Predicting Song Popularity Using Machine Learning Algorithms

Bihag Dave*, Koushiki Bhattacharyya†, Prayag Savsani‡ and Yashraj Kakkad§
Ahmedabad University
Enrollment No: *AU1949007, †AU2029005, ‡AU1841035, §AU1841036

***Abstract*—Predicting popularity of a song would be of great industrial importance. We apply machine learning algorithms to metadata of a song for the same. We use data obtained from Spotify Web API, perform pre-processing and test several regression and classification algorithms. Models are tuned to give optimal test results.**

***Index Terms*—machine learning, music, popularity, prediction, songs, regression, classification**

I. Introduction

Hundreds of songs are published every year, but very few of them make it to the top charts. Music analysis has made it possible to generate metadata of a song. Similarly, such features are available for artists and genres. We aim to answer the question - "Is it possible to predict the popularity of a song using these attributes in Machine Learning algorithms?". Along with that, we learn more about the features involved, how important (or unimportant) they are and what are the possible limitations.

Our key contributions are the following -

1) We explore a suitable dataset which contains the requisite features and tune it to our needs.
2) We perform Exploratory Data Analysis (EDA) on the data.
3) We apply several machine learning algorithms to predict the popularity of a song. We use both regression and classification algorithms.
4) The model hyperparameters are tuned to fetch the best possible results. Different models are compared and inferences gathered.

II. Literature Survey

The ability to predict whether a song will be a hit or not is of commercial importance, due to which there has been a keen interest in using Machine Learning techniques in predicting the popularity of a song. This is often called Hit Song Science, a term coined by Mike McCready. There are several papers that quantify certain lyrical and acoustical characteristics of a song and use those to predict popularity. An early attempt in using such features is by B. Logan et. al [1]. Another paper by R. Nijkamp [2] uses the same Spotify dataset that this project aims to use and uses the attributes provided by Spotify as features to build a linear regression model.

There have also been attempts to predict song popularity from just lyrical properties. One such attempt is by A. Shinghi et. al [3] where they use rhyme, syllable and meter features to predict popularity. They found that lyrical features worked better than audio features in popularity predictions. Another approach to hit song prediction is to ignore intrinsic characteristics of a song altogether (lyrical or acoustical) and use social media metrics of the artist or buzz created by users. Such attempts were made as early as 2008 by Bischoff et. al [4]. In a similar vein E. Zangerle et. al investigates correlations between twitter hashtag #nowplaying and billboard top 100 charts [5].

III. Implementation

A. Introduction to Dataset

Spotify has a public API which serves information about tracks, albums, artists, genres etc. [6] We obtained our dataset from Kaggle which is a collection of information of more than 175,000 songs collected from Spotify Web API. [7] Features such as acousticness, danceability, energy etc. are available for tracks, artists, genres and years.

B. Problem Statement - Song Popularity Prediction

Our aim is to build and test models which can predict the song popularity score (regression) and whether or not the song is popular (classification).

C. Pre-processing Steps

We apply the following pre-processing steps to our data, to suit our analysis better -

1) We remove the features not useful for our analysis such as 'id', 'duration' etc.
2) We convert the categorical variables 'key' and 'mode' using One-Hot Encoding to make them more suitable for machine learning algorithms.
3) Track and artist tables are (left-outer) joined over artist, whose aggregation operations are performed as summarized in Table I. Similarly, track and year tables are also joined.

As a result, we obtain 72 columns which can be used as features.

TABLE I: Aggregate Operations for Joining Track Artist Data

Feature	Operation	Feature	Operation
Acousticness	Mean	Danceability	Mean
Energy	Max	Instrumentalness	Max
Liveness	Max	Loudness	Mean
Speechiness	Mean	Tempo	Mean
Valence	Mean	Popularity	Max
Count	Max	Mode	Max

D. Exploratory Data Analysis

We analyzed the data to obtain - correlation matrix, strongly correlated feature pairs, boxplots and histograms.

E. Regression algorithms

1) Linear Regression and Polynomial Regression: Performing iterative feature selection based on correlation with popularity, we noticed a plateau at around 27 features after which the co-efficient of determination did not increase appreciably and sacrificed computational time.

For polynomial regression, we went up to degrees 2 and 3, beyond which we were computationally limited, owing to the size of the dataset.

2) Lasso Regression: Lasso Regression has the property of acting has a feature selector. [8] It reduced all our feature weights to zero except for two - artist popularity and year popularity, while giving a reasonable accuracy.

3) Decision Tree Regression: We consider only binary trees for Decision Tree algorithms. The algorithm leads to severe overfitting (99% accuracy on cross validation vs 75% accuracy on test set), and therefore we tune the model hyperparameters, whose optimum values are summarized in Table II.

TABLE II: Hyperparameters of Decision Tree Algorithms

Hyperparameter	Regression	Classification
Maximum depth	10	8
Minimum samples to Split	6	8
Minimum samples required at leaf	7	2

F. Classification algorithms

The popularity values between 0-100 is converted to labels based on a decided threshold value. For all the classifiers given below, we have used 50 for the same which is slighty higher than the 3rd inter-quartile range. The positive class is in minority in comparison to the negative class i.e. the non-popular class. Therefore, we used stratified Kfolds technique instead of normal Kfolds during splitting our dataset and also during cross validation.

In this case, classification accuracy alone cannot be used as a reliable performance metric [8]. Therefore, we use other metrics like precision, recall and F1-score. More on this will be explained in the results section.

1) Support Vector Machines (SVM): During our experimentation, we observed that a non-linear SVM classifier gives better results. We applied the so-called *kernel trick* which is just exploiting math to map the data to a higher dimensional space. We use the Gaussian RBF kernel during training. Such a kernel has various regularization hyperparameters such as γ and C, which were tuned using grid search technique to give us the best possible F1-score.

2) Decision Tree Classification: We used "Entropy" impurity (versus the conventional "Gini") because it was giving slightly better results. The model hyperparameters and their optimum values are summarized in Table II.

IV. Results

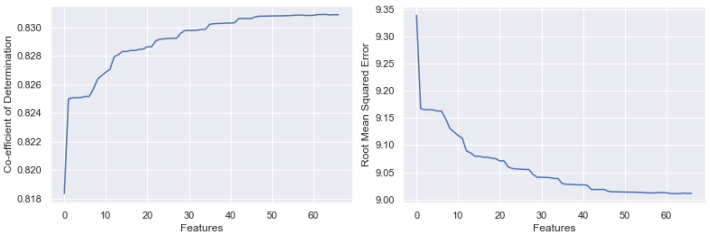
We used Scikit-learn, a popular machine learning library in Python [9], for the required tools.

A. Regression algorithms

TABLE III: Linear Models

Regression	Metrics			
	$R2_{Train}$	$R2_{Test}$	$RMSE_{Train}$	$RMSE_{Test}$
Linear Regression	0.82907	0.82920	9.03576	9.05644
Polynomial Regression	0.84282	0.84208	8.66458	8.70819
LASSO Regression	0.79677	0.83086	8.98626	9.01212
Decision Tree	0.86986	0.85649	7.88407	8.30137

Decision Tree yields very competitive results compared to the simpler algorithms. In Linear Regression, the results don't improve significantly with increased number of features.



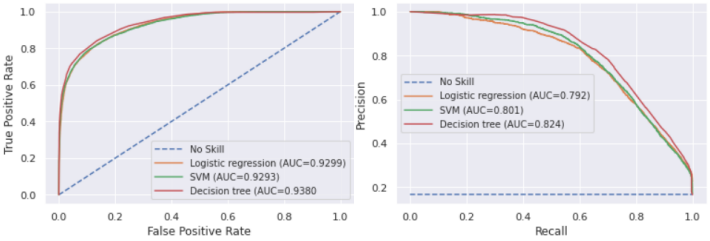
(a) R2 score vs. Number of features (b) RMSE vs. Number of features

TABLE IV: Performance metrics of classifiers

Classifier	Metrics				
	Train accuracy	Test accuracy	Precision	Recall	F1 score
Logistic Regression	0.9105	0.9110	0.79	0.64	0.70
SVM	0.9144	0.9142	0.86	0.58	0.69
Decision tree	0.9198	0.9211	0.83	0.65	0.73

B. Classification algorithms

As discussed before, it can be seen that the accuracy is not a suitable metric for this problem. Our recall scores may have some scope for improvement. Number of false negatives is considerably high.



(a) ROC curve (b) Precision-Recall curve

We plot the standard ROC curve and the Precision-Recall curve, the latter being more suited given our imbalanced class distribution. Here, the Decision Tree Classifier's curve is closest to the perfect curve which agrees with the metrics shown in Table IV.

V. Conclusion

We conclude that popularity can be predicted reasonably well using machine learning techniques. With further feature engineering, we aim to improve performance of current models. For further work, we plan to optimize current implementation. We intend to leverage domain knowledge to improve our performance metrics.

References

[1] R. Dhanaraj and B. Logan, "Automatic prediction of hit songs," pp. 488–491, 2005.

[2] R. Nijkamp, "Prediction of product success: explaining song popularity by audio features from spotify data," in *11th IBA Bachelor Thesis Conference*, 2018.

[3] A. Singhi and D. Brown, "Can song lyrics predict hits?" 2015.

[4] K. Bischoff, C. S. Firan, M. Georgescu, W. Nejdl, and R. Paiu, "Social knowledge-driven music hit prediction," in *Proceedings of the 5th International Conference on Advanced Data Mining and Applications*, ser. ADMA '09. Berlin, Heidelberg: Springer-Verlag, 2009, p. 43–54.

[5] E. Zangerle, M. Pichl, B. Hupfauf, and G. Specht, "Can microblogs predict music charts? an analysis of the relationship between nowplaying tweets and music charts," in *Proceedings of the 17th International Society for Music Information Retrieval Conference 2016 (ISMIR 2016)*. ISMIR, 2016.

[6] "Web api reference." [Online]. Available: https://developer.spotify.com/documentation/web-api/reference/

[7] Y. E. Ay, "Spotify dataset 1921-2020, 160k tracks (version 10n," Jan 2021. [Online]. Available: https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks/version/10

[8] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.