:0.4680   1st Qu.: 188907   1st Qu.:0.4830   Class :character
##  Median :0.5860   Median : 224747   Median :0.6780   Mode  :character
##  Mean   :0.5755   Mean   : 232445   Mean   :0.6387
##  3rd Qu.:0.6970   3rd Qu.: 263646   3rd Qu.:0.8290
##  Max.   :0.9870   Max.   :4795973   Max.   :1.0000
##  Instrumentalness       Liveness         Loudness            Mode
##  Min.   :0.0000000   Min.   :0.0000   Min.   :-60.000   Min.   :0.000
##  1st Qu.:0.0000000   1st Qu.:0.1020   1st Qu.: -9.852   1st Qu.:0.000
##  Median :0.0000033   Median :0.1540   Median : -6.992   Median :1.000
##  Mean   :0.0630116   Mean   :0.2543   Mean   : -8.027   Mean   :0.683
##  3rd Qu.:0.0008690   3rd Qu.:0.3260   3rd Qu.: -5.166   3rd Qu.:1.000
##  Max.   :0.9980000   Max.   :1.0000   Max.   :  3.515   Max.   :1.000
##      Name             Popularity     Speechiness         Tempo
##  Length:154931      Min.   : 0.00   Min.   :0.0000   Min.   :  0.00
##  Class :character   1st Qu.: 6.00   1st Qu.:0.0349   1st Qu.: 94.37
##  Mode  :character   Median :17.00   Median :0.0528   Median :118.86
##                     Mean   :20.25   Mean   :0.1224   Mean   :119.22
##                     3rd Qu.:31.00   3rd Qu.:0.1450   3rd Qu.:139.82
##                     Max.   :91.00   Max.   :0.9690   Max.   :248.06
##  TimeSignature      Valence
##  Min.   :0.000   Min.   :0.0000
##  1st Qu.:4.000   1st Qu.:0.3090
##  Median :4.000   Median :0.4980
##  Mean   :3.904   Mean   :0.4985
##  3rd Qu.:4.000   3rd Qu.:0.6880
##  Max.   :5.000   Max.   :0.9950
```

Notice that there are 989 artists and 9,799 albums in the dataset:

```
summarize(songs_dataset,
          unique_artist = n_distinct(Artist),
          unique_albums = n_distinct(Album))
```

```
##   unique_artist unique_albums
## 1           989          9799
```

By Slicing the data to extract the Top 20 songs in popularity, we can see singers like Taylor Swift, Ed Sheeran, and Drake.

```
songs_dataset %>%
  arrange(desc(Popularity)) %>%
  slice(1:20) %>%
  mutate(Ranking = 1:n()) %>%
  select(Ranking, Popularity, Name, Artist) %>%
  kable()
```

Ranking	Popularity	Name	Artist
1	91	Hot (feat. Gunna)	Young Thug
2	91	Lover	Taylor Swift
3	91	South of the Border (feat. Camila Cabello & Cardi B)	Ed Sheeran
4	90	No Guidance (feat. Drake)	Chris Brown
5	89	You Need To Calm Down	Taylor Swift
6	89	Beautiful People (feat. Khalid)	Ed Sheeran
7	88	Only Human	Jonas Brothers
8	88	Perfect	Ed Sheeran
9	87	Bad Bad Bad (feat. Lil Baby)	Young Thug
10	87	High Hopes	Panic! At The Disco
11	87	ME! (feat. Brendon Urie of Panic! At The Disco)	Taylor Swift
12	87	Shape of You	Ed Sheeran
13	87	I Don't Care (with Justin Bieber)	Ed Sheeran
14	86	EARFQUAKE	Tyler The Creator
15	86	Going Bad (feat. Drake)	Meek Mill
16	86	The Man	Taylor Swift
17	86	God's Plan	Drake
18	85	I Forgot That You Existed	Taylor Swift
19	85	Cruel Summer	Taylor Swift
20	85	Photograph	Ed Sheeran

2.3 Data selection

For the model we kept just the continuous features (Danceability, Energy, Loudness, Speechiness, Acousticness, Instrumentalness, Liveness, Valence, Tempo, and Duration), and the Spotify's Popularity ranking.

```
# Variables selection
songs_dataset <- songs_dataset %>%
  select(Popularity, Danceability, Energy, Loudness, Speechiness, Acousticness,
         Instrumentalness, Liveness, Valence, Tempo, Duration)
```

Analyzing the correlation between Popularity and the audio features, Loudness has the highest percent, and Danceability the lowest (in absolute values).

```r
# Correlation between Popularity and features
cor(songs_dataset[ ,1],
    songs_dataset[ ,sapply(songs_dataset, is.numeric)])
```

```
##      Popularity Danceability     Energy  Loudness Speechiness Acousticness
## [1,]          1 -0.007350333 0.09923257 0.1469342  -0.1261396     -0.10745
##      Instrumentalness    Liveness     Valence     Tempo   Duration
## [1,]      -0.06061138 -0.05928446 -0.02940333 0.0449777 0.04561194
```

Average Popularity is 20.25, while the standard deviation is 16.51. This way, we can categorize Hit as a song with a popularity higher than the mean plus one standard deviation, which is approximately 40.

```r
# 'Hit' flag defined as songs with Popularity greater or equal to 40
mean(songs_dataset$Popularity)
```

```
## [1] 20.24911
```

```r
sd(songs_dataset$Popularity)
```

```
## [1] 16.50665
```

```r
mean(songs_dataset$Popularity) + sd(songs_dataset$Popularity)
```

```
## [1] 36.75576
```

```r
songs_dataset <- songs_dataset %>%
  mutate(Hit = as.factor(ifelse(Popularity < 40, 0, 1))) %>%
  select(-Popularity) %>%
  select(Hit, everything())
```

2.4 Data partition

Now that we have a binary outcome for prediction, and the chosen features, we can split the data in a train set with 80% of the observations, and a test set with the remaining 20% for validation purposes.

```r
# Index with 0.2 of probability
set.seed(1, sample.kind = "Rounding")
test_index <- createDataPartition(songs_dataset$Hit,
                                  times = 1,
                                  p = 0.2,
                                  list = F)
```

The Test Set has 30,987 rows, and the Train Set 123,944, where 13.7% are Hits.

```
# Train and Test sets creation
test_set <- songs_dataset[test_index,]
nrow(test_set)
```

```
## [1] 30987
```

```
train_set <- songs_dataset[-test_index,]
nrow(train_set)
```

```
## [1] 123944
```

```
mean(train_set$Hit == 1)
```

```
## [1] 0.1371264
```

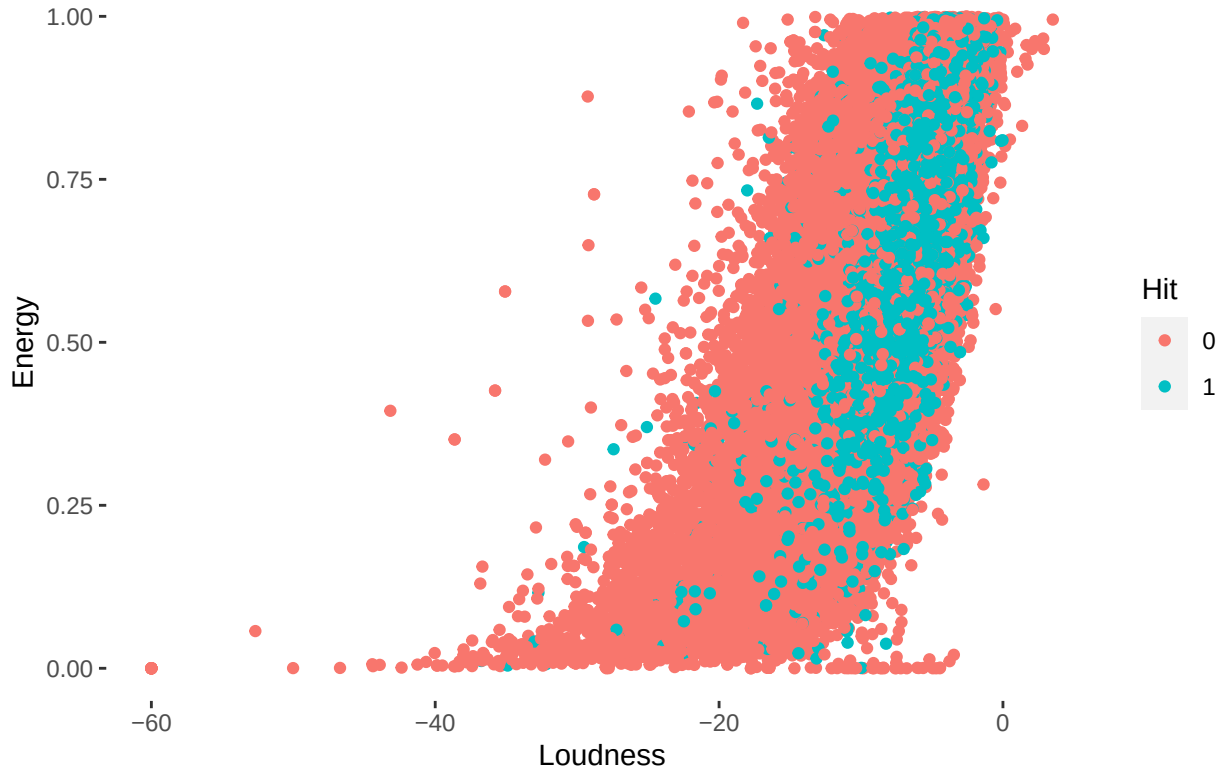
2.5 Data visualization

In order to begin with the data visualization, there's a need to explore the relationship between the audio features and their behavior with Hit Songs:

- Hits are mostly all over the spectrum of energy (except near 0 values), but are more common with high levels of loudness (near 0).

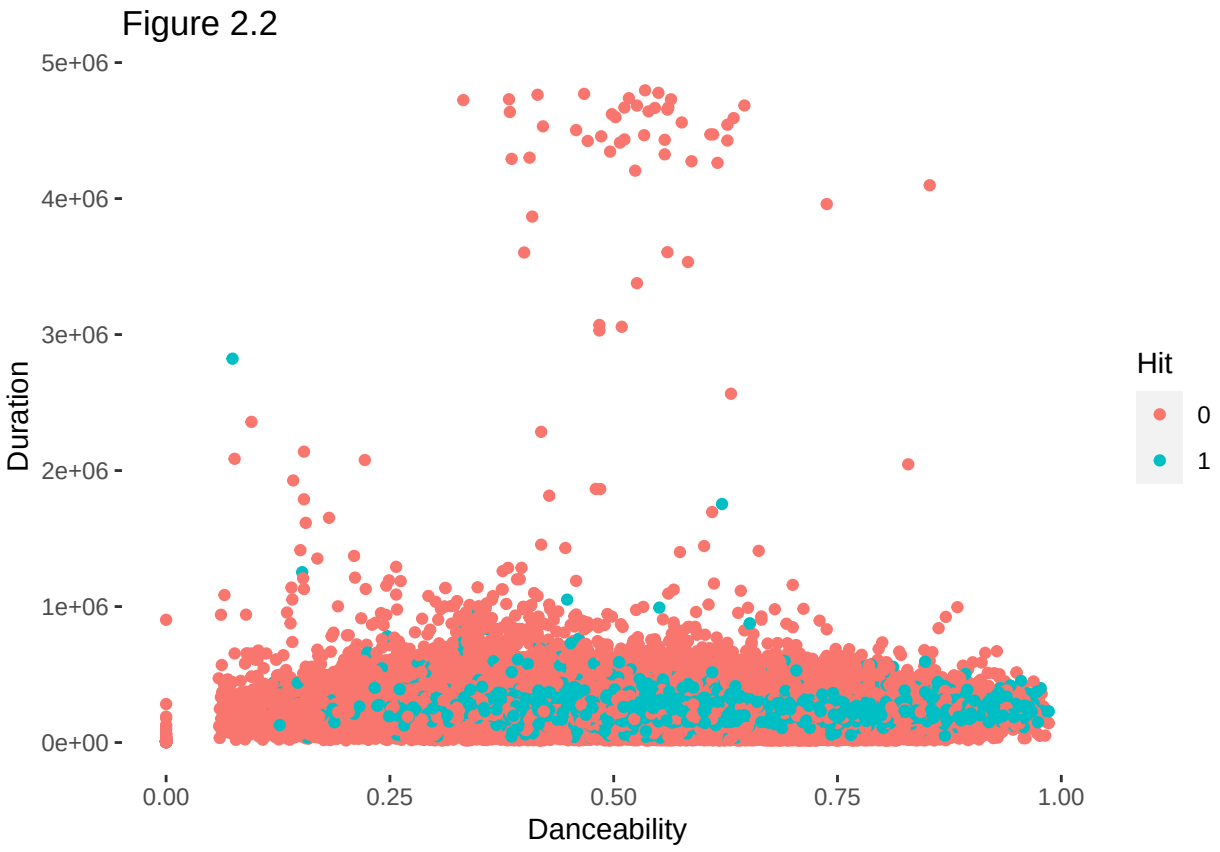
```
train_set %>% ggplot(aes(Loudness, Energy, color = Hit)) +
  geom_point() +
  theme_update() +
  labs(x = "Loudness", y = "Energy") +
  ggtitle("Figure 2.1") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white"))
```

Figure 2.1



- Hits tend to have higher values of Danceability, and standard Duration time.

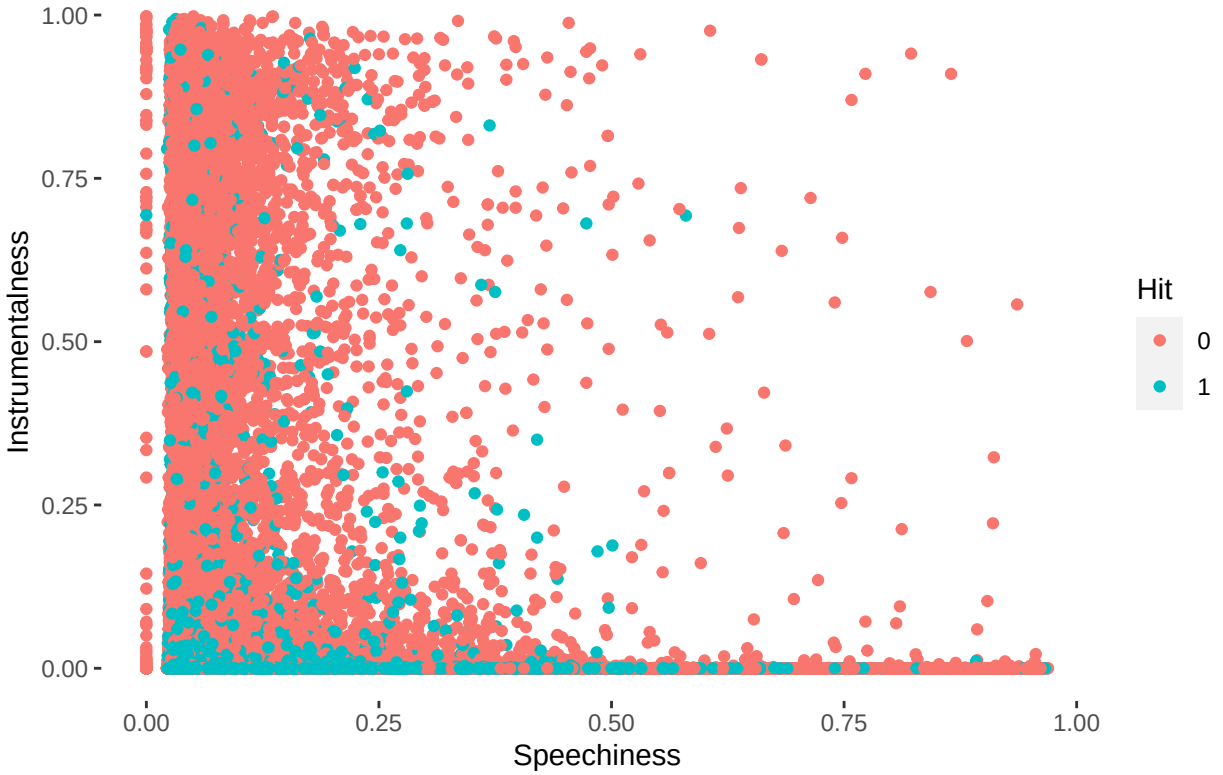
```
train_set %>% ggplot(aes(Danceability, Duration, color = Hit)) +
  geom_point() +
  theme_update() +
  labs(x = "Danceability", y = "Duration") +
  ggtitle("Figure 2.2") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white"))
```



- Hits usually keep low levels of Speechness, and lower compared to Instrumentalness.

```
train_set %>% ggplot(aes(Speechiness, Instrumentalness, color = Hit)) +
  geom_point() +
  theme_update() +
  labs(x = "Speechiness", y = "Instrumentalness") +
  ggtitle("Figure 2.3") +
  theme(panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_rect(fill = "white"))
```

Figure 2.3



2.6 Data modelling

The first step of this section is to run a Logistic Regressions for the Hit outcome against every feature individually. Based on this information, all the audio features have significant p-values (lower than `0.001`).

```
# Logistic Regressions
tidy(glm(Hit ~ Danceability, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic      p.value
##   <chr>           <dbl>     <dbl>     <dbl>        <dbl>
## 1 (Intercept)    -2.00     0.0299    -66.7  0
## 2 Danceability    0.271    0.0495      5.47 0.0000000450
```

```
tidy(glm(Hit ~ Energy, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)   -2.38     0.0263     -90.5 0.
## 2 Energy         0.830    0.0373      22.3 9.89e-110
```

```
tidy(glm(Hit ~ Loudness, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)  -1.15     0.0195      -58.8 0.
## 2 Loudness      0.0920   0.00251      36.7 3.19e-295
```

```
tidy(glm(Hit ~ Speechiness, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    -1.67    0.0108    -155.  0.
## 2 Speechiness    -1.51    0.0685     -22.1 4.15e-108
```

```
tidy(glm(Hit ~ Acousticness, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic   p.value
##   <chr>           <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    -1.65     0.0106    -155.  0.
## 2 Acousticness   -0.785    0.0305     -25.8 1.60e-146
```

```
tidy(glm(Hit ~ Instrumentalness, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term             estimate std.error statistic  p.value
##   <chr>               <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)        -1.80    0.00856    -210.  0.
## 2 Instrumentalness   -0.719   0.0500      -14.4 9.08e-47
```

```
tidy(glm(Hit ~ Liveness, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)    -1.59    0.0124    -128.  0.
## 2 Liveness       -1.07    0.0425     -25.2 5.14e-140
```

```
tidy(glm(Hit ~ Valence, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   -1.73     0.0187    -92.2  0.
## 2 Valence       -0.224    0.0344     -6.50 7.91e-11
```

```
tidy(glm(Hit ~ Tempo, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) -2.16     0.0332       -65.2 0.
## 2 Tempo        0.00267  0.000265      10.1 7.68e-24
```

```
tidy(glm(Hit ~ Duration, data = train_set, family = binomial))
```

```
## # A tibble: 2 x 5
##   term            estimate    std.error statistic  p.value
##   <chr>              <dbl>        <dbl>     <dbl>    <dbl>
## 1 (Intercept) -1.93        0.0155         -125.   0.
## 2 Duration     0.000000401 0.0000000552      7.25 4.02e-13
```

However, when running the Logistic Regression, Energy shows as not significant.

```
# Logistic Model
log_model <- glm(Hit ~ ., data = train_set, family = binomial)

summary(log_model)
```

```
##
## Call:
## glm(formula = Hit ~ ., family = binomial, data = train_set)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0251  -0.6046  -0.5098  -0.3882   3.3360
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -7.706e-01  9.219e-02  -8.359  < 2e-16 ***
## Danceability      2.543e-01  6.332e-02   4.017 5.90e-05 ***
## Energy           -1.596e-03  7.576e-02  -0.021  0.98319
## Loudness          8.002e-02  4.067e-03  19.675  < 2e-16 ***
## Speechiness      -1.440e+00  7.862e-02 -18.319  < 2e-16 ***
## Acousticness     -3.078e-01  4.398e-02  -6.999 2.58e-12 ***
```

```
## Instrumentalness -7.306e-01  5.425e-02 -13.467  < 2e-16 ***
## Liveness         -9.954e-01  4.509e-02 -22.076  < 2e-16 ***
## Valence          -6.157e-01  4.402e-02 -13.987  < 2e-16 ***
## Tempo             1.202e-03  2.855e-04   4.208 2.57e-05 ***
## Duration          2.034e-07  6.574e-08   3.094  0.00198 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 99084  on 123943  degrees of freedom
## Residual deviance: 95896  on 123933  degrees of freedom
## AIC: 95918
##
## Number of Fisher Scoring iterations: 5
```

```
tidy(log_model, conf.int = T)
```

```
## # A tibble: 11 x 7
##    term           estimate   std.error statistic   p.value   conf.low  conf.high
##    <chr>             <dbl>       <dbl>     <dbl>     <dbl>      <dbl>      <dbl>
##  1 (Intercept)    -7.71e-1     9.22e-2   -8.36   6.35e- 17   -9.51e-1   -5.90e-1
##  2 Danceabili~     2.54e-1     6.33e-2    4.02   5.90e-  5    1.30e-1    3.79e-1
##  3 Energy         -1.60e-3     7.58e-2   -0.0211 9.83e-  1   -1.50e-1    1.47e-1
##  4 Loudness        8.00e-2     4.07e-3   19.7    3.52e- 86    7.21e-2    8.80e-2
##  5 Speechiness    -1.44e+0     7.86e-2  -18.3    5.83e- 75   -1.60e+0   -1.29e+0
##  6 Acousticne~    -3.08e-1     4.40e-2   -7.00   2.58e- 12   -3.94e-1   -2.22e-1
##  7 Instrument~    -7.31e-1     5.43e-2  -13.5    2.45e- 41   -8.38e-1   -6.25e-1
##  8 Liveness       -9.95e-1     4.51e-2  -22.1    5.43e-108   -1.08e+0   -9.07e-1
##  9 Valence        -6.16e-1     4.40e-2  -14.0    1.87e- 44   -7.02e-1   -5.29e-1
## 10 Tempo           1.20e-3     2.86e-4    4.21   2.57e-  5    6.42e-4    1.76e-3
## 11 Duration        2.03e-7     6.57e-8    3.09   1.98e-  3    7.07e-8    3.29e-7
```

We know that the Train Set has a prevalence of 13.7% of Hits. Based on this, we guess with a 10% odd that a random song will be a Hit. The Guess Model got an accuracy of 0.7899.

```
# Guessing Model
p <- 0.9
set.seed(2, sample.kind = "Rounding")
guess <- sample(c(0,1),
                length(test_index),
                replace = T,
                prob=c(p, 1-p)) %>%
  factor(levels = levels(test_set$Hit))

mean(guess == test_set$Hit)
```

```
## [1] 0.7899442
```

For the k-Nearest Neighbors model, we performed a tryout with the trainControl function, using 10-fold cross validation (10 samples with 10% of the observations each) with 25 cross validations from 3 to 51 neighbors. As we observe in the plot, the accuracy tend to stabilize as k increases (code available in R script):

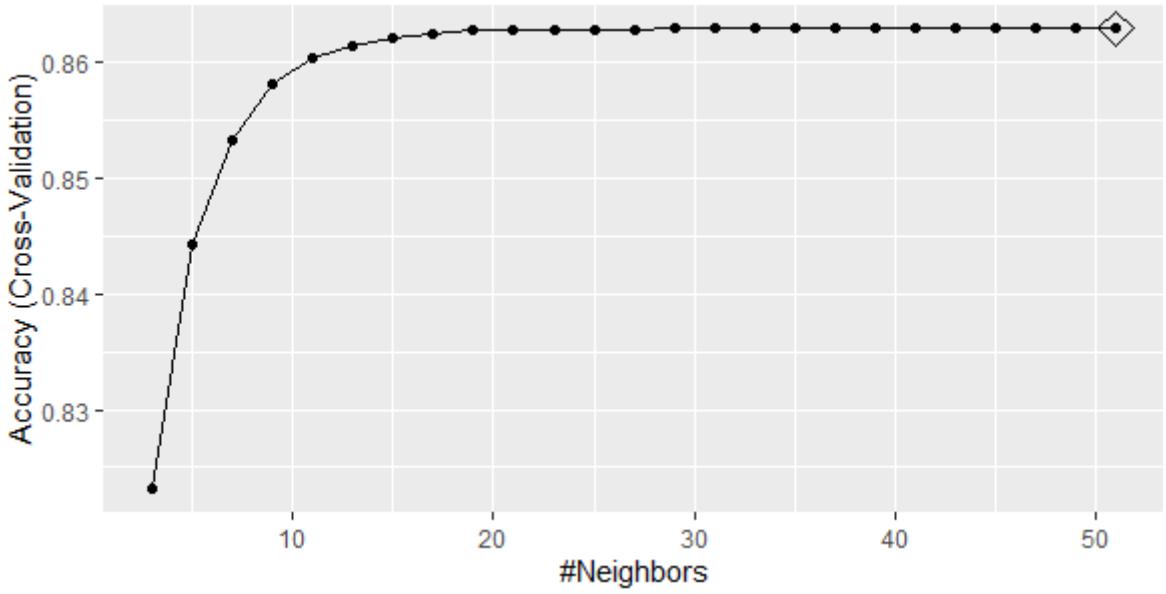
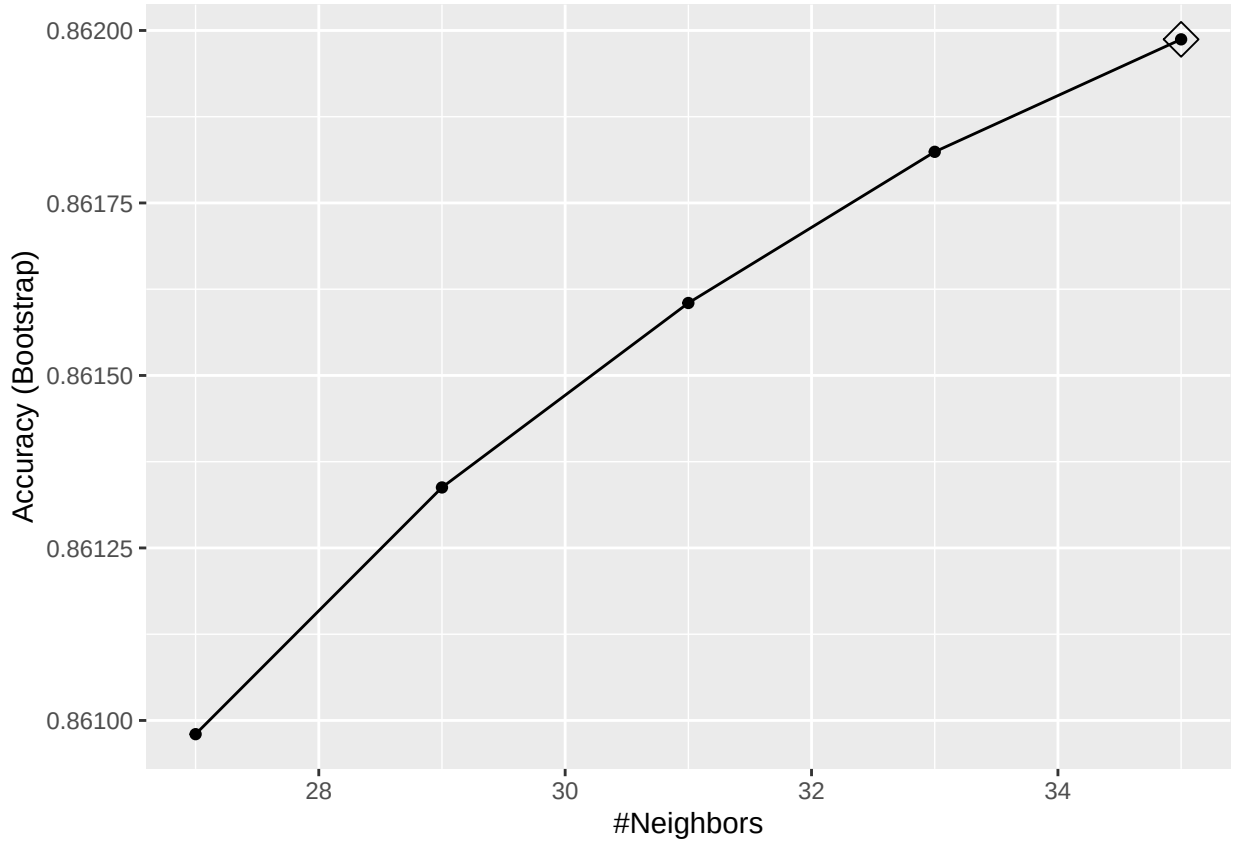


Figure 1: train_knn_cv

We trained a `KNN` complete model comparing the number of neighbors from 27 to 35. We got a `0.8628` of accuracy with `K = 35`.

```
# k-Nearest Neighbor Model
set.seed(3, sample.kind = "Rounding")
train_knn <- train(Hit ~ .,
                   method = "knn",
                   data = train_set,
                   tuneGrid = data.frame(k = seq(27, 35, 2)))

ggplot(train_knn, highlight = T)
```




```
train_knn$finalModel
```

```
## 35-nearest neighbor model
## Training set outcome distribution:
##
##      0      1
## 106948  16996
```

```
confusionMatrix(predict(train_knn, test_set, type = "raw"),
                test_set$Hit)$overall["Accuracy"]
```

```
##  Accuracy
## 0.8628134
```

The third and fourth models are GLM and LDA. The accuracy was `0.8629` approximately for both.

```
# Generalized Linear Model
set.seed(2, sample.kind = "Rounding")
train_glm <- train(Hit ~ .,
                       method = "glm",
                       data = train_set)
```

```
## Generalized Linear Model
##
## 123944 samples
##     10 predictor
##      2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 123944, 123944, 123944, 123944, 123944, 123944, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8629126  -5.27819e-06
```

```
# Linear Discriminant Analysis Model
set.seed(2, sample.kind = "Rounding")
train_lda <- train(Hit ~ .,
                   method = "lda",
                   data = train_set)
```

```
## Linear Discriminant Analysis
##
## 123944 samples
##     10 predictor
##      2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 123944, 123944, 123944, 123944, 123944, 123944, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.8629152  0
```

Finally, we trained a Random Forest Model, limiting the numbers of trees to grow to 100, and the sample variables from 1 to 10 (code available in R script).

As we can see, the `mtry` with best performance was `1`, with an accuracy of `0.8676`.

3 Results

As final model, we executed a Random Forest with the default 500 trees, exploring the grid of 1 to 5 variables randomly sampled.

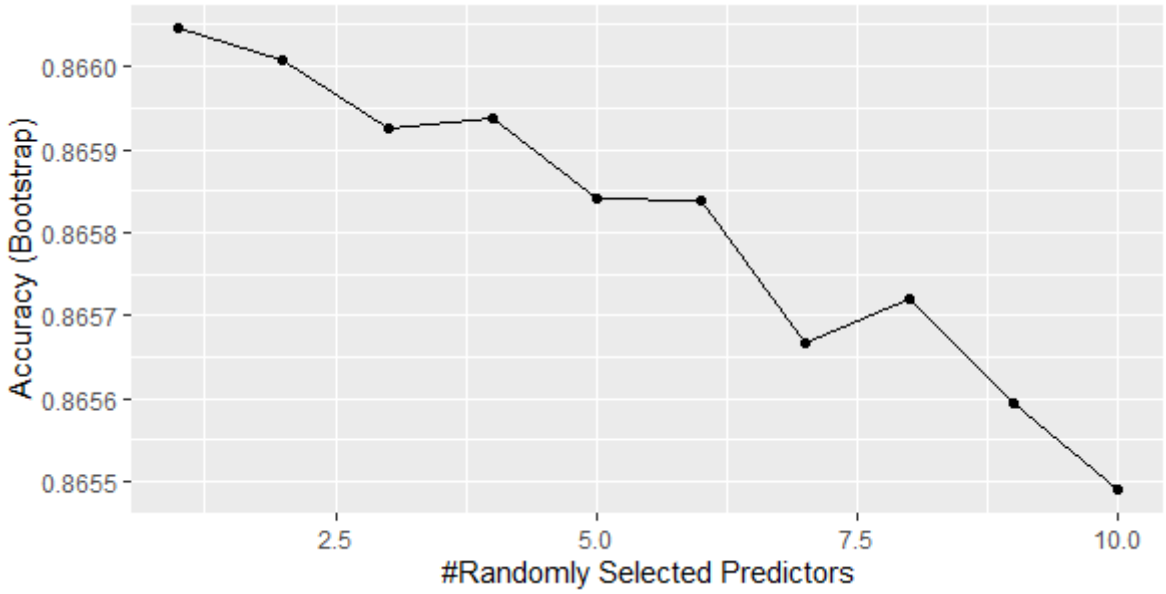


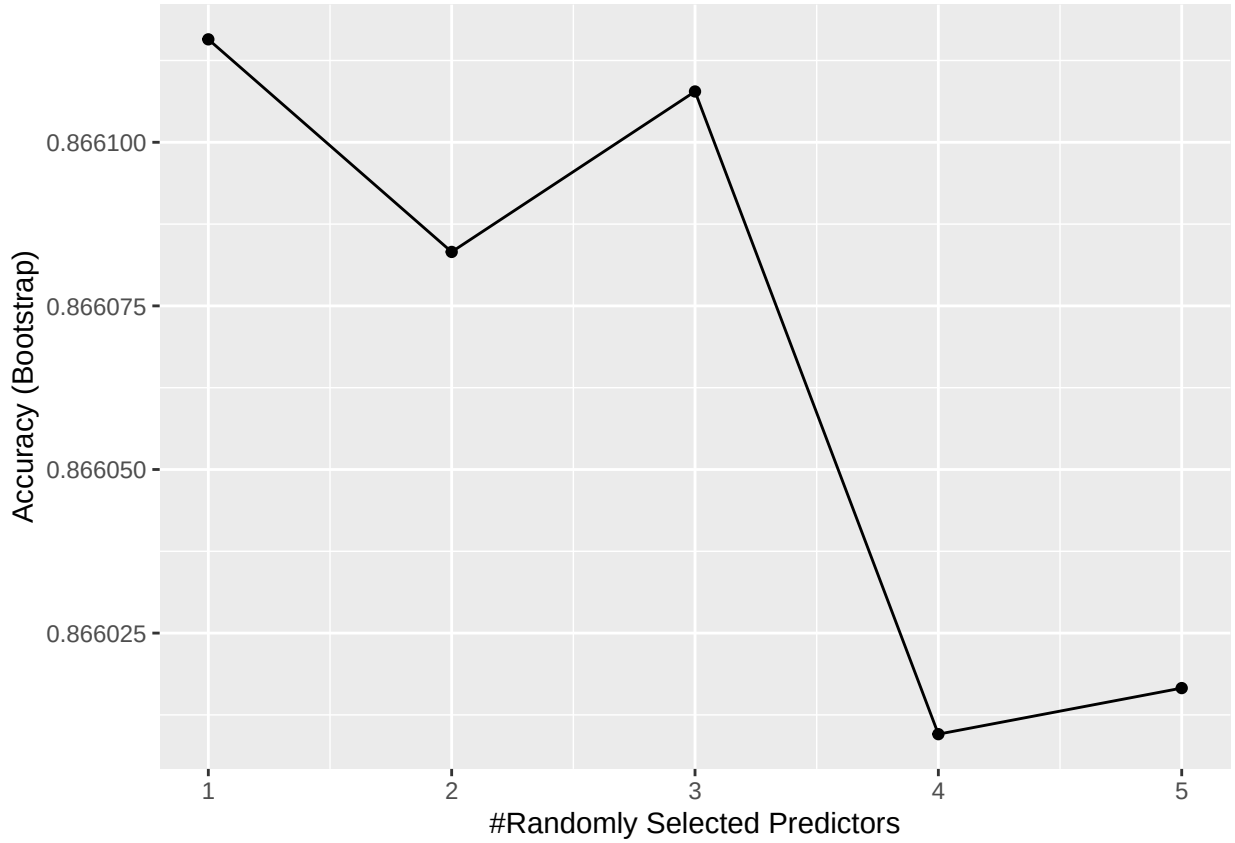
Figure 2: train_rf_1oot

```
# Random Forest Model
set.seed(5, sample.kind = "Rounding")
```

```
## Warning in set.seed(5, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
train_rf <-  train(Hit ~ .,
                   data = train_set,
                   method = "rf",
                   tuneGrid = data.frame(mtry = 1:5))

ggplot(train_rf)
```



The model is a classification forest of 500 tress, keeping `mtry = 1` as the best performer.

```
train_rf$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 13.21%
## Confusion matrix:
##        0    1 class.error
## 0 106264  684 0.006395632
## 1  15691 1305 0.923217228
```

Duration appears as the most important variable, while Instrumentalness has 0 relevance for the model.

```
varImp(train_rf)
```

```
## rf variable importance
##
##                   Overall
## Duration           100.00
## Loudness            99.31
## Acousticness        88.61
## Tempo               88.12
## Liveness            85.15
## Speechiness         83.66
## Valence             79.62
## Energy              79.59
## Danceability        78.97
## Instrumentalness     0.00
```

The final accuracy is 0.8678801.

```
rf_preds <- predict(train_rf, test_set)

cm <- confusionMatrix(rf_preds, test_set$Hit)
cm$overall["Accuracy"]
```

```
##  Accuracy
## 0.8678801
```

However, reviewing the Confusion Matrix, we can observe that the specificity of the model (the proportion of negatives that are called negatives) is 0.08. The matrix shows how 3,909 songs with high Popularity were predicted as no-hits. Sensibility (the proportion of positives that are called positives) is 0.99.

```
cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 26553  3909
##          1   185   340
##
##                Accuracy : 0.8679
##                  95% CI : (0.8641, 0.8716)
##     No Information Rate : 0.8629
##     P-Value [Acc > NIR] : 0.005192
##
##                   Kappa : 0.1158
##
##  Mcnemar's Test P-Value : < 2.2e-16
```

```
##
##             Sensitivity : 0.99308
##             Specificity : 0.08002
##          Pos Pred Value : 0.87168
##          Neg Pred Value : 0.64762
##              Prevalence : 0.86288
##          Detection Rate : 0.85691
##    Detection Prevalence : 0.98306
##       Balanced Accuracy : 0.53655
##
##        'Positive' Class : 0
##
```

4 Conclusion

Across this project, we first described Spotify's features while exploring a `155K` songs dataset from Kaggle. We perform five different machine learning techniques in order to maximize the accuracy of predicting a song's popularity based on audio characteristics. We achieved an accuracy of `0.8679` in the Random Rofest model, but the analysis also shows that the model has a high value for sensitivity (`0.99`), but a low performance on specificity (`0.08`).

We can notice how our result is consisting with Middlebrook & Sheik (2019) in terms of the accuracy level and the final technique; however, it is also important to remember that the Music Industry is lead not just by the songs attributes, but also by the artist's popularity, number of followers, the record label under which the artist is signed and how much marketing investment they put into the project, as well as awards received in the past, among other factors unavailable for the audience. That's the reason why it is not surprising finding out that the audio features have power to predict odds for a song to become popular, but there is also a need to consider other music industry factors.

For future works, there are many improvements we can do over the model:

1. We constructed the `Hit` variable according to a cut based on the mean and standard deviation of the popularity index. It is also interesting to perform the model with other outcomes like the odds of winning an RIAA certification or a Grammy award.
2. Data enrichment is important too, so it could be a good exercise to combine audio features with other information like song's release year (to see changes in listener's tastes), artist's number of followers (to see social media impact), or artist's former popular songs (to see if odds increase for a song if the performer already had x number of hits before).
3. Due to software limitations, future work should consider datasets with less number of songs, or ML techniques that can be able to deal with high dimension matrices.

5 References

- Garg, M. (2020). "Build Your Spotify Playlist Using Machine Learning". Medium. Available in: https://medium.com/swlh/build-spotify-playlist-using-machine-learning-

45352975d2ee

- Herremans, D. (2019). "Data science for hit song prediction". Medium. Available in: https://towardsdatascience.com/data-science-for-hit-song-prediction-32370f0759c1
- Irizarry, R. A. (2020). "Introduction to Data Science: Data Analysis and Prediction Algorithms with R". Available in: https://rafalab.github.io/dsbook/
- Middlebrook, K. & Sheik, K. (2019). "Song Hit Prediction: Predicting Billboard Hits Using Spotify Data". arXiv:1908.08609v2 [cs.IR] 18 Sep 2019
- Pachet, F. & Roy, P. (2008). "Hit Song Science is Not yet a Science". Proc. of Ismir '09, Philadelphia, pp. 355-360
- Spotify for Developers. (2020). "Get Audio Features for a Track". Available in: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/

6 Appendix

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19041)
##
## Matrix products: default
##
## Random number generation:
##  RNG:     Mersenne-Twister
##  Normal:  Inversion
##  Sample:  Rounding
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] data.table_1.13.2   randomForest_4.6-14 knitr_1.30
##  [4] broom_0.7.2         caret_6.0-86        lattice_0.20-41
##  [7] forcats_0.5.0       stringr_1.4.0       dplyr_1.0.2
## [10] purrr_0.3.4         readr_1.4.0         tidyr_1.1.2
```

```
## [13] tibble_3.0.4        ggplot2_3.3.2       tidyverse_1.3.0
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.2           jsonlite_1.7.1       splines_4.0.3
##  [4] foreach_1.5.1        prodlim_2019.11.13   modelr_0.1.8
##  [7] assertthat_0.2.1     highr_0.8            stats4_4.0.3
## [10] cellranger_1.1.0     yaml_2.2.1           ipred_0.9-9
## [13] pillar_1.4.7         backports_1.2.0      glue_1.4.2
## [16] pROC_1.16.2          digest_0.6.27        rvest_0.3.6
## [19] colorspace_2.0-0     recipes_0.1.15       htmltools_0.5.0
## [22] Matrix_1.2-18        plyr_1.8.6           timeDate_3043.102
## [25] pkgconfig_2.0.3      haven_2.3.1          scales_1.1.1
## [28] gower_0.2.2          lava_1.6.8.1         farver_2.0.3
## [31] generics_0.1.0       ellipsis_0.3.1       withr_2.3.0
## [34] nnet_7.3-14          cli_2.2.0            survival_3.2-7
## [37] magrittr_2.0.1       crayon_1.3.4         readxl_1.3.1
## [40] evaluate_0.14        fs_1.5.0             fansi_0.4.1
## [43] nlme_3.1-150         MASS_7.3-53          xml2_1.3.2
## [46] class_7.3-17         tools_4.0.3          hms_0.5.3
## [49] lifecycle_0.2.0      munsell_0.5.0        reprex_0.3.0
## [52] e1071_1.7-4          compiler_4.0.3       rlang_0.4.9
## [55] grid_4.0.3           iterators_1.0.13     rstudioapi_0.13
## [58] labeling_0.4.2       rmarkdown_2.5        gtable_0.3.0
## [61] ModelMetrics_1.2.2.2 codetools_0.2-18     DBI_1.1.0
## [64] reshape2_1.4.4       R6_2.5.0             lubridate_1.7.9.2
## [67] utf8_1.1.4           stringi_1.5.3        Rcpp_1.0.5
## [70] vctrs_0.3.5          rpart_4.1-15         dbplyr_2.0.0
## [73] tidyselect_1.1.0     xfun_0.19
```