Analyzing And Predicting Popularity Trends in 30000 Spotify Songs

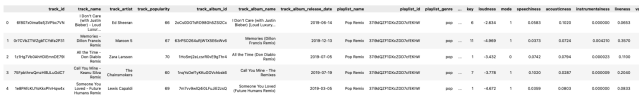
YIHONG LI, University of California, San Diego
ZICHEN YU, University of California, San Diego
ZEYANG YU, University of California, San Diego
ZIXIN WEI, University of California, San Diego

1 DATASET

1.1 Introduction

In this project, we used the “30000 Spotify Songs Data”[1] from the Kaggle website[2]. The dataset consists of approximately 30,000 songs records, featuring information such as unique song IDs, song names, artists, popularity scores, album details, playlist information, and various musical attributes like danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, and song duration. The dataset contains 32833 data and 23 columns in total and 32828 data and 20 columns after we dropped the rows containing null values and the meaningless columns.

Fig. 1. An Overview of Dataset



1.2 Data Cleaning

In this section, we eliminate redundant or irrelevant information by selectively excluding columns such as “track_id”, “track_album_id” and “playlist_id”. The resulting dataset, cleaned_data contains only essential features, optimizing it for subsequent modeling and analysis. We examined the dataset for missing values and found a total of 15 null values distributed across 5 rows. These null values are specifically present in the "track_name," "track_artist," and "track_album_name" columns. Due to the small number of missing values, we decided to drop these 5 rows from the dataset.

1.3 Exploratory Data Analysis

Following our exploratory data analysis, we found that the average track popularity stands at approximately 42.48. As depicted in Figure 2, it's evident that the most frequent popularity scores fall within the range of 0 to 10. When examining the distribution of track popularity, it closely resembles a symmetric distribution centered between 50 to 60, excluding the first bar. Consequently, we computed the percentage of songs with a popularity score of 0, yielding a result of 8%. Given that this percentage is relatively low, we decided to maintain this distribution for our dataset.

We then plotted the distribution for all the numerical features from the data to target the features valuable for the predictive model. Based on the observations from Figure 3, most of the numerical features exhibit symmetric distributions. Notably, variables such as 'speechiness,' 'acousticness,' and 'liveness' display negatively skewed distributions, indicating that their tails extend more towards lower values. Importantly, despite the variations in these distributions, no extreme outliers were identified in the dataset. Therefore, there is no requirement to exclude any outliers from the dataset, affirming its suitability for analysis.

Fig. 2. Distribution of popularity

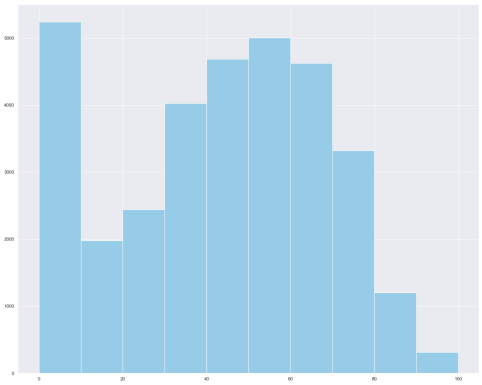
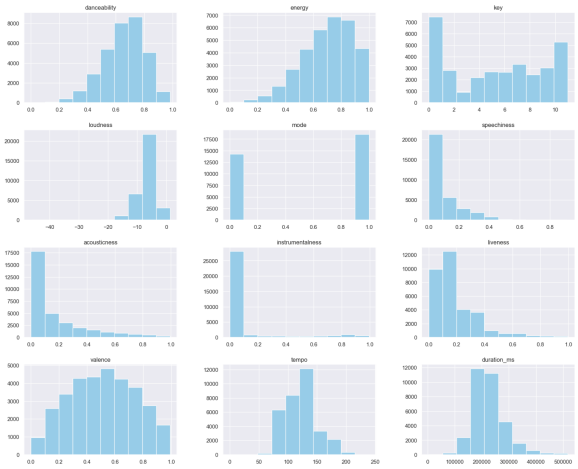


Fig. 3. Distribution of numerical features



For the categorical features, there is an intuition that the popularity of a song will be affected by the artist and the album. Therefore, we aggregated the data by the artist and album and plotted the average song popularity for each artist.

[1] https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs/data
[2] https://www.kaggle.com/

Figure 4 displays the average song popularity for the top 50 artists, while Figure 5 shows the average song popularity for the top 50 albums. Despite examining a subset of the dataset, it is worth noting that different artists and albums exhibit considerable variations in the average song popularity. This suggests that these factors play a significant role in determining the popularity of songs. Considering the importance of the artist and album features in our predictive model, we encountered a challenge due to a large number of unique artists (10692) and albums (19743). Traditional one-hot encoding for these categorical features would result in an unwieldy number of columns. To address this, we employed an aggregation approach. We performed a 'groupby' operation on the 'track_artist' and 'track_album_name' variables, calculating the mean 'track_popularity' for each artist and album. Subsequently, we merged this aggregated data back into the original dataset. As a result, we introduced two numerical features, 'mean_artist_popularity' and 'mean_album_popularity,' which will be more manageable and suitable for our subsequent analysis.

Fig. 4. Bar Chart for Average Track Popularity By Artist (50 data)

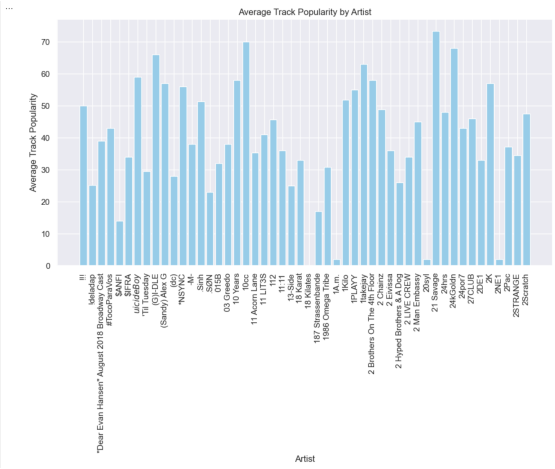
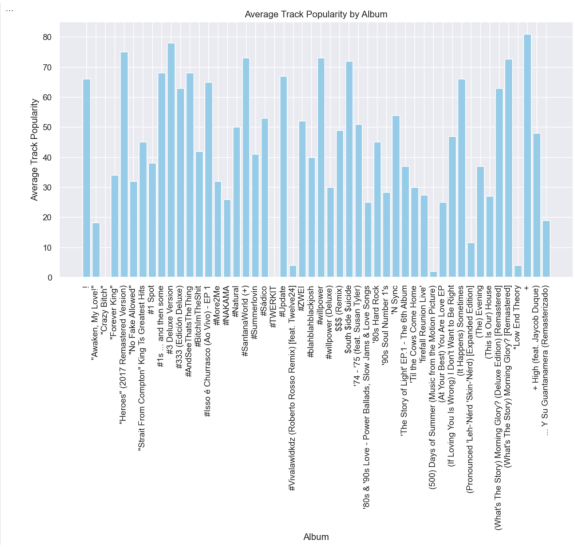


Fig. 5. Bar Chart for Average Track Popularity By Album (50 data)



2 PREDICTIVE TASK

Our primary objective in this dataset is to develop a predictive model for estimating a song's popularity score, denoted as "track_popularity." To evaluate our predictive task, we will employ regression metrics given the continuous nature of the popularity score. Common regression includes Mean Absolute Error(MAE) and Mean squared Error(MSE). These metrics will provide insights into the accuracy and precision of our model in predicting the popularity of songs. This predictive task holds significant utility as it can inform Spotify's marketing strategies, enabling more informed decision-making. Additionally, it can provide valuable insights to artists, empowering them to create music compositions that have a higher likelihood of becoming popular. The baseline model we created used a simple linear regression model. Based on our results from EDA, we decided to use all the numerical features for the model including: 'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', and 'duration_ms'. The mean absolute error for the baseline model is approximately 20.154. We chose the Mean Absolute Error (MAE) as our evaluation metric to assess model validity because MAE imposes a lighter penalty on outliers. Given the distribution of "track_popularity," which exhibits a heavy tail with many instances at popularity 0, MAE was chosen to accommodate and tolerate this tail effectively. To improve the baseline model, we include two categorical features: ‘playlist_genre’ and ‘playlist_subgenre’. To enhance the baseline model's performance, we incorporated two categorical features: 'playlist_genre' and 'playlist_subgenre.' To integrate these categorical variables into the model, we applied the one-hot encoding technique, converting them into binary numerical variables. The 'playlist_genre' has unique entries such as 'rock,' 'edm,' 'rap,' 'r&b,' 'pop,' and 'latin,' while 'playlist_subgenre' contains unique entries like 'hip hop,' 'electropop,' 'reggaeton,' and others. By one-hot encoding these features, we introduced an additional 30 binary columns. This improvement led to a reduction in Mean Absolute Error (MAE), with the updated baseline model achieving an MAE around 19.078.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right|$$

Formula 1, Mean Absolute Error

3 MODEL

To Improve the MAE of the model, we decided to change the model architecture and add additional features. Specifically, we changed the base model from simple Linear Regression to ensemble learning algorithms. Also, we thought the model would benefit from adding more features. We decided to engineer track_artist, playlist_genre, and the year of release and incorporate them into the final model.

3.1 Ensemble Tree

Choosing the most effective model architecture is essential for good performance. We switched from the baseline linear models to ensemble models because ensemble models can handle non-linearity more effectively and are more robust to outliers. To select the best-fitting model for our task, we compared the performance of 6 different kinds of models (AdaBoost, Random Forest, Gradient Boost, etc.) with various searchable parameters using sklearn randomized search and cross-validation. We found that AdaBoost with 100 estimators and a learning rate of 0.1 got the best result on the hold-out validation set. We will stick to this model architecture from now on. AdaBoost iteratively combines weak classifiers, assigning higher weights to misclassified samples in each iteration, enabling the model to focus on correcting previously misclassified instances and gradually improving overall classification performance.[3]

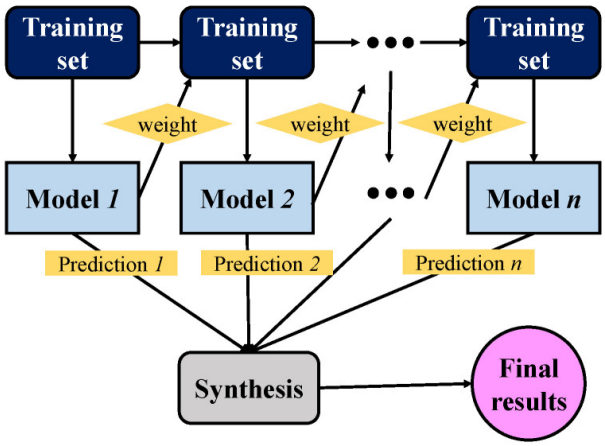


Fig.6 , AdaBoost algorithm calculation process.

$$F_n(x) = F_{m-1}(x) + argmin_h \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + h(x_i)\right)$$

Formula2, Ada calculation formula

Fn(x) represents the complete model at a given iteration, Fn-1(x) denotes the model obtained in the preceding round, yi signifies the prediction outcome of the i-th tree, and h(xi) represents the contribution of the newly introduced tree.

3.2 Feature Engineering

We conducted feature engineering on the dataset. Initially, we decided to pass the One-Hot encoded artist name directly into the model. However, we discovered that the model contains too many artists, and thus One-Hot encoding of the artist name attribute results in a huge sparse matrix that becomes infeasible for the ensemble algorithm to analyze and might also raise overfitting issues. So we devised a strategy to represent artist influence by computing the average popularity of each artist's tracks. This process involved aggregating 'track_popularity' for each 'track_artist,' creating a new feature termed 'mean_artist_popularity.' This feature offers a crucial advantage—it encapsulates the aggregated popularity trend associated with each artist. Through this engineering, we got a refined predictive model that better captures the interplay between artists and track popularity, reducing the MAE from 15.92 to 10.5. However, one caveat is that our model might learn a shortcut for prediction by simply outputting the 'mean_artist_popularity' feature. This problem is more severe for artists that publish few songs. In the extreme case, if some artists only released one song, then 'mean_artist_popularity' is nothing but the popularity of that particular song which the model tries to predict. Getting more training data for artists will help to mitigate this problem.

Besides representing artists influenced by their mean popularity, we also conducted PCA on the artist names. With 10,692 unique artist names, direct One-Hot Encoding of this categorical data would create an excessively large number of features, leading to computational challenges. Thus, we first used One-Hot Encoding to convert the artist names to a sparse matrix, then we applied Principal Component Analysis (PCA) via TruncatedSVD on the sparse matrix to mitigate dimensionality from around 10,000 to 10 while retaining critical information about the artist. The resulting model reduced MAE by 0.2 points. We also included the year of song release as an additional feature in the model since intuitively the time of release is correlated with the popularity and influence of the albums. We discovered that adding this time information reduced MAE by 0.15.

3.3 Evaluation

The evaluation of our final model reveals promising predictive capabilities for track popularity. Leveraging a combination of numeric and categorical features from the test set, the model accurately predicted track popularity, yielding a Mean Absolute Error (MAE) of 9.37. Considering the scale and context of the predictions, such an MAE value is indicative of the model's ability to make reasonably accurate predictions regarding track popularity. It underscores the model's capacity to capture underlying patterns and trends within the dataset, showcasing its potential utility in real-world applications where understanding and forecasting track popularity is valuable.

4 LITERATURE

Another research endeavor, based on the "30000 Spotify Songs" dataset from Kaggle, is centered on predicting song popularity through regression analysis. This study employs RandomizedSearchCV to explore various regression models, including Random Forest regressor and Gradient Boosting Regressor. Evaluation of these models is conducted based on the mean absolute error (MAE) on the test set, and the chosen model's feature importance is depicted through a bar plot. This methodology incorporates a hyperparameter grid, employing an iterative process to identify the most effective model from

[3] Wang, Changbai et al. "Adaboost Algorithm in Artificial Intelligence for Optimizing the IRI Prediction Accuracy of Asphalt Concrete Pavement." Sensors (Basel, Switzerland) vol. 21,17 5682. 24 Aug. 2021, doi:10.3390/s21175682

the specified options. In contrast, our approach adopts a pipeline strategy, integrating Linear Regression and Gradient Boosting Regressor within a preprocessing pipeline. The initial phase focuses solely on numeric features, while the subsequent stage incorporates both numeric and categorical features. Notably, our study introduces an innovative technique by incorporating artist mean popularity and leveraging PCA for artist information preprocessing. The assessment of model performance is consistent across both studies, relying on MAE as the key metric for predictive accuracy.

5 CONCLUSION

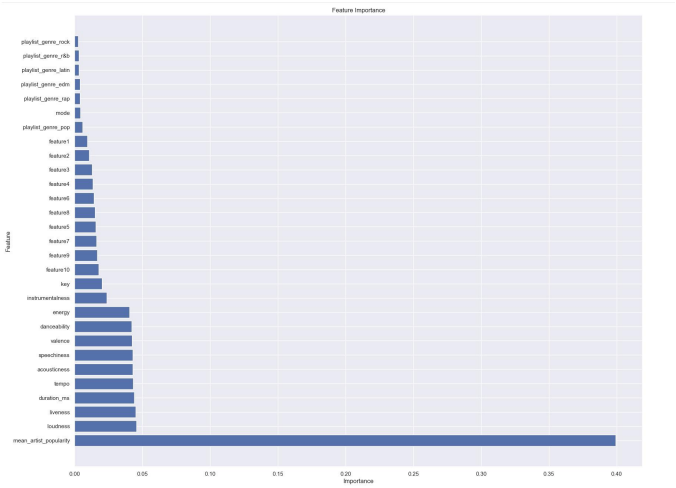
In our quest to identify the most effective regressor for predicting track popularity, we rigorously tested multiple ensemble learning techniques, including Random Forest Regressor, Gradient Boosting Regressor, ExtraTrees Regressor, Bagging Regressor and AdaBoost Regressor, each employing DecisionTreeRegressor as a base estimator. The selection process involved meticulous parameter tuning and evaluation using metrics such as Mean Absolute Error (MAE) to assess predictive performance. Among these diverse ensemble methods, we conclude that AdaBoost Regressor demonstrated superior efficacy compared to its counterparts and gave the lowest MAE.

Model	MAE
Baseline Linear Regression	20.154
Baseline Linear Regression (with one-hot-encoding)	19.078
GraidentBoost, adding mean popularity score feature	10.621
AdaBoost Regressor	9.375

AdaBoostRegressor model has a 'feature_importances_' attribute, which quantifies the importance of each feature in the model's decision-making process. We plot a horizontal bar chart (Figure 6) depicting the feature importance values, sorted in descending order, against their respective feature names. This visual representation offers a clear understanding of the impact and relevance of each feature in influencing the AdaBoostRegressor model's predictions regarding track popularity. We used Mean Absolute Error to evaluate our model. Mean Absolute Error on the test set: 9.372680665074145, which makes a considerable amount of improvement from the baseline model.

The feature importance bar chart generated from the AdaBoostRegressor model underscores the critical role played by 'mean_artist_popularity,' demonstrating the most substantial influence on the model's predictions with an approximate importance of 0.4%. This feature stands out as a pivotal contributor to our model, aligning with expectations regarding the significant impact an artist's popularity has on a track's overall reception. The prominence of 'mean_artist_popularity' reaffirms the notion that a song's popularity score is intricately linked to the renown and reach of the artist. Notably, the subsequent features—energy, danceability, valence, speechiness, acousticness, tempo, duration_ms, liveness, and loudness—display importance levels ranging between 0.03% to 0.05%. These numeric attributes are instrumental in defining various sonic qualities of songs, encompassing aspects like rhythm, acoustic properties, and intensity, contributing significantly to the predictive capabilities of the model. Furthermore, features beyond this subset exhibit importance levels of less than 0.025%, signifying their relatively lower influence on the model's predictions. This delineation of feature importance aids in identifying key predictors influencing track popularity while emphasizing the pivotal role of both artist-related attributes and nuanced musical qualities in our predictive framework.

Fig. 7. Bar Chart for Feature Importance



[1] Freund, Y., Schapire, R.E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, P. (eds) Computational Learning Theory. EuroCOLT 1995. Lecture Notes in Computer Science, vol 904. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-59119-2_166

[2] Friedman, Jerome H. "Greedy Function Approximation: A Gradient Boosting Machine." The Annals of Statistics, vol. 29, no. 5, 2001, pp. 1189–232. JSTOR, http://www.jstor.org/stable/2699986.

[3] Khan, F.; Tarimer, I.; Alwageed, H.S.; Karadağ, B.C.; Fayaz, M.; Abdusalomov, A.B.; Cho, Y.-I. Effect of Feature Selection on the Accuracy of Music Popularity Classification Using Machine Learning Algorithms. Electronics 2022, 11, 3518. https://doi.org/10.3390/electronics11213518

[4] Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324