Predicting Song Popularity Using Spotify's Audio Features Dataset

Data 1030 Fall 2021 | Brown University | December 7, 2021

Mark A. Adut, M.Sc. Candidate in Data Science

1. Introduction

Recommendation algorithms are being increasingly incorporated in our daily lives. Companies such as Google, Facebook, Amazon, Apple, and Spotify are utilizing these technologies to further optimize their core product offerings and attract more customers. This project explores the ways in which audio streaming companies such as Spotify provide their customers with tracks/albums that align with their listening preferences. For this ML project, the *Spotify Audio Features* dataset[1] was used. The dataset consists mainly of technical sound features such as *song loudness*, *speechiness*, *energy*, *danceability*, *tempo*, etc. The target variable in the Spotify Audio Features dataset is *popularity*, a variable that represents how popular a track is, with values ranging from 0-100. The goal of this project is to, through finding statistically significant patterns between feature variables and the target variable, predict the popularity of any given, unheard song. To achieve this goal, regression analysis will be used.

One of the previous works that has derived valuable insights from the *Spotify Tracks DB* dataset is composed of contrasting the accuracy and performance of various ML methods such as K Nearest Neighbor (KNN) Classification, Logistic Regression, Random Forest Classification, Decision Tree Classification, Linear Support Vector Classification, and XGBoost[2]. The key findings of that data analysis were as follows: Random Forest Classifier (RFC) had the highest accuracy score with a score of 92%, and Linear SVC maintained the lowest, with an accuracy score of 69%. The goal of this project is to achieve an accuracy score that is higher than the existing ones published by previous authors.

2. Exploratory Data Analysis (EDA)

Exploratory data analysis is an integral step in any data analysis process. To initiate the EDA, the size of the dataset was checked. The dataset consists of 228,159 rows and 18 columns. Specifically, the column names are as follows: genre, artist name, track name, popularity, acousticness, danceability, duration_ms, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo, time_signature, valence. As the next step of the EDA, descriptive statistics were utilized by calling Python's *.describe* function as well as *.value_counts*. Some of the important statistics retrieved were the mean of the target variable, which stood around 44.209, and that the top 25 songs had a popularity level greater than or equal to 57. The *mean* value would especially prove effective later in the analysis as it was used in calculating the baseline regression (R2) score.

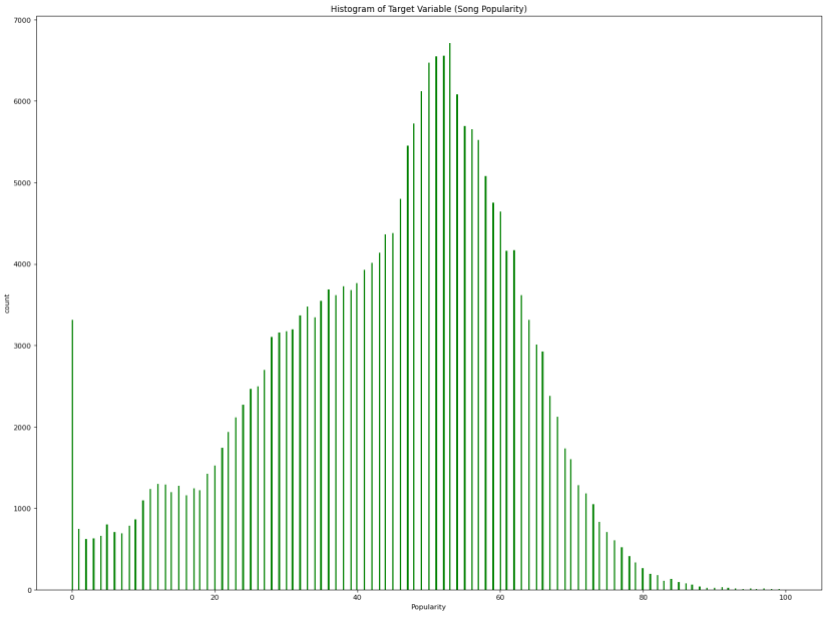

Figure 1*: Histogram of Target Variable (Song Popularity)*

Figure 1: The figure above displays a histogram of the target variable. At a first glance, it can be inferred that the plot is in alignment with the derivation made in the previous step. The mean of the target variable was calculated as 44.209; the accumulation of bins in the range 40-50 is in accordance with this result.

Figure 2: To get a holistic understanding of the relationship between feature variables and the relationship between the target and feature variables, a correlation matrix was generated.

[1] "https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-audio-features

[2] "Spotify Song Popularity Prediction", Tran, Steven., 2019, https://www.kaggle.com/huanntran100/spotify-song-popularity-prediction/notebook

Looking at the correlation matrix, the variables that have positive or negative correlations can be distinguished. For example, if we look at the cell at which *loudness*

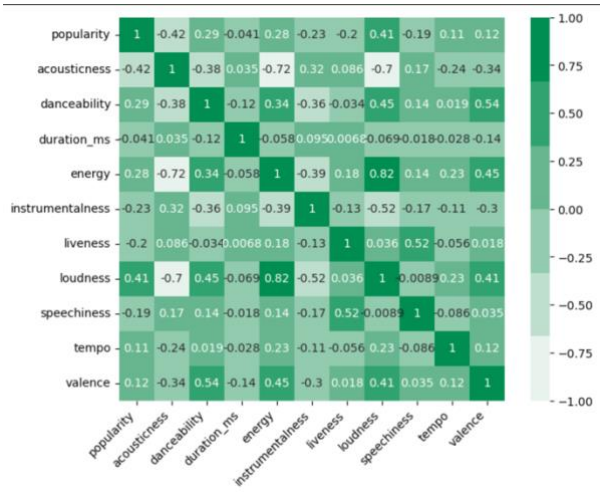

***Figure 2**: Correlation Matrix*

meets *energy*, it can be inferred that they are positively correlated (this is aligned with what is intuitively expected from an energetic song). An example of negatively correlated features can be given as *energy* and *acousticness* (-0.72). At a first glance, there doesn't seem to be a strong correlation between individual feature variables and the target variable. However, it will later be derived in this project that, feature variables, grouped and organized in a certain fashion, will have predictive capabilities.

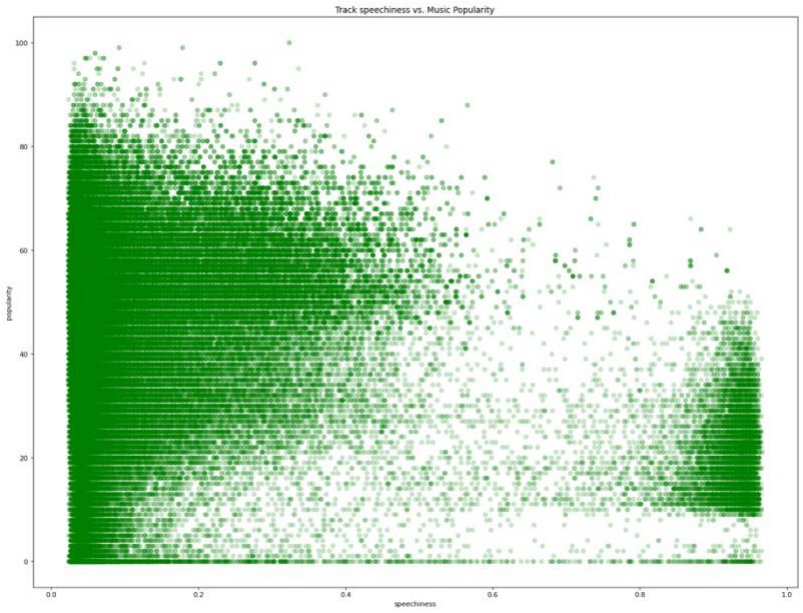

***Figure 3**: Scatter Plot Demonstrating Relationship Between Track Speechiness & Song Popularity*

As the final step of the EDA, the feature variables of the dataset were categorized as being categorical and continuous. This was a crucial step given that it was being incorporated in the preprocessing part of the data. The breakdown of the categorical variables of the dataset are as follows:

Musical Keys	A, A#, C, C#, D, D#, B, A, G, E, G#, F#
Musical Modes	Major | Minor
Time Signature	4/4, 3/4, 1/4, 5/4, 0/4
Genre	See Jupyter Notebook for Full List

***Table 1**: Categorical Features of the Dataset*

***Figure 4**:* To further explore the relationship between certain feature variables, a plot was generated using the Seaborn library that depicts the relationship between sound keys and music track popularity, grouped by sound modes (major, minor). The most popular key among the Major sound mode is C# (denoted as *C Sharp Major*). Open Mic UK, a British live music competition agency, describes this pattern as follows: *"There are some common keys for pop songs, for example. C major, G major, D major".* The widespread adoption of song genres such as pop, R&B, and HipHop, also reconfirms this pattern.

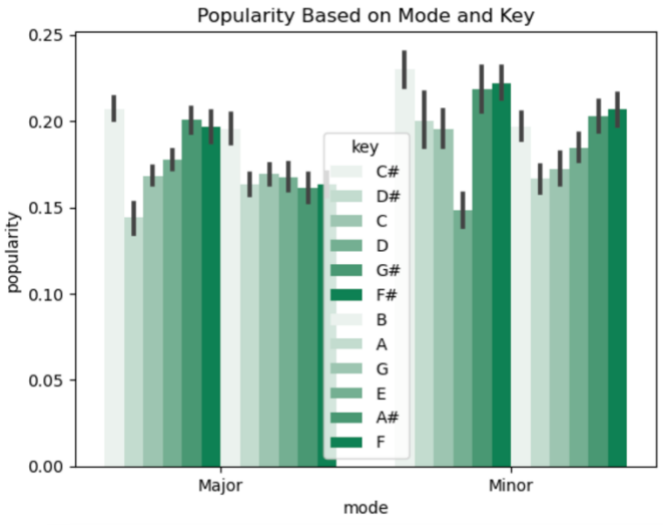

***Figure 4**: Song Popularity Based on Audio Keys and Modes*

3. Methods

Data Preprocessing, Splitting & Hyperparameter Tuning

Before initiating preprocessing, it had to be reconfirmed that the data is independent and identically distributed (iid). Since the data is not time-series, a group structure is non-existent, and each row is unique to each song, it can be derived that the data is iid. Another fundamental step taken before initializing preprocessing was to drop the column that stores songID. Since each songID is a unique key for each song, it has no predictive capacities. After separating the one-hot encoded and standard scaled features, Sklearn's preprocessing pipeline was initialized for fitting, scaling, and transforming the data. The OneHotEncoder was applied to the categorical features of the dataset because there isn't any specific hierarchical nature of the categories, such as in music modes (major key, minor key). The StandardScaler was applied to continuous features since they are not reasonably bounded and hence not suitable for the MinMaxScaler.

Before various models were deployed on the data, two functions, *MLPipe_R2* and *MLpipe_KFold_R2* were being tested to split, preprocess, and cross-validate the data. The first function is an example of basic hyperparameter tuning, and the second one applies cross-validation using a KFold split and the GridSearchCV functionality. After comparing the results, it was determined that the GridSearchCV had a better model performance enhancement than the basic hyperparameter tuning method. The *MLpipe_KFold_R2* function takes an unprocessed feature matrix, target variable, a preprocessor (ColumnTransformer), an initialized ML algorithm, and a corresponding parameter grid as inputs. Inside the function, the data is split into other and test (80-20) dataframes and then KFold is used with 4 folds. Later, the data is preprocessed, and cross-validation is performed, and lastly, test scores are calculated. This process was repeated 5 times for 5 different random states, and the function returns the Best Model Parameters, the Baseline R2 Score, the mean_test_score stored in GridSearchCV, the Validation score, and the 5 Test R2 Scores. An optimal evaluation metric had to be determined to assess model performances. The evaluation metric would be both used in the calculation of the validation score that the GridSearchCV function outputs, as well as in the calculation of the test scores. For this project, the *coefficient of determination* score, or the R2 score, was used as an evaluation metric. It was used to determine how well each models' regression predictions fit the data. Below are the parameters used to tune each model:

Model	Parameters
Lasso	{**'lasso__alpha'**: np.logspace(-4,2,7)}
Ridge	{**ridge__alpha**: np.logspace(-4,2,7)}
ElasticNet	{**elasticnet__alpha**: np.logspace(-4,2,7)}
RF	{**'bootstrap'**: [False], **'max_depth'**: [80], **'max_features'**: [2, 3], **'min_samples_leaf'**: [3, 4, 5], **'n_estimators'**: [100, 200, 300, 1000]}
KNN	{**'kneighborsregressor__n_neighbors'**: [1, 10, 30, 100]}
SVR	{**'svr__C'**: [0.1, 1, 10, 20], **'svr__gamma'**: [1e-7, 1e-4]}
XGB	{ **'gamma'**: [0.5, 1, 1.5, 2, 5], **‘'max_depth'**: [3, 4, 5]}

Table 3*: Parameters used for tuning of each model*

4. Results

4.1 Model Performances Comparison

After the appropriate parameters were tuned, the mean and standard deviation of the Test R2 Scores were stored for comparison (*See Figure 5 Below*)

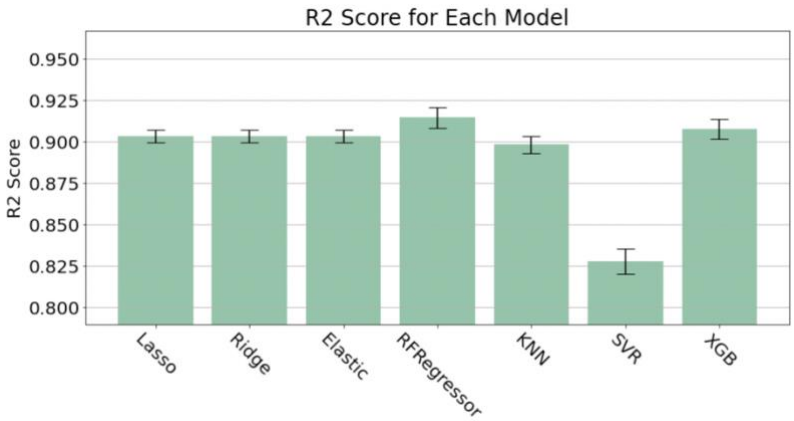

Figure 5*: Average R2 Scores for 7 Regression Models*

Holistically speaking, Lasso, Ridge, and ElasticNet regression had very similar performances with an average R2 score standing around 0.90. The best performing model was the Random Forest Regressor with a mean R2 test score of 0.91 and the second best mean R2 score was achieved through XGBoost Regressor. Moreover, the least best performing model appeared to be Support Vector Regression, with a mean R2 score of roughly .835. One important consideration that was made while testing the model performances was to train

the models on only a portion/sample of the data. When the whole data (roughly 230,000 rows) was being used, the local computer used was not able to complete a single random state iteration. In order to benchmark the R2 Scores of each model, a baseline R2 score had to be generated. Sklearn's DummyRegressor function with 'mean' as its strategy of was used as a simple baseline to compare with other regressors.

4.1 Final Model Formulation

After the best model performance and its respective hyperparameters were determined, the Random Forest Regression model was re-trained using its optimal hyperparameters. Since RandomForestRegression is a highly execution-intensive model and due to the size of the dataset, the final model was trained using 10 random states. Again, for each random state, the validation R2 score, the test R2 score, and the mean train score was recorded.

4.2 Evaluation of the Final Model

Over the 10 random states, the Test R2 Score of the best performing model with optimal hyperparameter running returned a mean Test R2 of .914 and a standard deviation of 0.011. The baseline model R2 score was earlier calculated in the Model Selection section as being 0. Therefore, it can be noted that the trained models achieved an R2 score that was roughly 80 standard deviations above the baseline model.

4.3 Interpretation of Findings (Local & Global Feature Importance)

Global Feature importance for the best performing model was calculated using the coefficients of the models and a permutation test over 10 shuffles was applied for each random state.

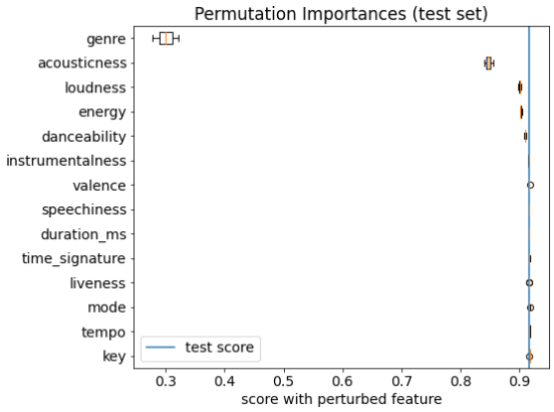

Figure 6: Permutation Importances for 14 Features

The permutation feature importance test was used as a model-agnostic test which followed the steps below:

1. A single feature in the test set is randomly shuffled
2. Test scores are recalculated using the shuffled data
   - The model score worsens as the shuffling breaks the relationship between the feature and target variable
3. The larger the difference, the more important the feature

Figure 6 compares the permutation importances for each perturbed feature. It can be noticed that *genre* has the largest difference due to shuffling/permuting, indicating high importance. This is in accordance with what could be anticipated; as certain genres are substantially more popular compared the others, it shows a higher predictive capacity. There were 2 additional global importance tests that were performed: a SHAP value calculation and scaled and not-scaled coefficient importances. Both analyses showed that the *instrumentalness* feature had the highest impact on the model output (See *Figure 7* Below for SHAP comparisons)

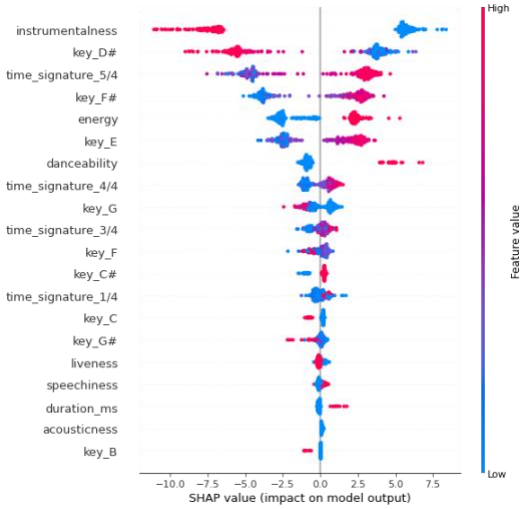

Figure 7: SHAP Values for Global Feature Importance

Lastly, a local feature importance calculation was executed to check whether we trust the model's prediction for one specific data point.

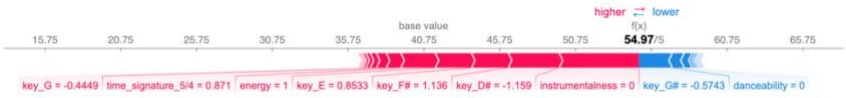

Figure 8: SHAP Force Plot for Local Feature Importance

Looking at the force plot, we compare which feature had the most influence on the model output given an incremental increase in data points. We notice that *instrumentalness*, again, had the largest impact on the model output.

5. Outlook

One aspect this analysis could benefit is the tackling of the problem of song names/artist names that are not encoded using the Latin alphabet. During the EDA portion of

this project, the non-alphabetical values in the dataset were calculated. These values constituted for a negligible portion of the data. Since Sklearn does not account for values that are not encoded in en_core_web_sm format, these values did not have a drastic impact on model performances. Additionally, the model would also highly benefit being trained on a high-performance computer that could run through more random states and ingest the whole data. Lastly, additional historical audio features data can be collected from Spotify's API to further improve model performance.

6. References

1. Spotify Audio Features, https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-audio-features
2. Mohamed Nasreldin., May 4, 2018., https://towardsdatascience.com/song-popularity-predictor-1ef69735e380
3. "Spotify Song Popularity Prediction", Tran, Steven., 2019, https://www.kaggle.com/huanntran100/spotify-song-popularity-prediction/notebook
4. "Guide to Time Signatures in Music'., Nov 2018., https://www.masterclass.com/articles/guide-to-time-signatures-in-music#:~:text=Time%20signatures%2C%20or%20meter%20signatures,the%20clef%20and%20key%20signature.
5. "How to decide what a key a song is in"., Open Mic UK., April 2020., https://www.openmicuk.co.uk/advice/how-to-decide-what-key-a-song-is-best-key-to-write/
6. Anat Peled, Kaggle, https://www.kaggle.com/anatpeled/spotify-popularity-prediction, 2020
7. https://docs.google.com/document/d/1YSJojnBuM97GLkpMlQoxMDW_eUiiDAx0Ja6beBwWpPs/edit#

GitHub: https://github.com/markadut/Spotify-Song-Popularity-Prediction