Hot or Not: Predicting Song Popularity with Supervised Classification Algorithms

Tiffany Moran
210513435
ECS784P Data Analytics
MSc Computing and Information Systems

Abstract—**What really makes a song a hit? This project investigates if by analysing the One Million Song Dataset [1], extracting various audio analysis features, song features, and metadata features, and applying supervised learning methods to them, we can better understand and predict what makes a song popular.**

Keywords—music, hit song prediction, classification, logistic regression, K-nearest neighbour, supervised learning

I. Introduction

How does one define what makes a song popular? Is it the lyrics, the melody, or are more ambiguous external factors involved? Through this report we further investigate whether there are certain characteristics that make a hit song, and if old hit songs can predict the popularity of new songs. To analyse the popularity of music, we look at Columbia's One Million Song Dataset [1] to find links between *hotttnesss* (the dataset's popularity metric) and various features including *tempo, artist_familiarity, loudness,* etc. Through machine learning methods including feature extraction and selection, logistic regression, and K-nearest neighbour classification, we can attempt to answer these questions and output a model that predicts whether a song will be popular or not.

The music industry, and more specifically the popular music industry, brings in billions of dollars in revenue each year, so there is money and real world application in algorithmically evaluating the potential popularity of a new song. Even by just determining the important features, it allows studios and musicians to hone in on a few key factors and invest in music that gives them a good return on their investment.

Objectives

- Identify heavily weighted features that contribute to a song's popularity, or hotness.
- Conduct exploratory analysis to find correlation between certain features in the One Million Song Dataset.
- Use a minimum of two classification/prediction models on song popularity to produce an outcome.
- Analyse the strengths and weaknesses of the models and evaluate both.
- Highlight possible improvements for future use and analysis.

II. Literature Review

The question of what determines a song's popularity has been posed and studied numerous times before, and four different works will be presented and dissected in this section.

The first report is titled 'Can Song Lyrics Predict Hits?' by *Abhishek Singhi* and *Daniel G. Brown* [2]. The aim of this report was to use standard machine-learning algorithms to analyse audio features, rhyme and syllable features, meter features, and lyrics features to define what is a hit and what is a flop. While they were successful in determining clever lyrics and high-quality songwriting, their determinisation of hit songs was only mildly successful. Using just lyrics features, they were able to identify about half of the hits, while misclassifying about 12.8% of flops as hits. However, they had more trouble in correlating rhyme and meter features with hit songs. The obvious drawback in this approach is that any cover or remake of a song cannot be predicted. It appears that there is merit to examining lyrical elements, but more auditory features are worth exploring as well.

Another report looked at the effect of vocals on an audience, rather than lyrics in "Vocals in Music Matter: The Relevance of Vocals on the Minds of Listeners" by *Andrew Demetriou, Andreas Jansson, Aparna Kumar,* and *Rachel M. Bittner* [3] and found it to be a salient aspect of music, ranking it second most important behind emotion. This also shows the importance of audio features and artistic influence.

In another popular music prediction research report "Experimental Study of Inequality and Unpredictability in an Artificial Cultural Market," by *Matthew J. Salganik, Peter Sheridan Dodds,* and *Duncan J. Watts* [4], the experimental study focused specifically on the affect social influence, or information about the choices of others, has on determining hit songs. In this experiment they created an artificial music market in which participants either listened to music independently or with a social influence condition. They determined that while the best songs never performed very poorly and the worst songs never performed incredibly well, the quality of a song played less of a role than social influence in predicting whether it becomes popular. Though this experiment is unlike actual cultural markets with significantly more variables at play, it does indicate the exploration of other features beyond the scope of just auditory are worth considering when attempting to predict chart toppers.

Similarly, *Noam Koenigstein* and *Yuval Shavitt* looked at different social influence features in their report "Song Ranking Based on Piracy in Peer-to-Peer Networks" which hypothesised the billboard chart was not reflective of popularity due to an increase in file sharing and torrenting [5]. However, they found that the billboard ranking is still relevant and in-tune with what people are listening to, but that other features like social media comments were indicative of popularity, confirming that not just auditory features are worth exploring.

All of this research is indicative of how difficult it can be to strike the right balance of features to include in a model, and we have to be holistic in our approach for song popularity specifically.

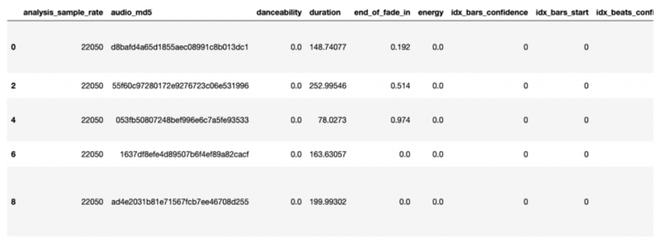
III. Data Management

A. *Dataset and Description*

The dataset used in this report was The Million Song Dataset, a collection of music data for one million popular music songs up until the year 2011, when the full dataset was published. Most of the feature analysis and metadata for the songs was provided by The Echo Nest, a music intelligence platform. A range of information is provided including audio features such as song title, key, and loudness, as well as more abstract features such as song hotttnesss, artist familiarity, and danceability. For this analysis only a subset of 10,000 songs was used due to the size and subsequent computational requirements of the full one million song dataset.

One challenge the dataset provided was that every song in the subset came in an individual h5 file. A script was provided to convert the h5 files into readable, working data but it was created for an older version of Python. Some modifications were made to update the script as well as have it produce a csv file that could be used to train our models via a data frame.

Table 1 Data Sample from original dataset



	analysis_sample_rate	audio_md5	danceability	duration	end_of_fade_in	energy	idx_bars_confidence	idx_bars_start	idx_beats_confi
0	22050	d8bafd4a65d1855aec08991c8b013dc1	0.0	148.74077	0.192	0.0	0	0	
2	22050	55f60c97280172e9276723c06e531996	0.0	252.99546	0.514	0.0	0	0	
4	22050	053fb50807248bef996e6c7a5fe93533	0.0	78.0273	0.974	0.0	0	0	
6	22050	1637df8efe4d89507b6f4ef89a82cacf	0.0	163.63057	0.0	0.0	0	0	
8	22050	ad4e2031b81e71567fcb7ee46708d255	0.0	199.99302	0.0	0.0	0	0	

5 rows × 53 columns

B. *Missing Data*

This database contains a total of 53 features ranging from audio analysis features to artist related features to song related features. In order to get a better idea of what values were missing, we used the df.info() and df.describe() functions to grasp the number of null values and see if any columns had a standard deviation of zero, or contained zero values where it did not make sense.

There were several cases of missing information in the form of null inputs or zeros that signified missing data. Large portions of *year, artist_latitude*, *artist_longitude*, and *location* values were missing from the data points. Calculating the mean, median, or mode for *year* or the location features did not serve helpful in this case, so we dropped the location columns and the missing year rows.

As for *song_hotttnesss*, there were many null values. There were also many rows containing zeros, which initially were interpreted as songs with no popularity, but after reading The Echo Nest documentation, it became clear these were unclassified and should be counted as null. So for these features, we dropped any songs with missing fields or a value of zero. This still left a solid 2,712 rows of data to analyse.

Other columns that we had hoped would provide interesting data points like *danceability* and *energy* had only been filled with 0.0 data points for every row, rendering them ineffectual, and so these were dropped as well.

C. *Feature Selection*

After deciding to drop rows with large gaps of important missing data points, the data needed to be further cleansed by narrowing down columns.

Any fields that related to an id, including *song_id, track_id*, or *artist_id* were dropped from the dataset as they merely provided identification rather than information. Song, album, and artist name information was also dropped. At the end of this process we were left with 18 columns and 2,712 rows of data.

```
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   analysis_sample_rate  2712 non-null   int64
 1   artist_familiarity    2712 non-null   float64
 2   artist_hotttnesss     2712 non-null   float64
 3   artist_terms          2712 non-null   int64
 4   bars_start            2712 non-null   int64
 5   beats_start           2712 non-null   int64
 6   duration              2712 non-null   float64
 7   end_of_fade_in        2712 non-null   float64
 8   key                   2712 non-null   int64
 9   loudness              2712 non-null   float64
 10  mode                  2712 non-null   int64
 11  mode_confidence       2712 non-null   float64
 12  segments_start        2712 non-null   int64
 13  song_hotttnesss       2712 non-null   float64
 14  start_of_fade_out     2712 non-null   float64
 15  tempo                 2712 non-null   float64
 16  time_signature        2712 non-null   int64
 17  year                  2712 non-null   float64
```

D. Measuring Popularity (song_hotttnesss)

The target variable of popularity is labeled in the data set as *song_hotttnesss* and had been predefined by Echo Nest as a score between 0 and 1.

For simplicity sake to answer our question, we decided to create a new column of data with binary bin labels called *is_hot*. Given the hotness scale is from 0 to 1, we labelled all songs with a hotttnesss score below 0.5 as not hot, and all songs with a hotttness score above 0.5 as hot. *is_hot* became the target variable, and it also made data exploration much easier by reducing the unique values for hotttnesss to two.

E. External Libraries

NumPy: A library including support for mathematical operations on large, multi-dimensional arrays and matrices.
Pandas: A data focused library used for data manipulation and analysis. Allows for the use of the DataFrame data structure.
Seaborn and *Matplotlib:* Two data visualization libraries that allow for data exploration in the form of attractive and informative statistical graphs.
Scikit-learn: A machine learning library that is widely used. It features various supervised and unsupervised learning algorithms .

IV. Methodology

For the given dataset, it was appropriate to choose supervised machine learning methods over unsupervised machine learning methods. This is because we have labelled data with input and output variables given. The target variable, *is_hot* is being derived from previous song popularity.

The two models used in predicting song popularity are logistic regression and K-nearest neighbour, which are used for classification.

A. Logistic Regression Algorithm

Logistic regression is a classification machine learning algorithm which can be used for predicting categorical values. Logistic regression learns the sigmoid of a function *f(x)* in order to predict, in our case, song popularity given new data *x*. Sigmoid is used to map predictions to probabilities by mapping values into another value between 0 and 1. In this example we have two classes, hot or not hot (1–hot and 0–not hot).

This algorithm is centered around probability, and a value *P(x)* is given as a result of the logistic function that an instance belongs to a category in the prediction space. The probability of that same instance belonging in the opposite category is given the label *1-P(x)*.

B. K-Nearest Neighbour Algorithm

K-nearest neighbor (KNN) is also a supervised learning model that can be used for classification. The goal of a KNN algorithm is to use historical data to predict, in this case, song popularity given new data *x*. It does this by calculating the distance between the test data and the training points, determining the *K* number of points closest to the test data, calculating the probability of the test data belonging to the correct class, and picking the class with the highest probability.

Unlike logistic regression, KNN does not require model training, because it is instance based.

C. Justification

Though logistic regression and KNN are both classification algorithms, they are different in their approach of categorising. We considered using support vector classification, but the results produced are very similar to those of logistic regression, and we wanted to compare and contrast different methods to pick the best one.

Logistic regression is best used to predict categorical outcomes such as pass(1)/fail(0) or healthy(1)/sick(0), so it made sense to use it in this case to predict hot(1)/not hot(0). As this was a binary classification problem and a probability problem, this model was appropriate to use and easy to implement.

KNN is one of the most easy machine learning techniques to implement, as it is a lazy learning model. Unlike logistic regression, it is a non-parametric method that supports non-linear solutions. In case music popularity prediction is a problem of non-linear classification, we wanted to explore other classification methods to create the most accurate model.

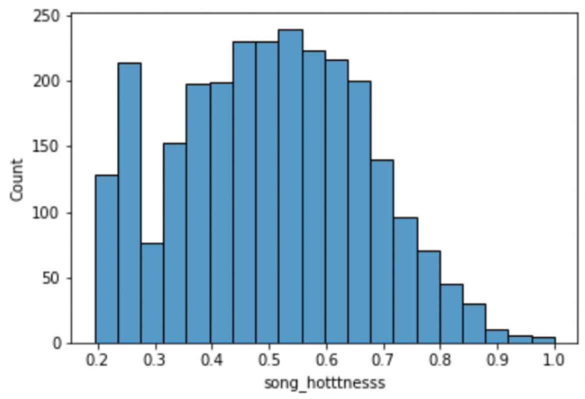
V. Data Exploration

After reading through the dataset documentation and cleaning the data to be used, we moved on to data exploration and looking into trends relating to the target variable and determining feature importance.

A. Trends

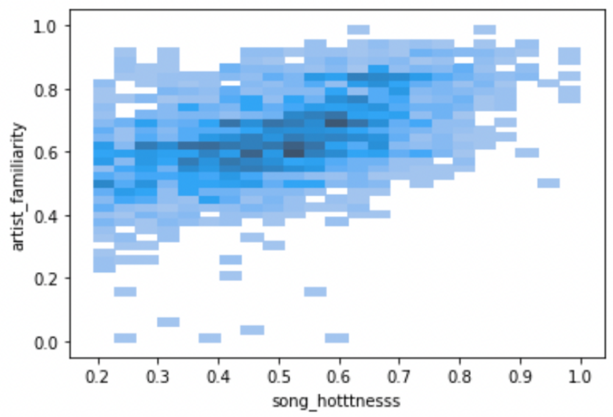
To start, we plotted *song_hotttnesss* as a histogram using seaborn and matplotlib.

song_hotttness

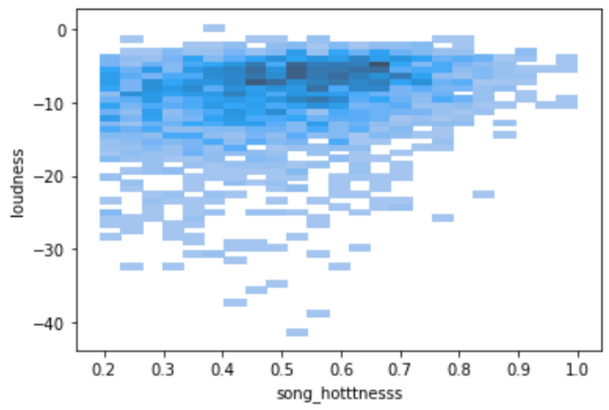


We can see that the distribution for this subset is between 0.2 and 1.0, with the majority of the songs scoring under a 0.7. We then compared features we thought could contain a correlation to the hotness value by plotting them in the form of heatmaps.

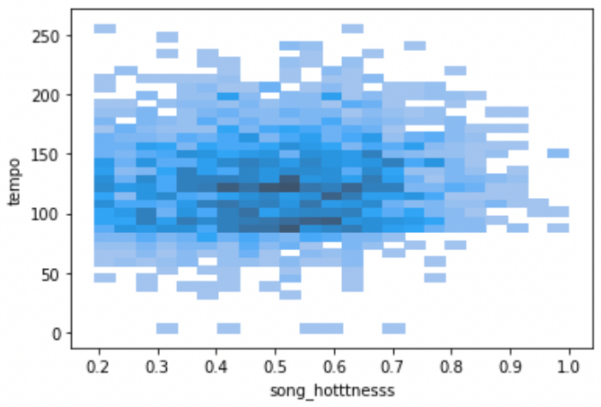
artist_familiarity vs song_hotttnesss



loudness vs song_hotttnesss



tempo vs song_hotttnesss

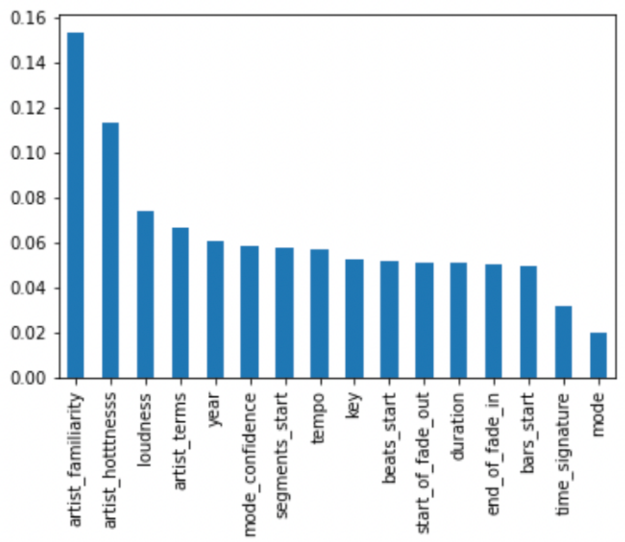


We decided to look at *artist_familiarity, loudness, and tempo* as these naturally felt like they could have a correlation to song popularity. For *artist_familiarity*, the hottest songs appeared to at least have an *artist_familiarity* score of about 0.5. With *loudness*, the more popular songs tended to have dBFS values between 0 and -10, making them loud but not the loudest possible. And with *tempo*, songs with relatively average BPMs (90-150) were more popular than very slow or very fast songs.

B. *Feature Importance*

In order to verify that correlation between certain features and the target existed, we decided to use the ExtraTreesClassifier and SelectKBest from sklearn in order to rank the features in order of what is highest predictive of our target. Both methods agreed that *artist_familiarity, artist_hotttnesss, loudness,* and *artist_terms* were highly predictive of song popularity. Interestingly enough, *duration* ranked pretty low, and *tempo* ended up in the middle when we were expecting it to rank higher. No features were deemed unimportant, so we kept the list that we had.

ExtraTreesClassifier ranked features



SelectKBest ranked features

	Score	Feature_name
0	12.080441	artist_familiarity
1	9.689970	artist_hotttnesss
8	7.388624	loudness
2	1.126279	artist_terms
9	0.826099	mode
15	0.461513	year
7	0.453713	key
6	0.388371	end_of_fade_in
14	0.324071	time_signature
13	0.287015	tempo
4	0.220683	beats_start
11	0.048437	segments_start
3	0.020382	bars_start
12	0.018995	start_of_fade_out
5	0.018879	duration
10	0.006690	mode_confidence

VI. Testing and Results

For both machine learning models we decided to split the data up into 10% for testing and 90% for training as well as use 5-fold cross validation.

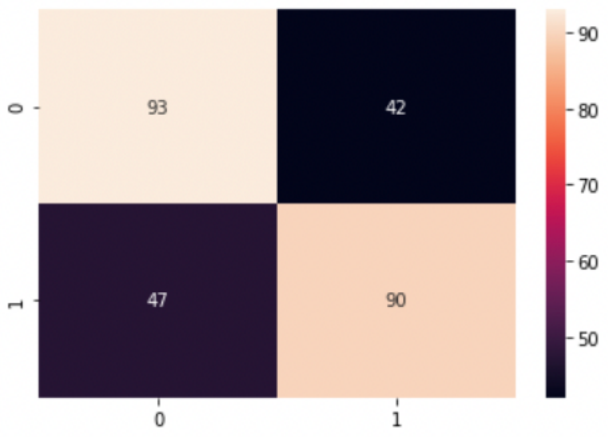
A. *Logistic Regression Results*

Applying the logistic regression model to the dataset was simple, as we used sklearn to import LogisticRegression.

Table 2 Logistic regression results

Accuracy	67.2794
Cross-validation Accuracy	68.4731

To test the accuracy of the model, we also used a confusion matrix, which places the correctly categorised and incorrectly categorised results in a table. The true positive rate (TPR) and true negative rate (TNR) are the correctly predicted results.



True Positive Rate	0.34
True Negative Rate	0.33
False Positive Rate	0.15
False Negative Rate	0.17

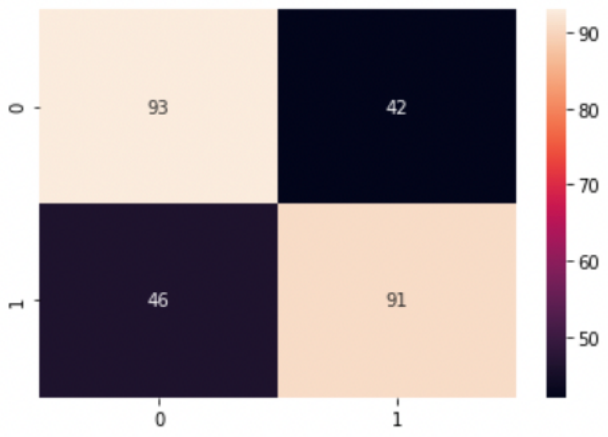
B. *K-Nearest Neighbour Results*

To apply KNN to the dataset, we utilised a similar method as the logistic regression model, by using sklearn to import KNeighborsClassifier. The default value for *k* or *n_neighbors* is set to 5, but we found that setting *k* equal to 18 yielded the most accurate results.

Table 3 KNN results

Accuracy	67.6471
Cross-validation Accuracy	64.1213

We tested the accuracy of the KNN model by yet again utilising a confusion matrix.



True Positive Rate	0.34
True Negative Rate	0.33
False Positive Rate	0.15
False Negative Rate	0.16

VII. Conclusions

We believe that the original aims of this project have been achieved, though with results that leave much room for improvement. The main aim was to determine the potential popularity of a song given certain information about said song. We were able to identify the most influential features in our dataset, and found the more ambiguous features to be more predictive than the metadata features. A possible reason for this is that the traits of a song are not as reflective on performance than the description of a song, and that there are more factors to consider in popularity than just the year of release or how long the song is.

The two models we used for prediction, logistic regression and K-nearest neighbour, output results that were only slightly accurate, with KNN just barely outperforming logistic regression. The models were more accurate than flipping a coin in predicting hottness, but there is a lot of room for improvement and refinement.

Some of the limitations of the project were the format the data came in, the processing required for the full dataset, and the large number of missing data. In order to improve model performance, we would love to run this experiment on the full one million song dataset or supplement the data with more features. This could include sentiment analysis of the lyrical data, genre features, marketing spend features, and even creating our own metric of popularity based on Spotify streams or billboard information.

In addition to being used as a tool for studios or musicians to predict hits, this model could also be reworked as a music reccomender given the datapoints.

References

[1] Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The Million Song Dataset. In Proceedings

of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011), 2011.

[2] Abhishek Singhi, and Daniel G. Brown. “Can song lyrics predict hits,” 11th International Symposium on Computer Music Multidisciplinary Research, 2015.

[3] Andrew Demetriou, Andreas Jansson, Aparna Kumar, Rachel M. Bittner. “Vocals in Music Matter: the Relevance of Vocals in the Minds of Listeners”, 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

[4] Salganik, Matthew J., Peter Sheridan Dodds, and Duncan J. Watts. "Experimental study of inequality and unpredictability in an artificial cultural market." *science* 311.5762 (2006): 854-856.

[5] Koenigstein, Noam & Shavitt, Yuval. (2009). Song Ranking based on Piracy in Peer-to-Peer Networks.. Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009. 633-638.