Predicting song popularity on Spotify

Christopher Chung, Trang Ngo cc20, tqn1
Statistics

1. Introduction

Music is a huge part of many people's daily lives; the world collectively spent over 110 billion hours listening to music on Spotify in the year 2021 alone. But if everyone has a slightly different taste in music, what makes some songs more popular than others? Obviously more famous artists would generally lead to more clicks, but is there more to it? Do audiences prefer longer or shorter songs, or is there a sweet spot? Are newer songs more popular? What about tempo, loudness, liveness, and other attributes?

In order to gain more insight into these questions, we conducted statistical analysis on the Spotify datasets found at (Ay, 2022). One of the datasets, `artists.csv`, contains information about artists (their names and their popularity). The other dataset, `tracks.csv`, contains most songs on Spotify from 1921-2020, along with their popularity and other related attributes. In this work, we attempt to use these datasets to develop a model that predicts song popularity.

2. Methods

First, we cleaned the data to remove all rows with missing data. Then, we split the dataset into a training set and a testing set, where the training set size is 80% the full dataset size. Finally, we trained various statistical models on the training set before testing their performance on the test set. Given that the dataset associates every song with a popularity score between 0 and 100, we decided to pursue two types of analysis:

- Quantitative estimations (regression): We tried to predict the raw popularity number of each song.
- Categorical estimations (classification): We say a song is *not popular* if it has a popularity score below the median, which is 41; otherwise, we say a song is *popular*. We tried to predict which of the two categories each song belongs to.

We also incorporated information about the artists' popularity (from the artist dataset, `artists.csv`) into our dataset.

Note that the method we used to calculate overall artist popularity was a significant decision in this work. More specifically, if two or more artists worked on the same song, what should we use as the representative artist popularity? After some discussion, we decided that if p^* is the popularity of the main artist of the song, and $p_1, p_2, ..., p_n$ are the popularity scores of the featured artists, then the overall artist popularity P is calculated by:

$$P = \frac{1}{2}p^* + \frac{1}{2}\max\{p_1, p_2, ..., p_n\}$$

After all the pre-processing described above, the dataset we used had $p = 15$ predictors, and the training/testing set had 446940 and 111735 entries respectively. The predictors are: `explicit`, `danceability`, `energy`, `loudness`, `mode`, `speechiness`, `acousticness`, `instrumentalness`, `liveness`, `valence`, `tempo`, `time_signature`, `artist_popularity`, `duration`, and `year`.

3. Results

3.1 Principal component analysis

We started examining this dataset through unsupervised learning. Principal component analysis (PCA) is a method used to reduce the dimension of a given set of predictors. Given that our dataset contained ∼550k rows, $p = 15$ means dimension reduction probably is not too necessary; still, we wanted to observe whether there were clustering behavior. Since the dataset is too large to visualize the PC loadings, we plotted a biplot of the first two PCs using a random sample (including 2250 tracks) of the full dataset (Figure 1).

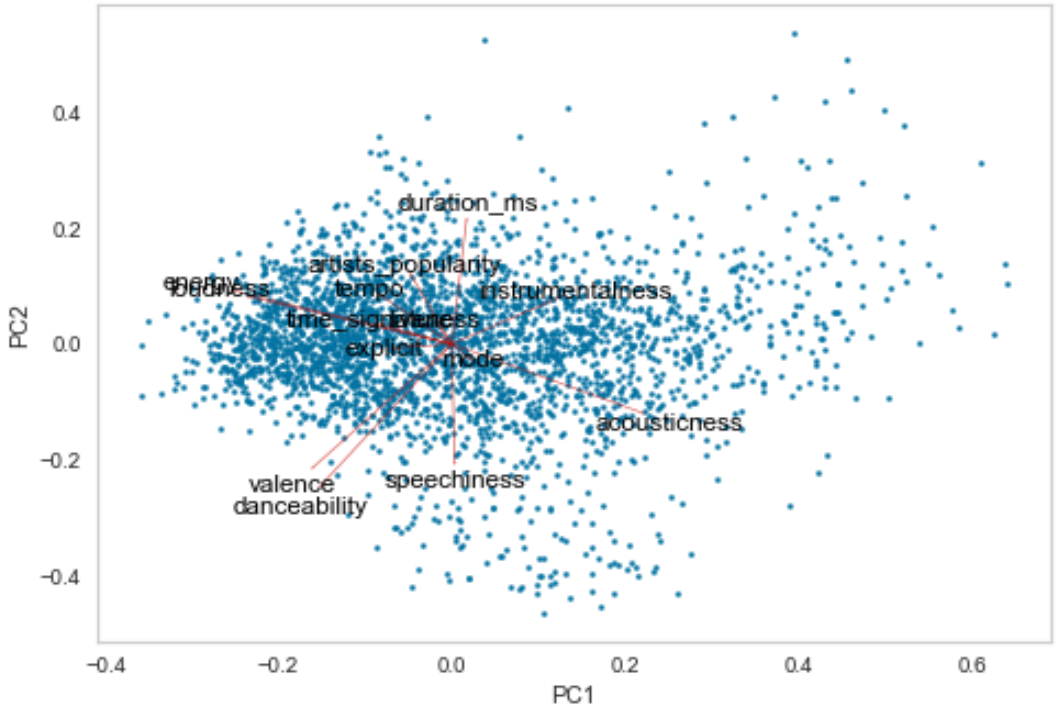

Figure 1: PC biplot of a random sample of Spotify dataset

The first two PCs are somewhat interpretable. The first PC is mostly correlated with loudness, tempo, energy and danceability, while the second PC is correlated with the songs' duration and speechiness. For instance, a song with low PC1 and PC2 might be a fast and energized rap song.

Although the first two PCs are fairly interpretable, we did not notice any clear clustering behavior from the PC biplot.

3.2 Regression

Table 1 lists the performances of various regression models on the test set.

Regression method	MSE (mean squared error)
Linear regression	142.23
Backward stepwise selection	142.23
L_1 regularization (Lasso)	142.22
L_2 regularization (Ridge)	142.22
Random forest	102.98
Smoothing splines	110.47
KNN (K-nearest neighbors)	167.30
Neural Network	116.55

Table 1: List of MSEs for regression models

We will now list some specific methods used to fit these models, and the corresponding observations we made.

- Stepwise selection did not remove any predictors. Both AIC (Akaike information criterion) and BIC (Bayesian information criterion) selected the entire set of predictors ($p = 15$) to keep.
- After conducting 3-fold cross-validation, we found that L_1 and L_2 regularization wasn't helpful on this dataset. This is surprising as Lasso generally works well in practice.
- Random forest was the best model among all models in regression. Using OOB (out-of-bag errors) as criteria, we first tuned the number of trees to be 200, then chose the number of features considered to be 9 (out of 15), and finally selected the minimum number of samples at the leaf to be 4.
- After fitting a final random forest regressor with these parameters, we found that `year` (release year of the song) and `artist_popularity` accounted for 0.66 of feature importance (0.38 and 0.28 respectively). No other variable had an importance over 0.07.

- Smoothing splines was the second best model; for this model we simply conducted cross-validation on the degrees of freedom of all predictors.

- K-nearest neighbors (KNN) did not work well on this dataset. We used 3-fold cross-validation to find the optimal number of neighbors (11), then made predictions based on this model.

- Due to the size of this dataset, fitting neural networks was somewhat time-consuming ($\sim$ 8 minutes per model). Still, we tested NNs with either 1, 2, or 3 hidden layers, with varying degrees of regularization and number of nodes. Our best model had 2 hidden layers with 12 nodes each.

3.3 Interpretation

In this section, we briefly interpret our results based on the previous random forest regression model. More specifically, how can we estimate the magnitude of effect that any predictor X (between 0 and 1) has on the overall song popularity?

To answer this question, we created two copies of the entire dataset, D_0 and D_1. Then, we set X to be all zeros in D_0 and all ones in D_1. Finally, we estimated the effect of X on the response variable by calculating $\hat{f}(D_1) - \hat{f}(D_0)$. A list of relevant (predictor, effect) pairs are listed in Table 2.

Predictor	Effect
Explicit (1 = explicit content)	0.95
Danceability	2.49
Energy	-1.16
Loudness	0.00
Mode (1 = major key)	0.09
Speechiness	-1.87
Acousticness	-2.41
Instrumentalness	-1.00
Liveness	-2.13
Valence	0.69

Table 2: Predictors and their effect

Notably, this calculated effect should not be interpreted causally. Still, it provides us some elementary ideas about what elements could contribute towards song popularity. In particular, if the goal is to create a popular hit, an explicit song with high danceability and low energy, speechiness, acoustics, liveness might be the way to go.

3.4 Classification

Now, we move on to classification. Table 3 lists the classification accuracy of various models on the test set.

Classification method	Accuracy
Logistic regression	78.38%
Logistic regression + BIC stepwise selection	78.67%
Logistic regression + L_1 regularization	78.67%
Logistic regression + L_2 regularization	78.47%
LDA (Linear discriminant analysis)	78.12%
QDA (Quadratic discriminant analysis)	74.99%
Random forest	81.91%
KNN	74.92%
NN	80.34%

Table 3: List of accuracies for classification models

All classifiers performed about the same, similar to the regression models; the best model was still random forest, with neural network being the second best. Some details worth noting are:

- The predictor `valence` was removed both from L_1 regularization and stepwise selection using BIC; this was not the case in regression.
- Since we used the median popularity as the threshold, we fitted both LDA and QDA with equal probability priors.
- KNN selected 18 neighbors instead of 11 (as in the regression case). However, its performance was still subpar compared to other models.

The receiver operating characteristic (ROC) curve is a common way to represent the false positive and false negative rates for any classification model. In addition, the total area under the ROC curve (AUC) is used as an overall score for the model. The plot of all ROC curves is given in Figure 2. As expected, we can see that the models with higher accuracy also have larger AUC.

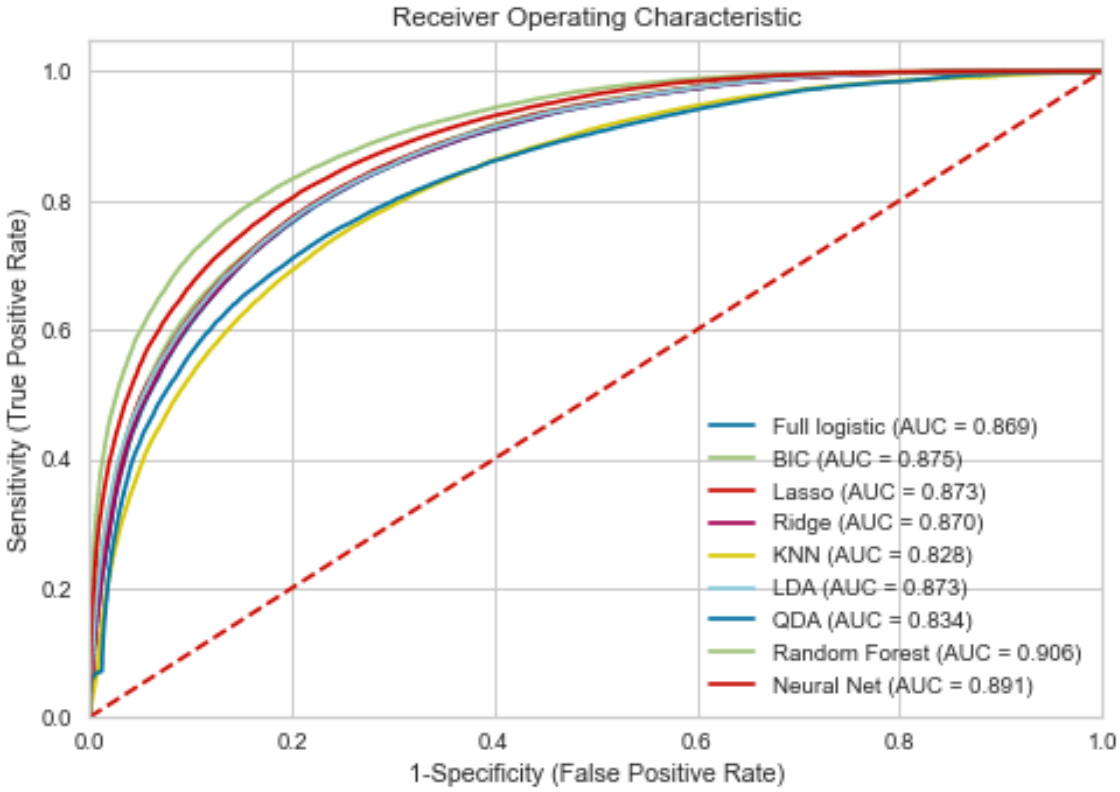

Figure 2: ROC plot of all classification models

Finally, we tried out the fancy stacked classifier. The key idea behind stacked classifiers is that we fit multiple models simultaneously; and by doing so, we hope to combine their advantages. For example, suppose we use two estimators f_1, f_2 and a meta-classifier F. Then, for any given data point x, the stacked classifier would return $F(f_1(x), f_2(x))$. In this work, we considered the case where F is a logistic regression.

Stacked classifiers take extremely long to fit due to the model complexity. In this particular dataset, the fitting time for a single model was around 25 minutes. As a result, we only tried out two models as listed in Table 4.

Classification method	Accuracy
Random forest + LDA	81.94%
Random forest + Neural Network	81.97%

Table 4: List of accuracies for stacked classifiers

Note that in the above stacked classifiers, the parameters for the individual estimators were not tuned due to extreme fitting time. As a result, we simply chose the optimal parameters from previous models and slightly adjusted for overfitting. Even without tuning, we could see that stacked classifiers were indeed able to out-perform their original respective models.

4. Discussion

Across both regression and classification models, random forests seem to perform the best overall. However, even random forests did not seem to significantly out-perform the others. In general, the performances of all models are not great, which suggests that the dataset is hard to predict. We have some speculations as to why this might be the case:

- The predictors are not very co-linear, which makes dimension reduction and reducing models' complexity difficult.
- The popularity of songs are highly variable; that is, the intrinsic variance of song popularity is quite large.
- There could be key attributes that this dataset does not contain. For example, it might be that the lyrics of a song are very predictive of the song's popularity, and we are missing out on crucial information.

References

Yamac Eren Ay. Spotify dataset 1921-2020, 600k+ tracks, Mar 2022. URL https://www.kaggle.com/datasets/yamaerenay/spotify-dataset-19212020-600k-tracks.