Capstone Project
Spotify Song Popularity Prediction

1. Definition

Project Overview

Music plays an integral part in our lives. Listening to music has never been so easy with a plethora of music apps streaming music with paid/unpaid membership offering any song with a simple click. Spotify is the largest music streaming service provider, and often the popularity of a song on Spotify reflects the popularity of the song in general. I've pondered on the question of why some songs become hits while some do not make it. While I or any individual may have a specific taste, it's interesting to find out what makes a song popular.
Here in this study, we try to answer what determines the popularity of a song. The methods applied are simplistic but provide promising results. The work done can also be deployed for people to use outside of the developer's environment.

Problem Statement

How can we determine the popularity of a song based on its features?

Evaluation Metric

Popularity as a target variable for prediction has continuous values ranging from 0 (least popular) to 1 (most popular). Since the problem at hand is regression, we use a classical evaluation metric for regression tasks i.e., RMSE. The goal is to minimize RMSE.

Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

2. Analysis:

Dataset information from Spotify

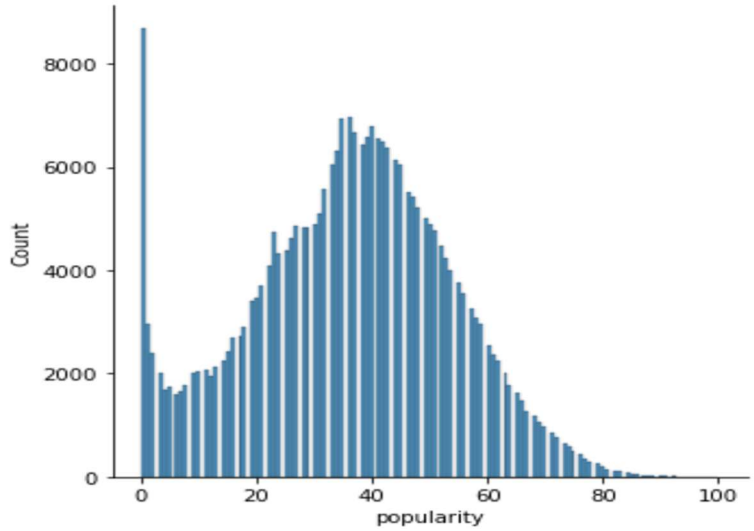
Spotify has provided information of each attribute on their website. The attribute information can be found here:
https://developer.spotify.com/documentation/webapi/reference/#endpoint-get-track

Following are the attributes and their types:
- id: string
- name: string, name of the song
- popularіy:continuous, ranging from 0 to 100
- duration_ms: continuous, duration of the song in ms
- explicit: categorical, determines whether the song contains explicit content or not (0/1)
- artists: list of strings, names of the artists
- id_artists: list of strings, ids of the artists
- release_date: date
- danceability: continuous, determines how suitable is the song for dancing (ranges from 0 to 1)
- energy: continuous, determined how energetic is the song (ranges from 0 to 1)
- key: categorical, major key of the track with 11 unique values
- loundness: continous, value in dB
- mode: categorical, track is major (1) or minor (0)
- speechiness: continous, ratio of spoken words to overall, 0(instrumental) and 1 (talk show)
- accousticness: continous
- instrumentalness: continous
- liveness: continous, 0 (studio recorded) and 1(concert)
- valence: continous, how positive is the music 0 (sad) and 1 (cheerful)
- tempo: continous, tempo of track in BPM
- time_signature: continous, specifies how many beats are in each bar

There are in total 279,057 observations and 19 variables.

Exploratory visualization

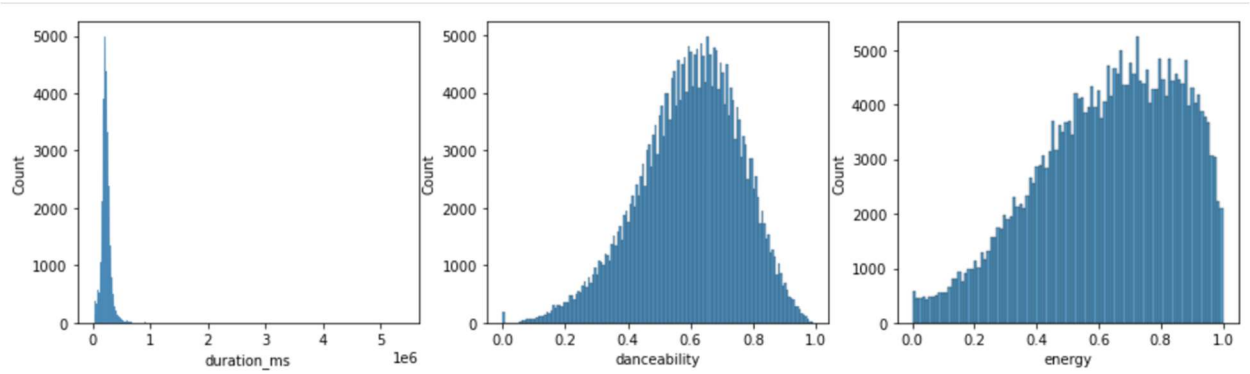
1. Output Variable: Popularity



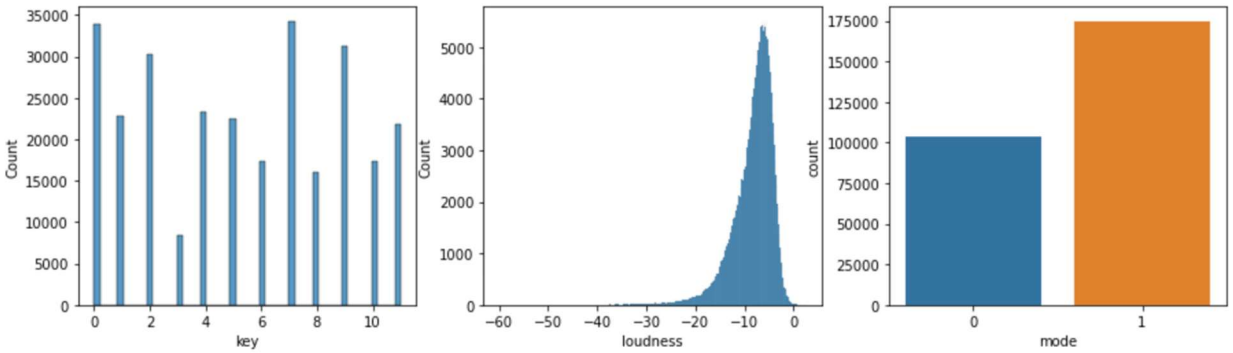
The mean value of popularity is 35.9

2. Input Variables

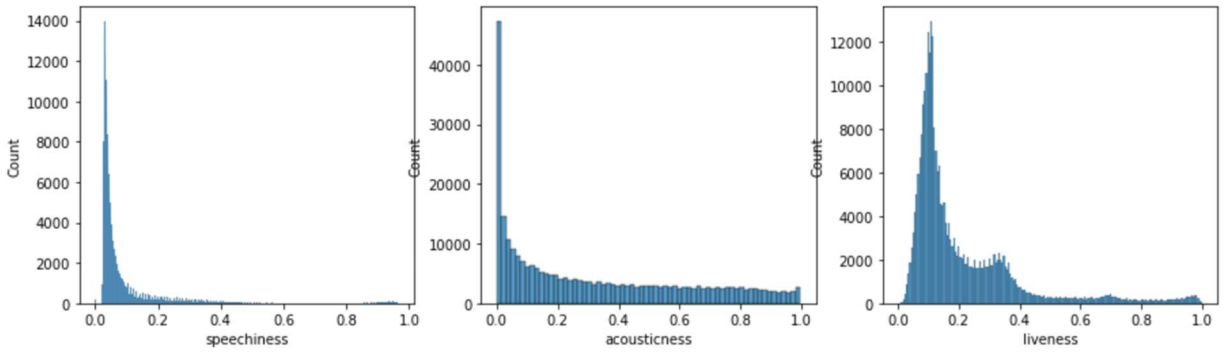
Duration, danceability and loudness



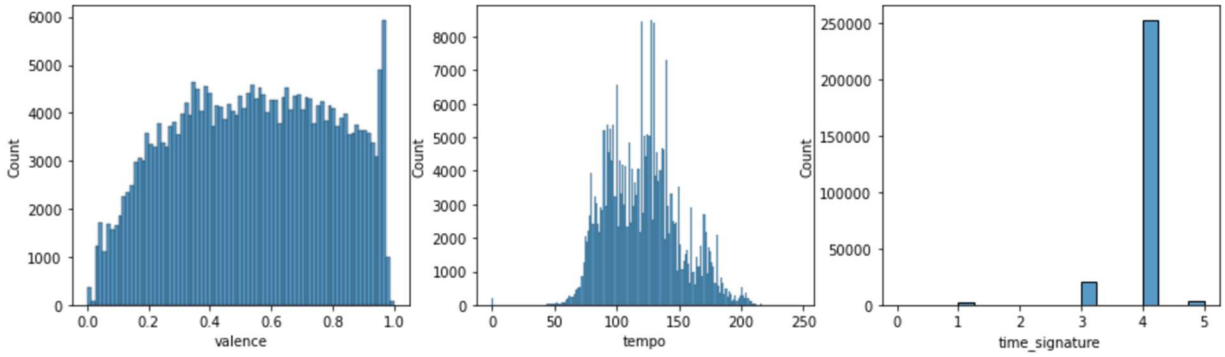
Key, loudness and mode



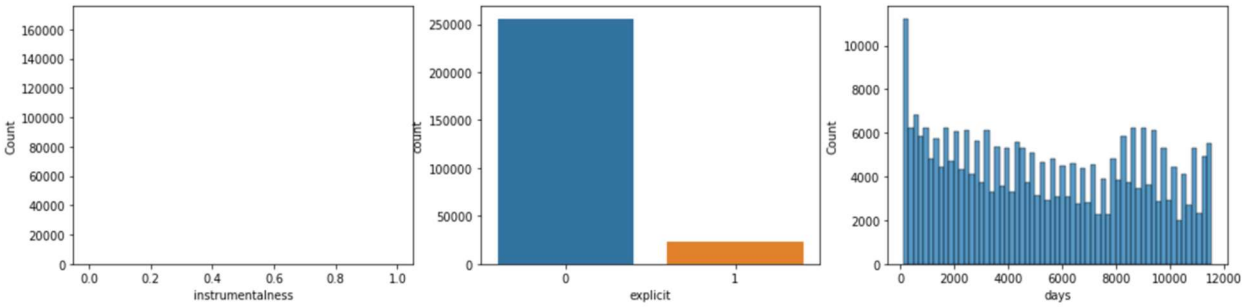
Speechiness, acousticness and liveness



Valence, tempo and time_signature



Instrumentalness, explicit and days (



We observe most songs do not have explicit content

Algorithms and Techniques

The problem at hand is a regression problem. In this study, I've tried to apply the simplest techniques that can be explainable first and then moved to other complex methods. The general approach while applying an algorithm has been:

- Select a type of algorithm
- Tune parameters of the algorithm to get the best metrics on the validation set
- Record the results on training and validation set

Some of the algorithms tried in this study are:

- Linear models: Ridge regression with different alpha parameters to minimize RMSE on the validation set.
- Tree-based models:
    - Decision Tree
    - Random Forest
    - XGBoost
- K-Nearest Neighbors

Benchmark Model

Null Model: In case we do not observe any attributes, our best guess would be the average value of the popularity rating. Hence, the benchmark model or null model predicts the average value of the training set for the validation set and test set.

3. Methodology

Data Preprocessing

In the study these are some of the steps taken for data preprocessing:

- Dropping null values and duplicates
  There are 3 observations with null values and there are no duplicates. Since, we do no lose much information by dropping 3 observation relative the large size of the dataset we drop them.

- Converting type of variable
  Categorical features are already encoded and have only 2 values. Continuous features are converted to float.

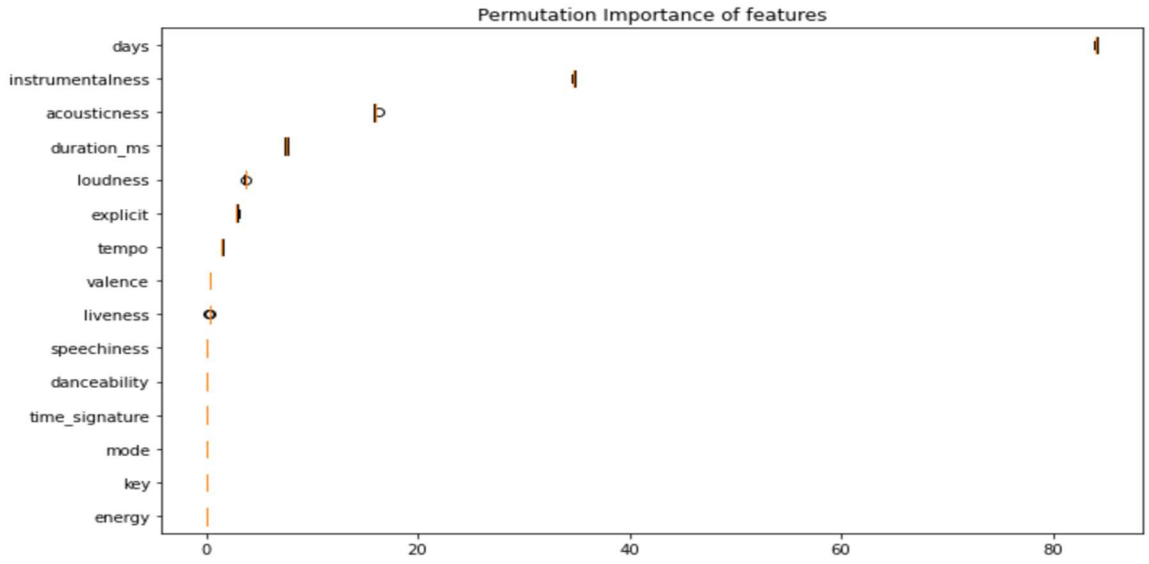
- Dropping variables
  Variables such as id, name, id_artists are dropped as they're identifiers. There are too many unique artists and hence this variable too is dropped.

- Encoding variables
  Variable release_date is encoded as days since release date. Since days will be a number it will be easier to handle for modeling.

- Dividing data into train, validation, and test
  Since we a large dataset we reserve randomly selected 1000 observation for validation and test sets while rest are used for training.

- Normalizing Variables
  All continuous variables are normalized using Min-Max scaler technique. This brings all variables within [0,1] range.

- Feature Selection
  A random forest regressor techinique is used for feature selection. Details of the technique can be found:
  https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html
  We finally select variables with some importance compared to its randomly permuted copy. The diagram below shows the variables sorted by their importance.



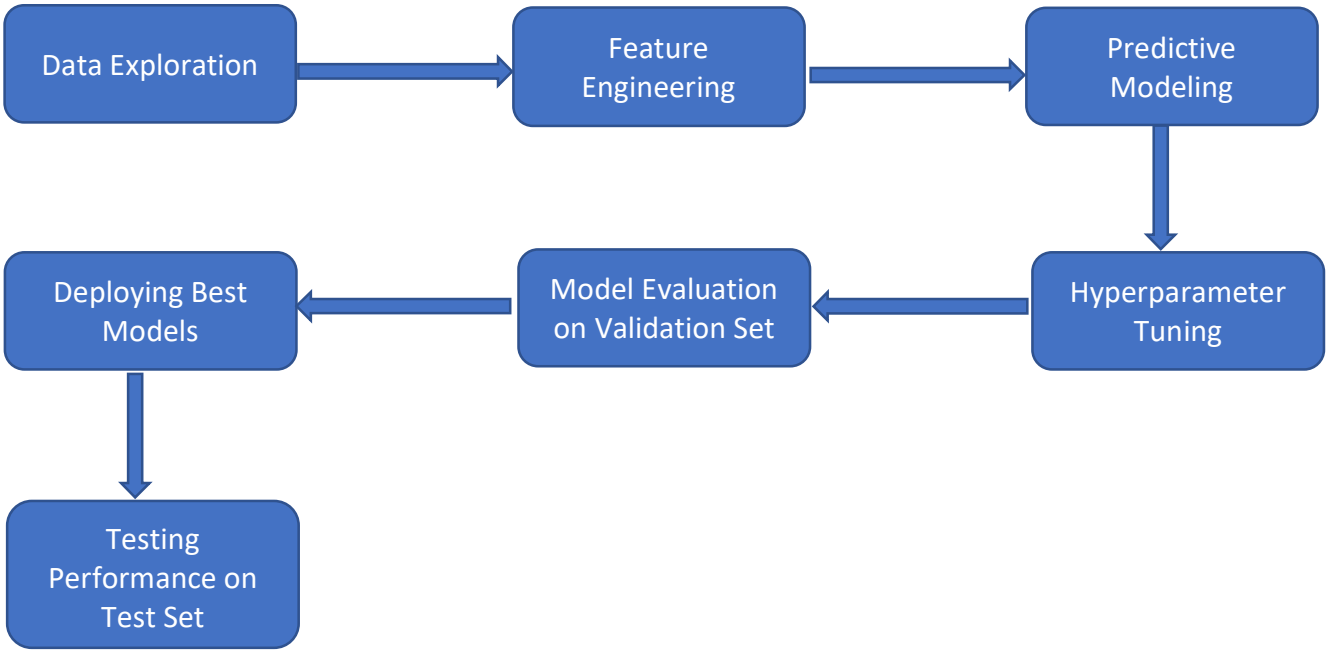
Following variables are selected:

- days
- instrumentalness
- acousticness
- duration_ms
- loudness
- explicit
- tempo

Most of the data pre-processing tasks are done by importing a python script with custom functions: data_processing.py

Implementation

Following flow captures the implementation



The steps are as follows:

- Data Explored using summary statistics and visualizations
- Features are engineered to suit the task at hand. The previous section on data preprocessing captures the details more in-depth.
- Predictive Modeling is done only on the training set.
- Hyperparameters are tuned with the Grid Search Cross-validation technique on the validation dataset.
- The model with the best parameters is again tested on the training and validation set and the results are recorded.
- Top models with the best validation set results are deployed at an endpoint
- The test set is run on the deployed models and the results are tabulated

Following algorithms are implemented:

- Ridge regression: SKLearn
- Decision Trees: SKLearn
- Radom Forest: SKlearn
- XGBoost: Sagemaker
- K-Nearest Neighbors: SKlearn

Refinement

In this study, we have used the Grid Search Cross-Validation technique from the SKLearn library for the task of hyperparameter tuning. The parameters are pre-decided based on an understanding of the algorithm and given a range suitable enough so that the best model hyperparameter does not exceed the limit. In the case of, XGBoost we use Sagemaker's built-in HyperparameterTuner() method. Intermediate results are checked and the best hyperparameters are selected for testing and deployment.

Deployment

Best models are deployed for testing on the test set. Following are the best two models selected based on Validation set RMSE:

- Random Forest
- K-Nearest Neighbors

4. Results

Model Evaluation and Validation

Following table showcases the performance of different algorithms on training sets and validation sets:

Classifier	Training set RMSE	Validation set RMSE
Null Model (Benchmark)	17.71	17.48
Linear Regression (Ridge)	16.52	16.59
Decision Tree	15.19	16.05
Random Forest	15.06	15.79
XGBoost	15.01	15.89
K-Nearest Neighbors	1.17	15.81

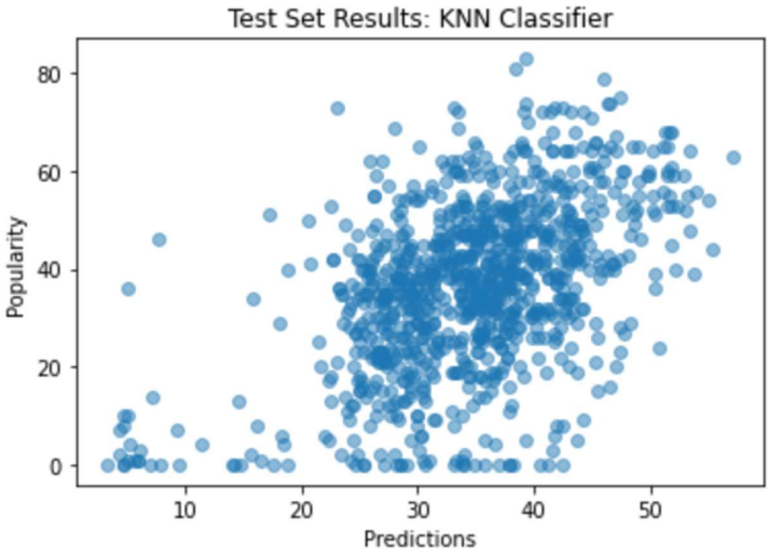
Training set RMSE can be a bit misleading if the model is overfitted on the data e.g., K-Nearest Neighbors. Hence, if we decide the model based on both Training set RMSE and Validation set RMSE for performance. Overall, Random Forest and K-Nearest Neighbors perform are the most robust in predicting the popularity.

Deployed models are tested on test set, the results are as follows:

Classifier	Test set RMSE
Null Model	16.91
Random Forest	14.98
K-Nearest Neighbors	14.78

Justification

Both deployed models performed better than the null model. As we see there is a significant drop in the RMSE we can conclude that the deployed model is better at predicting the popularity of the song given its attributes. Following chart shows some of the predictions on test set:



As we see from the chart above, majority of the predictions would lie on / around diagonal straight line which would give no error. Hence, we can conclude we have done a decent job of predicting popularity of the song.

Future Scope

In addition to the work done, there are many ways we can improve the performance and the usability of the task. Following are some ways we can improve the current work:

- Use artist name as one of the variables
- More feature engineering to get the best predictions
- Deploy model and make it available outside of developer environment with AWS Lambda