Spotify Popularity Predictor

Machine Learning Engineer Capstone Project
Pablo Assoni Raiter

December 22, 2020

Abstract

An independent record label is aiming to improve their artists album sales. To do it, they intend to apply Machine Learning in order to sculpt their songs in a more comercial way. The first attempt to do it is using Spotify Popularity feature. This feature measures a song success based on how many listenings a song had. The label firmly believe that some features can be good predictors of a song success and they want to know what are them. Also the label wants to predict with maximum accuracy how well a song will perform on Spotify every time a new song is releasen. We will create a model based on 19k songs available on Spotify and its main features, such as energy, key, loudness and many others.

1 Project Overview

Music business is getting more and more competitive. New record labels are being created day after day and everyone wants to be on the top charts. More recently Machine Learning became part of the music market. Record labels are using algorithms to find the features that make a song popular. Also record labels are using these algorithms to predict if a song will be a hit or not, i.e. its popularity.

Mostly music streaming platforms are providing data about songs features and its popularity. Pham, Kyauk and Park (2015) recently published a study providing results about relevant music popularity attributes. In this sense, we want to validate its results and create a model capable of predicting a song popularity based on Spotify data.

2 Problem statement

Our independent record label wants to achieve more success with its artists songs. In this project we aim to create a model capable of detecting key features for a popular song and capable of predicting if a song will be popular or not. To do it, we will use as a target Spotify popularity feature. This feature is an index based on how many plays a song had and how recent are those plays. To create our model, we also intend to use Spotify data. We intend to work with the following features: key, valence, tempo, speechiness, energy, loudness, and many others.

Our dataset need to be treated to serve as input to our model. As a benchmark, we will use Linear Regression, one of the simplest models available on Machine Learning. After that, we will try to test several models in order to find if we can beat Linear Regression or not. We expect to reduce our error to the maximum and to provide a model that really fits the data and give insights about it.

3 Datasets and inputs

In this project we will use a dataset available on Kaggle called "19000 Spotify Songs". It contains several features available on the platform that may be interesting to predict a song performance.

Initially we intend to use 13 features: danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration and time signature.

The descriptions of the variables are the following:

- Danceability (float): how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity.

- Energy (float): a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity.
- Key (integer): the estimated overall key of the track.
- Loudness (float): the overall loudness of a track in decibels (dB).
- Mode (integer): mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived.
- Speechiness (float): speechiness detects the presence of spoken words in a track.
- Acousticness (float): a confidence measure from 0.0 to 1.0 of whether the track is acoustic
- Instrumentalness (float): predicts whether a track contains no vocals.
- Liveness (float): detects the presence of an audience in the recording.
- Valence (float): a measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track.
- Tempo (float): the overall estimated tempo of a track in beats per minute (BPM).
- Duration (integer): the duration of the track in milliseconds.
- Time Signature (integer): an estimated overall time signature of a track

As expected, our dataset contains over 19000 songs observations, containing a few missing values. We believe it will be enough to fit a regression model capable of providing consistent results. Our data is not previously treated and need some processing. Also we will need to divide our dataset into train and test datasets in order to test our model without biased data.

You will find the dataset in: https://www.kaggle.com/edalrami/19000-spotify-songs.

4 Benchmark model

We have chosen Linear Regression as our benchmark model. Linear Regression is the most popular and known Machine Learning model and it is a good benchmark model to verify if more complex models really provide more accurate results. Because of its simplicity it is probable that our model will not perform very well on our complex dataset. Since we are not really trying to look for causality in our variables and we are aiming just to provide a main benchmark predictor, we believe that many statistical hypothesis could be dropped with the purposes of predicting only. Anyway we intend to at least provide normalized data to our model, since it is very well-known that General Linear Models handle data better if it is normalized.

A Linear Regression formula is provided below:

$$y = x_1\beta_1 + x_2\beta_2 + ... + x_i\beta_i + \varepsilon \tag{1}$$

5 Solution statement

In order to beat our benchmark model, we will test the following Machine Learning models:

- K-Neighbors Regression: non-parametric method for regression based on the average values of the k nearest neighbors.
- Lasso Regression: regression analysis methods that performs both variable selection and regularization in order to enhance the prediction accuracy.
- Ridge Regression: regression analysis methods that performs both variable selection and regularization, being useful to mitigate the problem of multicollinearity.
- Decision Tree Regression: model based on decision trees where target variable output is a continuous value.

- Random Forest Regression: ensemble learning method that operate by constructing a multitude of decision trees and take the average of the results.
- Extra Trees Regression: similar to Random Forest Regression, but Extra Trees does not use bootstrap method.
- Gradient Boosting Regression: ensemble method for regression which produces a prediction model in the form of an ensemble of weak prediction models, such as decision trees.
- Adaboost Regression: Adaptive Boosting is a meta-algorithm where the output of the weak learners is combined into a weighted sum that represents the final output of the boosted classifier.

The best way to search for a perfect Machine Learning model for the data we are working is simply to test for a bunch of them. Since we are working on AWS SageMaker environment, it will be easier than using local computational power.

6 Evaluation metrics

In order to evaluate our model, one metric will not be enough to see its performance. We will use four known metrics to get the properly results: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R-Squared (R2). All these metrics serve for different purposes and address model performance in different ways.

In this way we can definitely validate if our model will beat the benchmark Linear Regression model: we can use MAE, MSE and RMSE to verify the model mean error (its mean variance from the real result) and we can use R-Squared to verify the model fit. If our MAE, MSE and RMSE presents low values we are closer to a consistent model. If our R-Squared is closer to one, it means that our model is having a good fit to the data (if we do not put irrelevant data to the model, of course). The metrics formulas are provided below:

$$MAE = (\frac{1}{n}) \sum_{i=1}^{n} |y_i - x_i| \tag{2}$$

$$MSE = (\frac{1}{n}) \sum_{i=1}^{n} (y_i - x_i)^2 \tag{3}$$

$$RMSE = \sqrt{(\frac{1}{n}) \sum_{i=1}^{n} (y_i - x_i)^2} \tag{4}$$

$$R2 = SSR/SST \tag{5}$$

where:

- SSR: Sum of Squares of Regression.
- SST: Total Sum of Squares.

7 Project Analysis

In this section we are going to describe every step performed in our project, as well its motivations and its results.

7.1 Importing Data

Originally our dataset contains 45 features. The majority of these features are merely descriptive, i.e, they do not aggregate explanation to our model. The first transformation we are going to perform in our data is to drop 28 descriptive columns, such as: "disc number", "album id", "name", etc.

Although pandas has successfully imported our data, we still need to perform some datatype conversion. These trasformations are necessary because some variables were read as "strings", so in order to input these variables in our modelling we need to transform them in "numeric".

Finally, there are some problems with lines 7398 and 12599. These lines contains API links strings where some numerical values were supposed to be shown. A quick fix is to simply drop them, since it is not going to prejudice our model.

7.2 Exploratory Data Analysis

In this subsection, we are going to perform some exploratory data analysis in our target value (popularity) and in our predictors. We expect to get some knowledge about the data and extract some insights that may be useful for our modelling.

7.2.1 EDA: Target Value

We can gain some insight about how data is distributed by presenting a histogram and a boxplot.

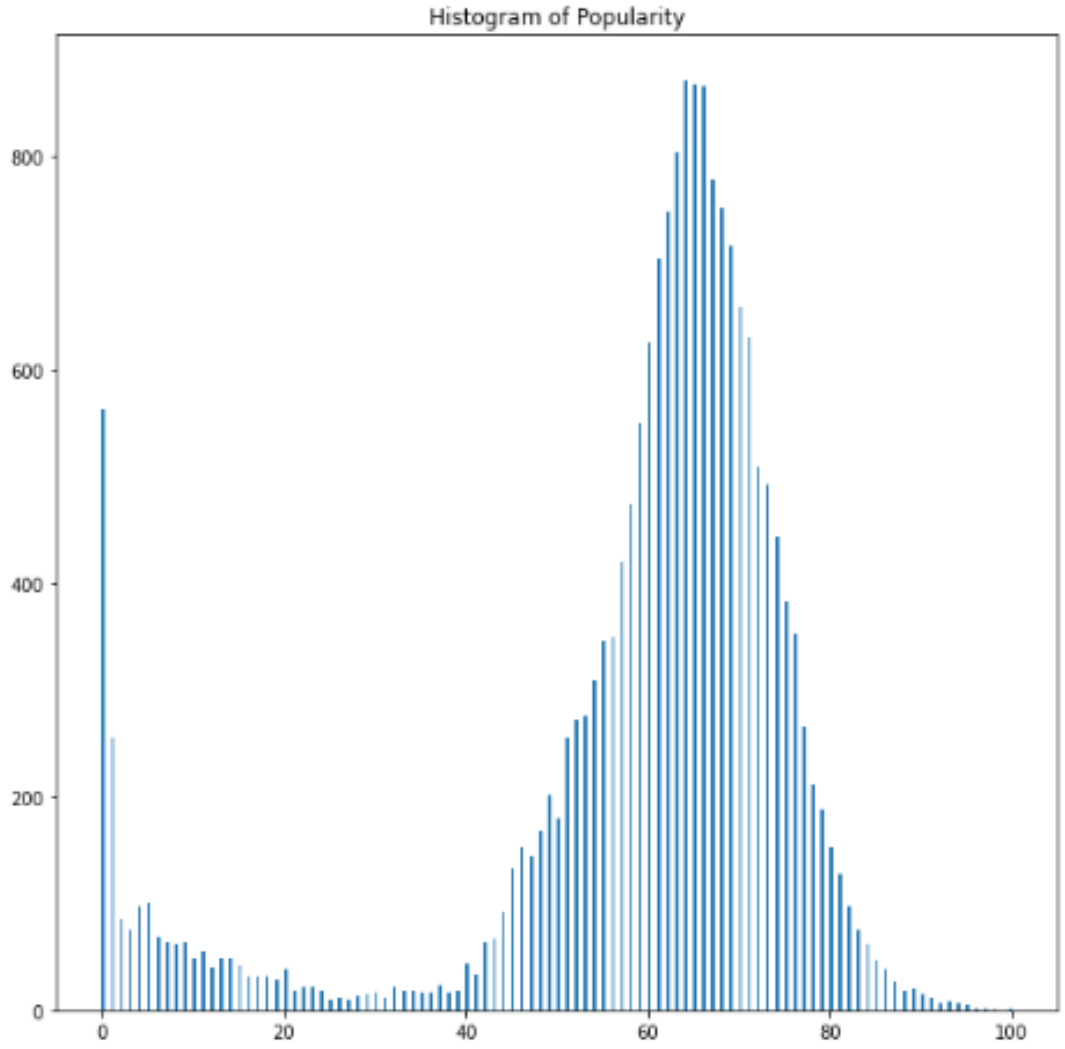

Figure 1: Histogram of Popularity

It seems that our popularity variable is somehow normally distributed around a value between 60 and 70. There are some outliers with very low popularity and some other outliers that are super popular songs. There is also a huge amount of zeros, i.e., songs with no popularity at all.

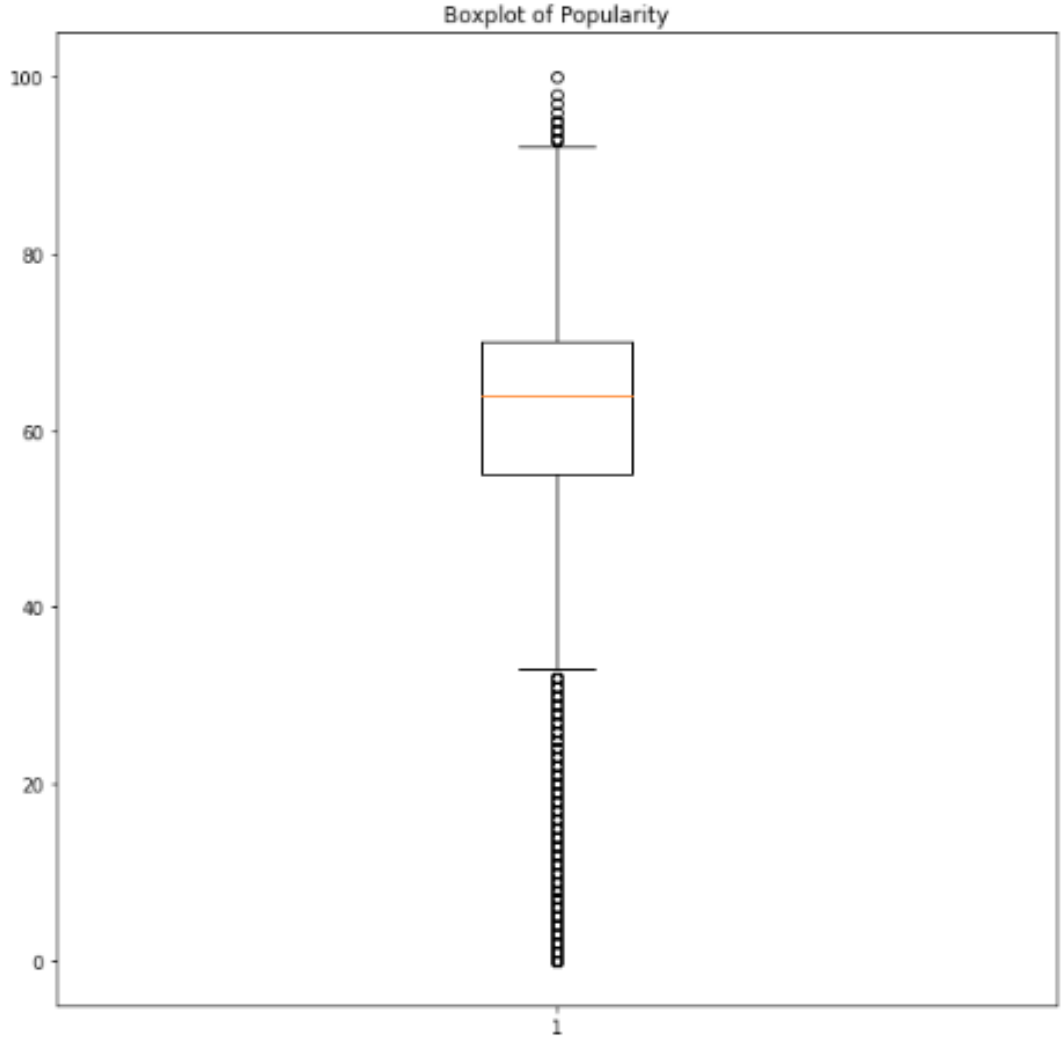

Figure 2: Boxplot of Popularity

It seems our data is centered around 55 and 70. Besides the huge amount of songs with no popularity at all, we can see that our variable has a considerable standard deviation. This value may serve as a reference to check our model performance later, comparing it to our score metric (RMSE).

7.2.2 EDA: Selected Features

Now we turn our attention to our selected features. The first step we are going to perform is to plot a correlation heatmap that may indicate some high correlated variables. High correlated variables may be transmitting the same information to the model, and it is good practice to drop them in order to get better results and avoid perfect multicollinearity.

Although there are some high correlated features, they may not be representing the same information.

We'll choose to maintain our original features. The correlation heatmap is provided in the next page.

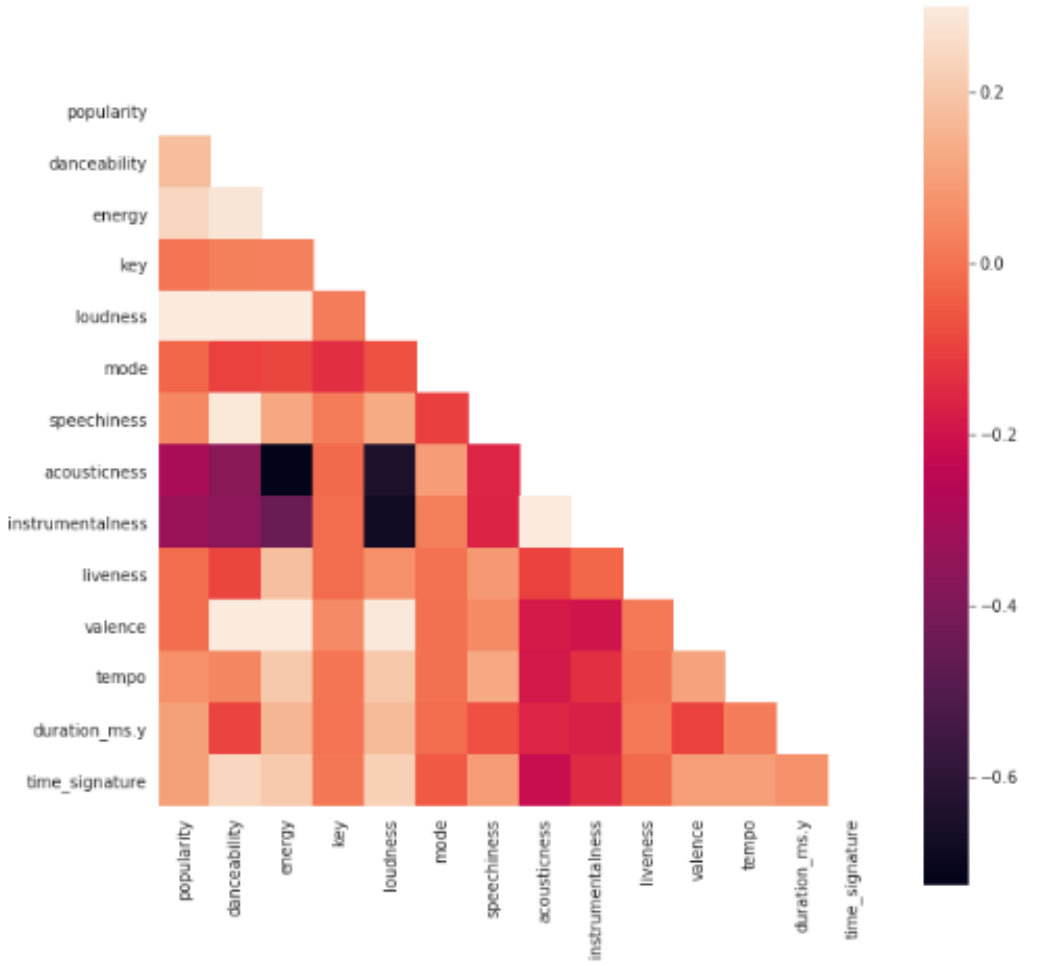

Figure 3: Correlation Heatmap

7.3 Data Pre-Processing

In order to gain more insights about the data and to prepare it for the model, we need to perform some pre-processing. First we will handle categorical values, perform data normalization and divide our dataset into train and test splits. After that, our dataset might be ready to be sent to the modelling step.

7.3.1 Handling Categorical Data

Since we do not have so many categorical columns, using pandas function "get dummies" will generate the dummies columns we are looking for. In order to avoid perfect multicollinearity, the first new column generated will be dropped.

7.3.2 Normalizing Data

Generally the variables that presents high variance in a dataset are more important to the model. However, in our dataset our variables are presented in different scales. In this way, it might get harder to compare different statistical attributes between our features. Also, there is huge evidence that Machine Learning models perform better if data is previously normalized.

Since we do not want to normalize categorical features, our best shot is pandas "MinMaxScaler". Min-Max Scaling reduces our model range between 0 and 1. In this way, we can compare standard deviation between features and find those who are the most promising ones.

Our top features in terms of standard deviations are:

- Mode: 0.4784

- Acousticness: 0.3232
- Key: 0.3244
- Valence: 0.2428
- Energy: 0.2369

We hope that these features can aggregate to the model performance.

7.3.3 Separating Data into Train and Test Splits

We need to evaluate our model performance. It is only possible if we divide our dataset into train and test dataset.

Fortunately, Scikit Learn provides a fast way to perform this split. All we have to do is to invoke "Train Test Split" function and define a test size. Also, is good practice to provide a random state for the purpose of reproducibility of the project.

Since we have 19k observations in our model, we will use 4/5 of the data to train our model and 1/5 to test it.

7.4 Model Training

In this section we train the models, compare it to the benchmark model and chose the final one.

7.4.1 Benchmark Model: Linear Regression

The general results of Linear regression using the metrics we defined in Section 6 are the following:

Table 1: Linear Regression results

	MAE	MSE	RMSE	R2
Linear Regressor	12,64	315,57	17,76	0,2038

Our benchmark model did not performed too good. Although we have a low RMSE (16.58) our R-Squared is too low. It means that our Linear Regressor did not fit too well to the data. It simply could not capture its variance. We may look for more complex models that may improve our RMSE and R-Squared results.

7.4.2 Machine Learning models

The results of our selected Machine Learning models are provided below:

Table 2: Selected Machine Learning models results

	MAE	MSE	RMSE	R2
K-Neighbors Regressor	10,45	236,34	15,37	0,4037
Lasso Regressor	12,92	350,20	18,71	0,1165
Ridge Regressor	12,64	315,57	17,76	0,2038
Decision Tree Regressor	12,04	329,60	18,15	0,1684
Random Forest Regressor	8,92	169,21	13,01	0,5731
Extra Tree Regressor	8,70	162,42	12,74	0,5902
Gradient Boosting Regressor	10,78	240,86	15,51	0,3923
Adaboost Regressor	15,60	368,53	19,19	0,0702

As we can see, Random Forest Regression and Extra Trees Regression performed similarly well. There are no significant differences between both models. Let's stay with Random Forest since it is more well known and it uses more robust methods, like bootstrap sampling.

7.4.3 Choosing the model

The chosen model performed this way:

Table 3: Final model results

	MAE	MSE	RMSE	R2
Random Forest Regressor	8,89	167,75	12,95	0,5768

These results are promising in comparison to the benchmark model. We will compare it in the next section.

8 Final Results

In this section, we will summarise the final results of our project.

8.1 Feature importance

One of the main objectives of the independent record label was to distinguish what variables were determinant to a song success or failure. What variables really matter to popularity?

It is possible to verify feature importance of our Scikit Learn models. Our Random Forest Regressor most important features are provided in the graph below:

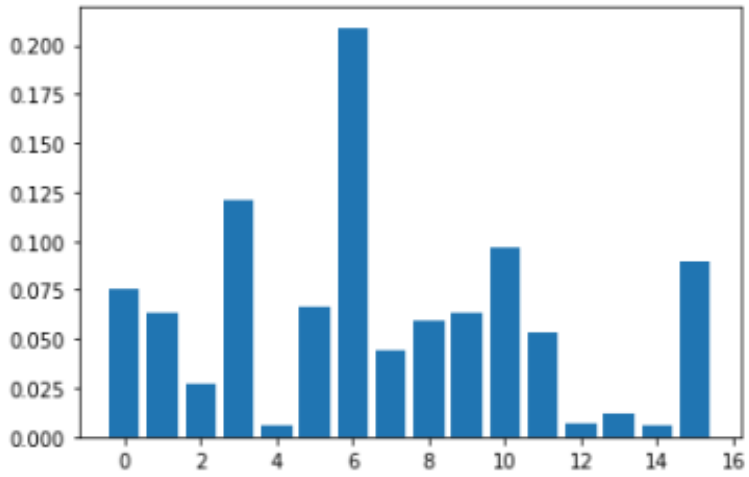

Figure 4: Feature Importnace

Our corresponding top performer features are:

- Speechiness - feature 6: 20,89%
- Key - feature 3: 12,10%
- Valence - feature 10: 9,65%

It seems Speechiness, Key and Valence are the top responsible features in determining a song commercial success.

As we saw before, both Key and Valence were two variables which had relative high variance in comparison with other variables. Although both Acousticness and Mode had higher variance than Key and Valence, they were not capable of explaining a success of a song. In this way, the focus of the record label may be to study how these features are related with a song success. It will allow producers to reshape its productions in order to increase sales.

Table 4: Model comparison

	MAE	MSE	RMSE	R2
Linear Regressor	12,64	315,57	17,76	0,2038
Random Forest Regressor	8,89	167,75	12,95	0,5768

8.2 Performance Comparison to the Benchmark Model

The table above summarises the final results between the benchmark model and the chosen one.

Our Random Forest Regression model had a superior performance in comparison to the benchmark model. It was over 30 p.p. R-squared gain over the benchmark. Also our RMSE lowered 5 p.p., indicating that this model is superior and capable of providing more accurate predictions of a song popularity.

9 Conclusion

The intention of this project was to develop a robust solution in song popularity prediction, finding the variables that matters to a song success and providing a model with the lowest possible error. We achieved it providing a model that had a superior fit to the data and lower comparative error.

Initially, we imported data and pre-processed it. We dealt with some inconsistencies in data and dropped irrelevant columns. After that, our path was clear to exploratory data analysis.

Performing exploratory data analysis, we acquired insight about our target value. Also we gained insight about general feature statistics and found whose were the most disperse along the distributions. These were elected as probable good predictors of a song performance.

After that, we made some final transformations in our data in order to feed our model. These transformations were: categorical features treatment, data normalization and separating data into train and test splits.

Initially, we trained our benchmark model to see how it performed. It had a poor performance, with a low R-Squared, indicating a low fit to the data. After that, we trained substantially more complex Machine Learning models and we got surprising good results. In our final model, we had over 30 p.p. R-squared gain over our benchmark and our RMSE lowered 5 p.p.. We also discovered which features were important to our modeling and may affect a song popularity. The producers of our independent record label might take a serious look on these variables.

References

[1] James Pham, Edric Kyauk, and Edwin Park. *Predicting Song Popularity.* Stanford University, 2015.

[2] Naomi S. Altman. *"An introduction to kernel and nearest-neighbor nonparametric regression"*. The American Statistician. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879. hdl:1813/31637, 1992.

[3] T. Hastie, R. Tibshirani, J. Friedman *The Elements of Statistical Learning (2nd ed.)*. New York: Springer. pp. 337–384. ISBN 978-0-387-84857-0.

[4] Spotify for Developers. *Get Audio Features for a Track.* Available on: https://developer.spotify.com/documentation/web-api/reference/tracks/get-audio-features/.