Machine Learning Predictions of Absolute Song Success on Spotify

Arnav Agarwal

Abstract

This research presents a new machine learning approach for predicting song popularity, using linear, polynomial, decision tree, and random forest regression models on a dataset of 7412 "Top 20" Spotify tracks. Unlike previous studies using Spotify's dynamic popularity metric, this study focuses on historical success metrics - maximum playlist ranks and chart duration. The findings reemphasize the influence of artist popularity on song success. Although the models were not highly accurate, I offer a unique angle on predicting song success.

1 Introduction

Music is an ubiquitous part of society. It impacts cultural phenomena and the emotions of people worldwide. Understanding what makes songs successful could not only aid understanding of why popular music has the impact it does, but also assist us in shaping popular music in the future.

An important consideration is what constitutes popularity. I refer to the definition of song popularity in [1] - music "that is loved, well-known, admired, and enjoyed by the general public." A pervasive indicator of popularity according to this definition is chart rankings. Being the industry leader of the most common method of music consumption today, I consider Spotify's charts to be an authoritative example. [2]

Through their API, Spotify provides acoustic and non-acoustic features for all tracks, providing researchers with the opportunity to analyse popular songs using this data. As part of the content for the course "Applied Machine Learning" at the University of Edinburgh, I was provided with a dataset containing the acoustic features, Spotify URIs, "Top 200" playlist ranks and dates these ranks were achieved for 7412 unique tracks between 01/01/2017 and 31/05/2023. I attempted to use this data to create effective predictive models which can reveal which song features are the most important drivers of achieved popularity. This can assist artists and labels in creating popular music more successfully, and can improve user satisfaction. [3]

Previous studies rely on the Spotify APIs 'popularity' metric [4][1]. Spotify continually updates the 'popularity' score based on recent streams [5]. Instead of focusing on popularity at the present day, I chose to focus my research on songs' achieved popularity during their most popular period; the maximum playlist rank they achieve and the number of days they on the charts.

In this paper, I present linear, polynomial, decision tree and random forest regression models predicting song popularity using acoustic, non-acoustic and artist popularity features.

2 Literature Review

Song popularity prediction is a widely researched field, with various studies focusing specifically on Spotify API data.

Some attempt to solve the problem using classification and binned Spotify popularity scores. [4] [1] [6] [7][8]. I hypothesise that regression is a superior approach, as it allows popularity prediction to a finer granularity. These also only predict the present popularity of songs. For this purpose some also include the release date as a feature [7] [4] . I argue that such papers measure present popularity of songs, not their absolute historical success.

Here, I present a novel combination - regression models to predict rank and chart longevity - which I believe can offer better granularity and an understanding of songs' absolute success.

3 Data Collection and Processing

3.1 Data Collection

In the dataset provided to us, the data I used included songs' 'Echo Nest' acoustical features excluding 'liveness' [5], their ranking on the playlists, the dates at which these ranks were achieved, the nationality of the primary artist and the URI code associated with each track. I used the URI codes provided to obtain more features ourselves from the Spotify API - the songs' modes (expressed as an integer), 'livenesses', keys (as an integer) and whether or not they were explicit.

3.2 Feature Engineering

3.2.1 Artist Popularity

Previous papers highlight the importance of artist popularity in predicting song popularity. [7] [8] In order to find a time independent measure of artist popularity to use as a feature in my models, I measured the number of unique songs each track's artist had in the dataset and thereby artists' propensity to appear in "Top 200" playlists.

3.2.2 Feature Scaling

Before passing numerical features to the linear and polynomial regression models, I scaled them. This was in order to ensure faster convergence to the optimal parameters in their gradient descent algorithms [9]. To prevent assuming the features are normally distributed, I use Scikit-Learn's MinMaxScaler, which scales the data to the 0-1 range whilst preserving the original distribution of the data. [10]

3.2.3 Feature Encoding

For the linear and polynomial regression models, I one-hot encoded the categorical features - artist nationality and explicit tag. This is so that they could be passed to the model as part of the $\mathbf{x}$ vector described in section 5. For each categorical feature, this introduced N binary components to the vector, each indicating whether the song belonged to the N^{th} category or not.

3.3 Target Engineering

Using the dates associated with each rank for each song, as provided in the dataset, I calculated the number of days and number of weeks each song was present in playlists. The number of days measures the longevity of a song's popularity, which can be considered a metric of popularity. This was used as a target for the models and the number of weeks was used for exploratory data analysis in section 4.

I also calculated the maximum rank achieved by every song. This was measured as the chart position, which was used for exploratory data analysis, and as 200 minus the chart position, which was used for the models.

4 Exploratory Data Analysis

4.1 Null and Unknown Values

I found no null values in the final set of features and targets for all songs through the `isnull()` method in Pandas [11].

I found that some songs had an "Unknown" nationality. However, this constituted very few examples, so I decided to continue with these unknown values.

4.2 Correlation Heatmap

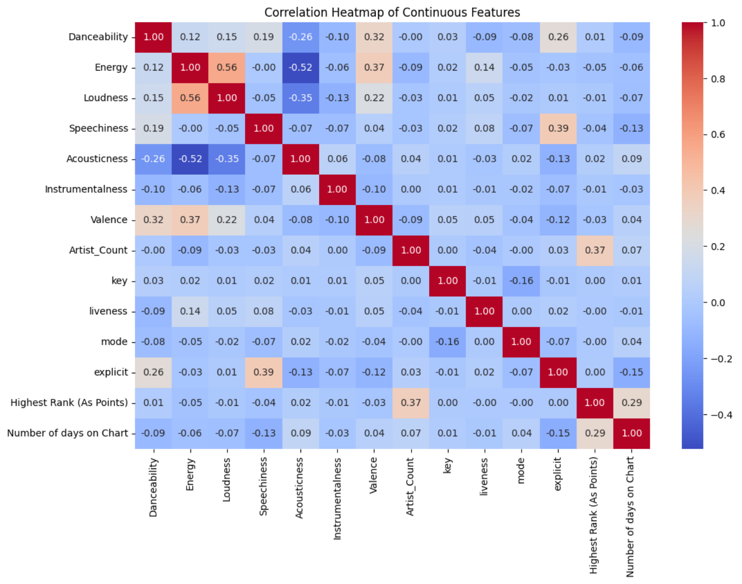


Figure 1: Matrix demonstrating the strength of linear correlations between numerical features, explicit tag, highest rank and number of days on charts.

The figure demonstrates intuitive relationships between certain features. For example, the positive correlations between loudness and energy, energy and valence, danceability and valence, loudness and valence, speechiness and explicit tag, as well as the negative correlations between acousticness and energy, acousticness and danceability, acousticness and loudness. This indicates that the features are representative of the music and the data is properly structured.

The interdependencies between features led us to hypothesise that tree-based models would perform better than others, as they will better capture listeners' genre-specific demands and better suit their reasons for listening. For example, consumers may be listening to relax, so may prefer lower energy music. It may be that lower energy music with higher acousticness is preferred over lower acousticness by listeners looking to relax. Tree-based models are able to capture the impact of such feature combinations

Furthermore, I can see a positive correlation between highest rank and days on chart, which indicated that my two popularity metrics are correlated. This matches domain knowledge; high ranking songs tend to stay on the charts longer.

4.3 Breakdown by Popularity Metric

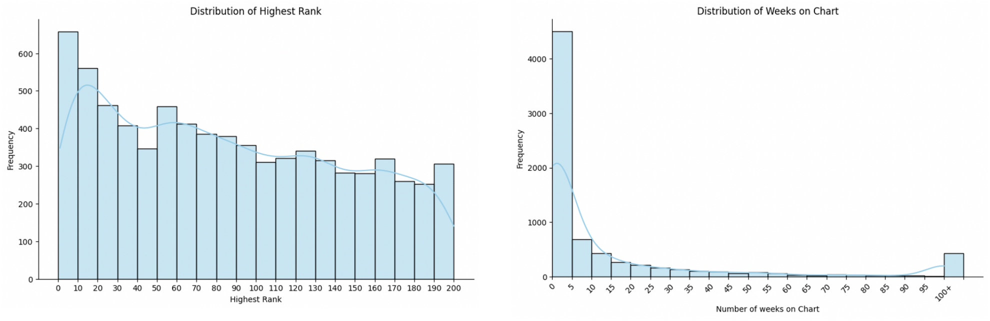


Figure 2: Frequency distribution of songs' highest ranks' (left) and weeks on charts (right).

The figure displays a breakdown of the songs for both of the popularity metrics. A downward trend is noticed as rank positioning increases, indicating that songs that enter the charts tend to reach higher positions rather than remaining at the lower ones. Furthermore, a downward trend is noted for the time spent on the charts, indicating that songs enter the charts for a brief "hit" period before becoming less popular. I could conclude that the two targets measure two different facets of popularity, popularity as rank measures reach during the popular phase, and longevity measures the "evergreen" nature of popular songs.

5 Models Used

5.1 Linear regression

Linear regression models a linear relationship between independent and dependent variables. Feature values x_D are assigned a coefficient w_d, which is found during training. The target values $f(\mathbf{x}, \omega)$ can be expressed as

$$f(\mathbf{x}, \omega) = \omega_0 + \omega_1 x_1 + \ldots + \omega_D x_D = \omega_0 + \sum_{d=1}^{D} \omega_d x_d = \omega^\top \phi(\mathbf{x})$$

where

$$\omega = [\omega_0, \omega_1, \ldots, \omega_D]^\top \quad \text{and} \quad \boldsymbol{\phi}(\mathbf{x}) = [1, x_1, \ldots, x_D]^\top.$$

There are no hyperparameters associated with linear regression, so cross-validation, as described in the next section does not apply.

5.2 Polynomial Regression

In polynomial regression, the relationship between independent and dependent variables is modelled as an n^{th} degree polynomial, allowing us to capture nonlinear superposition relationships. The targets are expressed as

$$f(x_i, \boldsymbol{\omega}) = \omega_0 + \omega_1 x_i + \omega_2 x_i^2 + \ldots + \omega_n x_i^n = \omega_0 + \sum_{d=1}^{n} \omega_d x_i^d = \boldsymbol{\omega}^\top \phi(x_i)$$

where

$$\boldsymbol{\omega} = [\omega_0, \omega_1, \ldots, \omega_n]^\top \quad \text{and} \quad \phi(x_i) = [1, x_i, x_i^2, \ldots, x_i^n]^\top.$$

Polynomial regression offers a single hyperparameter to be optimised, the order of the polynomial fit.

5.3 Decision Tree Regression

Decision trees recursively split the input feature space, dividing data based on input feature values. In regression trees, splits minimize the variance of the target variable in child nodes. Each node selects the feature and split point that most reduce target variance, aiming for more homogeneous target values in child nodes, enhancing predictive accuracy. The leaf nodes, or final child nodes, use the average target value of remaining training examples.

Decision tree regression requires four hyperparameters. The maximum tree depth limits the number of splits. The number of minimum samples required to split a node determines how many samples must be present in a node for it to be considered for a split. The minimum number of samples required in a leaf node specifies the smallest number of samples that a leaf node can have. Finally, the maximum number of features considered for splitting a node defines the subset of features to consider when looking for the best split. [12]

5.4 Random Forest Regression

Random forest regression facilitates the processing of large datasets by constructing multiple distinct decision trees during training and merging them together to output the average predicted value of a single tree. [13]

Random forest presents one additional hyperparameter to those required by a single decision tree - the number of individual trees to be averaged across. [14]

6 Training and Optimisation

6.1 Data Segmentation

I split the data randomly into 40% training and 60% test sets. The training set was used for hyperparameter optimisation and model training, the test set was used to evaluate generalisation error, as described in the following section. I chose this split to maximise the size of the training set whilst maintaining a low computational cost.

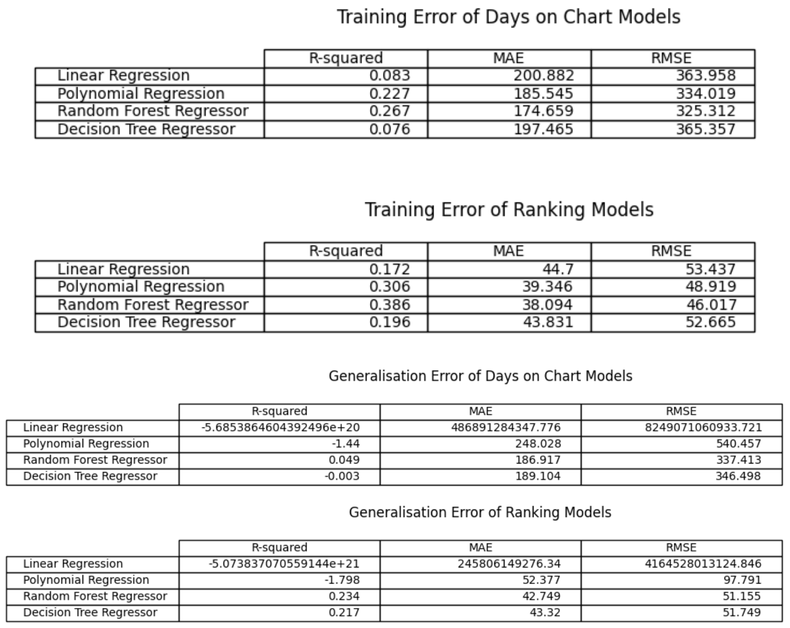
6.2 Optimisation

To optimize hyperparameters for each model, I used five-fold cross-validation with Scikit-Learn's `GridSearchCV`. This method splits the training set into five parts, using fmy for training and one for validation, rotating each subset for comprehensive evaluation. The `score` method of Scikit-Learn's models measures accuracy, which is averaged across folds for robustness. `GridSearchCV` tests all hyperparameter combinations within defined ranges. I tried various ranges until optimal values are found that aren't at the extremes of these ranges. This approach ensures finely-tuned hyperparameters for each model.

7 Results and Evaluation

7.1 Evaluation

Error on the training and testing sets was measured using coefficient of determination (R^2), mean absolute error and root mean squared error.



Training Error of Days on Chart Models

	R-squared	MAE	RMSE
Linear Regression	0.083	200.882	363.958
Polynomial Regression	0.227	185.545	334.019
Random Forest Regressor	0.267	174.659	325.312
Decision Tree Regressor	0.076	197.465	365.357

Training Error of Ranking Models

	R-squared	MAE	RMSE
Linear Regression	0.172	44.7	53.437
Polynomial Regression	0.306	39.346	48.919
Random Forest Regressor	0.386	38.094	46.017
Decision Tree Regressor	0.196	43.831	52.665

Generalisation Error of Days on Chart Models

	R-squared	MAE	RMSE
Linear Regression	-5.6853864604392496e+20	486891284347.776	8249071060933.721
Polynomial Regression	-1.44	248.028	540.457
Random Forest Regressor	0.049	186.917	337.413
Decision Tree Regressor	-0.003	189.104	346.498

Generalisation Error of Ranking Models

	R-squared	MAE	RMSE
Linear Regression	-5.073837070559144e+21	245806149276.34	4164528013124.846
Polynomial Regression	-1.798	52.377	97.791
Random Forest Regressor	0.234	42.749	51.155
Decision Tree Regressor	0.217	43.32	51.749

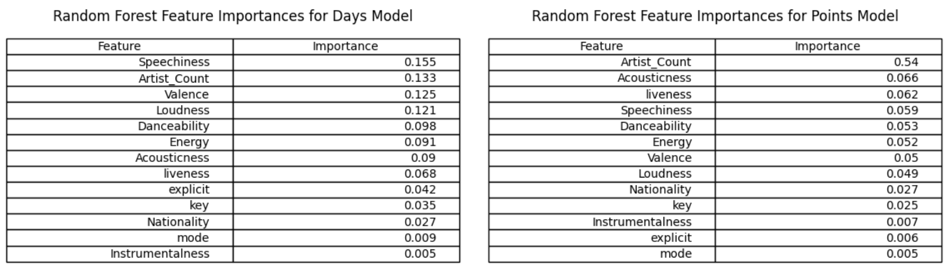
Figure 3: Error Metrics on Models

Performance across all models was overall poor, as shown in figure 3, with no coefficients of determination close to one. The random forest regressors were the best performing models across all metrics. In terms of generalisation error, the decision tree regressor was second best across all metrics. This aligns with my prediction from section 4.2 that tree-based models would perform best.

The best performing model overall was the Random Forest Regressor when predicting chart rankings, with a testing set coefficient of determination 0.234.

7.2 Feature Importances

The feature importances for both the days on chart and ranking Random Forest models is shown in figure 4



Random Forest Feature Importances for Days Model

Feature	Importance
Speechiness	0.155
Artist_Count	0.133
Valence	0.125
Loudness	0.121
Danceability	0.098
Energy	0.091
Acousticness	0.09
liveness	0.068
explicit	0.042
key	0.035
Nationality	0.027
mode	0.009
Instrumentalness	0.005

Random Forest Feature Importances for Points Model

Feature	Importance
Artist_Count	0.54
Acousticness	0.066
liveness	0.062
Speechiness	0.059
Danceability	0.053
Energy	0.052
Valence	0.05
Loudness	0.049
Nationality	0.027
key	0.025
Instrumentalness	0.007
explicit	0.006
mode	0.005

Figure 4: Feature importances output by the Random Forest Regressor for both models.

For the days on chart model, it seems that speechiness is the most important feature. This could be a valid conclusion; it aligns with domain experience that songs that stick around longest tend to be lyrical pop songs. However, due to the poor performance of this model, with a testing set coefficient of determination of 0.049, I cannot reasonably draw any conclusions.

The points model shows that the most important feature, by a large margin, is the number of songs the artist has had appear in the charts. This aligns with previous researchers' emphasis on the importance of artist popularity, as discussed in section 3.2.1.

8 Discussion

The performance of the models presented could have been significantly improved by employing feature selection. This would involve using either statistical, dimensionality reduction (such as principal component analysis), or feature importance based methods to determine which subset of features is most relevant. [15] As [4] states, "using excessive number of data dimensions can cause a model to interpret the data incorrectly by giving too much importance to not so relevant features." This would be the primary improvement to consider in a repeated study.

Furthermore, there were certain features which I neglected from the dataset provided due to time constraints and the additional complexity of including them. These included the number, nationalities and popularities of additional artists on tracks. Including these before conducting feature selection could yield better results.

Finally, it may that a bigger training set would have given better results. However due to time constraints and the lack of better computational resources, this was not feasible.

A possible alternative conclusion is that song features, when combined with artist popularity are not an effective predictor. However, this conclusion would be at odds with the consensus in the literature. I would conduct a study with the improvements suggested before considering this hypothesis.

An additional point I would like to note is that if I consider the purpose of such a paper to be a method of predicting the popularity of *any* song, using the Spotify "Top 200" dataset would not be a suitable choice. [4] highlights that training models on only songs which appear on ranking, I are training it to predict the popularity of a certain type of track, pop songs that make it to the top 200.

9 Conclusions

I aimed to explore absolute measures of song success through regression models, whose performance was poor. However, one key insight is the reinforcement of the importance of artist popularity for song popularity. I suggest incorporating refined feature selection, possibly additional features and better data segmentation in future research. I also suggest widening training data for such models to non-charting songs for broader applicability. Overall, the research provides a foundation for further exploration in the field of music popularity prediction.

References

[1] Harriman Samuel Saragih. Predicting song popularity based on spotify's audio features: insights from the indonesian streaming users. *Journal of Management Analytics*, 0(0):1–17, 2023. doi: 10.1080/23270012.2023.2239824. Available at: https://doi.org/10.1080/23270012.2023.2239824.

[2] Chris Cooke. Music consumption at all time high powered by streaming and video apps. URL `https://archive.completemusicupdate.com/article/music-consumption-at-all-time-high-powered-by-streaming-and-video-apps/`.

[3] Ahmet Cimen and Enis Kayi. A longitudinal model for song popularity prediction. In *In Proceedings of the 10th International Conference on Data Science, Technology and Applications (DATA 2021)*, pages 96–104, 2021. doi: 10.5220/0010607700960104.

[4] Lejla Vardo, Jana Jerkić, and Emir Žunić. Predicting song success: Understanding track features and predicting popularity using spotify data. In *2023 22nd International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6, 2023. doi: 10.1109/INFOTEH57020.2023.10094172.

[5] Spotify web api. URL `https://developer.spotify.com/documentation/web-api`.

[6] Joshua S. Gulmatico, Julie Ann B. Susa, Mon Arjay F. Malbog, Aimee Acoba, Marte D. Nipas, and Jennalyn N. Mindoro. Spotipred: A machine learning approach prediction of spotify music popularity by audio features. In *2022 Second International Conference on Power, Control and Computing Technologies (ICPC2T)*, pages 1–5, 2022. doi: 10.1109/ICPC2T53885.2022.9776765.

[7] Gao, Andrea. Catching the earworm: Understanding streaming music popularity using machine learning models. *E3S Web Conf.*, 253:03024, 2021. doi: 10.1051/e3sconf/202125303024. URL `https://doi.org/10.1051/e3sconf/202125303024`.

[8] Agha Raza and Krishnadas Nanath. Predicting a hit song with machine learning: Is there an apriori secret formula? 07 2020. doi: 10.1109/DATABIA50434.2020.9190613.

[9] Aniruddha Bhandari. Feature engineering: Scaling, normalization, and standardization, 2023. URL `https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/`.

[10] Sklearn.preprocessing.minmaxscaler. URL `https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html`.

[11] pandas.dataframe.isnull. URL `https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.isnull.html`.

[12] sklearn.tree.decisiontreeregressor, . URL `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html`.

[13] Rakesh Kanth. How does random forest work?, Feb 2023. URL `https://www.analyticsvidhya.com/blog/2023/02/how-does-random-forest-work/`.

[14] sklearn.ensemble.randomforestclassifier, . URL `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`.

[15] Aman Gupta. Feature selection techniques in machine learning, Apr 2023. URL `https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/`.

Appendix: Hyperparameters Used For Models

After trial and error with using various hyperparemeter grids in `GridSearchCV`, here were the final hyperparameters used in the models:

- Polynomial Regression (Both): 'degree': 2.
- Decision Tree (Days on Chart): 'max_depth': 7, 'max_features': 'log2', 'min_samples_leaf': 6, 'min_samples_split': 16
- Decision Tree (Ranking): {'max_depth': 3, 'max_features': 1.0, 'min_samples_leaf': 6, 'min_samples_split': 2}
- Random Forest (Days on Chart): {'max_depth': 10, 'max_features': 1.0, 'min_samples_leaf': 15, 'min_samples_split': 15, 'n_estimators': 400, 'n_jobs': -1}
- Random Forest (Ranking): {'max_depth': 8, 'max_features': 1.0, 'min_samples_leaf': 10, 'min_samples_split': 10, 'n_estimators': 350, 'n_jobs': -1}