Spotify Song Success Predictor

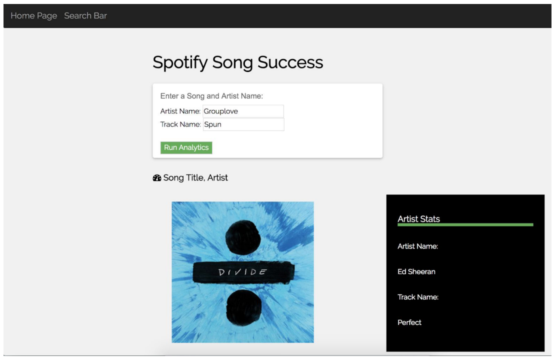
Martin Ziran Xu, Ricky Pan, Liam Shi, Deep Mistry, Brian Long
Data-X. IEOR 135, Spring 2018, University of California, Berkeley

I. INTRODUCTION AND PROBLEM STATEMENT

For new and upcoming artists, there is a constant struggle with getting one's foot into the music industry. Many aspiring artists create lots of great content, however, a lot of the content goes unnoticed because the artists didn't know where to market to. In order to address this issue, we decided to create a product that would use machine learning methods to predict whether a song would be successful in a country or not. The University of Antwerpen[1] performed a similar analysis, however, they only focused on overall popularity around the world. Our product differs from this as we are trying to predict success within each country. They were able to achieve an accuracy of 65% with their models, thus setting a benchmark as to how high our accuracy can reach.

II. USER INTERFACE

One of the very first steps of creating the product involved the user interface. We wanted to create an interface that was easy to interact with, while also providing useful and detailed information. We created a simple input form that would ask for the artist name and track. The first output (Figure 1) would consist of the album art, track name, and artist name.



Spotify Song Success

Enter a Song and Artist Name:
Artist Name: Grouplove
Track Name: Spun

Song Title, Artist

Artist Stats
Artist Name:
Ed Sheeran
Track Name:
Perfect

Figure 1: Sample Artist and Track Details Output

The next step involved the country by country analysis. We decided to display a map of the world with different countries being highlighted as indicators of song success. We see on our example run with Ed Sheeran's "Perfect" (Figure 2), that the dark red shaded areas are the countries where the song would be successful in. The light red countries are the areas where the song wouldn't be successful in, and finally the white areas are all of the countries that don't have sufficient data to make a conclusion.

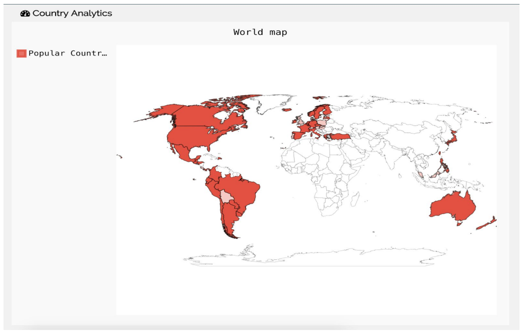


Figure 2: World Map Output

In order to be more clear with our results, we also outputted a table that would give two lists. One list would output the countries that would appreciate the song, and the other showing the countries that would not like the song.

III. BACKEND/SYSTEM ARCHITECTURE

Data is scraped from Spotify's API and Kaggle, with cross-references to Spotify "Top 50" Regional Playlists if possible. The data is originally stored in JSON files, accessible through "POST" and "GET" methods. The "Spotipy" package in Python allows for pythonic POST and GET methods, returning nested dictionaries and lists values for song/artist searches. Song value to song id matching is done through the 'uri' matching process, where each song/album/artist has a unique uri for each component. By matching song uri with song values, we can save these values as entries in a dataframe.

Pre-processing and basic feature extraction occurs through Pandas, Numpy, and Spotipy through various dictionary to dataframe functions (forming columns through dictionary labels). Spotipy's 'audio_features' and 'audio_analysis' functions return the basic features and timbre vectors respectively, with some cleaning performed through apply functions. Models for each country are then performed upon these features, with each model generating a '0' or '1' for failure or success. The Flask app pipelines the dictionary of 0' and 1's into PyGal's map template, and then reroutes this template's uri onto an html page.

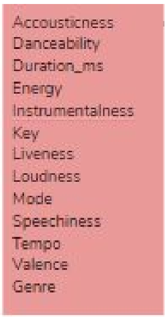
[1] *https://www.tandfonline.com/doi/abs/10.1080/09298215.2014.881888*

IV. MACHINE LEARNING

a) Regional Output Tags

Since the Spotify API only comes with an overall popularity value for each song, we needed to manually define the regional “success” of each song. To do so, we exploited the top 50 regional charts of 53 countries, which are generated and daily updated by Spotify. Luckily, we were able to find a Kaggle dataset[2], which tracked those 53 regional top 50 lists over a period of one year. We extracted all unique songs and cross-referenced them with the Spotify API. In order to transform it into a binary classification problem, we declared a song as “successful” (1) for a specific country, if the song showed up at least once in the Top 50 list of that specific country over the year. At the end we were able to build up a dataset containing 72,291 unique songs. To simplify processing, we only used a subset (1592 songs) of that dataset.

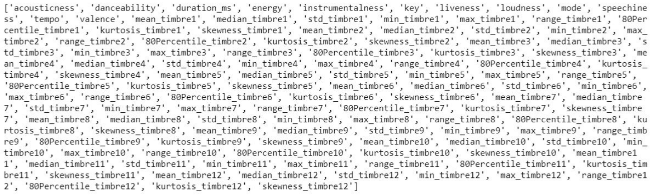
b) Feature Generation



Accousticness
Danceability
Duration_ms
Energy
Instrumentalness
Key
Liveness
Loudness
Mode
Speechiness
Tempo
Valence
Genre

The Spotify API comes with a collection of features, which are generated automatically through a Spotify algorithm.

However, it quickly became clear, that those features are not sufficient for a successful classification algorithm. Models trained on those features misclassified all songs in the same category. We thus extended our feature list, by introducing the timbre vectors of a song. The 12 timbre values describe the characteristic quality of a sound and can be interpreted as the tone colour. Spotify is able to generate those timbre values for a song, by separating a song into segments and assigning 12 timbre values for each of those segments. Each of those vectors comprises the information for all segments and have a dimension around 1000 entries. In order to generate features, we took statistical descriptions of those timbre vectors, such as: mean, median, percentiles, kurtosis, skewness, std, range, min/max. Using this feature extraction procedure we ended up with a collection of 120 features (see Figure 3).



```
['acousticness', 'danceability', 'duration_ms', 'energy', 'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'speechine
ss', 'tempo', 'valence', 'mean_timbre1', 'median_timbre1', 'std_timbre1', 'min_timbre1', 'max_timbre1', 'range_timbre1', '80Per
centile_timbre1', 'kurtosis_timbre1', 'skewness_timbre1', 'mean_timbre2', 'median_timbre2', 'std_timbre2', 'min_timbre2', 'max_
timbre2', 'range_timbre2', '80Percentile_timbre2', 'kurtosis_timbre2', 'skewness_timbre2', 'mean_timbre3', 'median_timbre3', 's
td_timbre3', 'min_timbre3', 'max_timbre3', 'range_timbre3', '80Percentile_timbre3', 'kurtosis_timbre3', 'skewness_timbre3', 'me
an_timbre4', 'median_timbre4', 'std_timbre4', 'min_timbre4', 'max_timbre4', 'range_timbre4', '80Percentile_timbre4', 'kurtosis_
timbre4', 'skewness_timbre4', 'mean_timbre5', 'median_timbre5', 'std_timbre5', 'min_timbre5', 'max_timbre5', 'range_timbre5',
'80Percentile_timbre5', 'kurtosis_timbre5', 'skewness_timbre5', 'mean_timbre6', 'median_timbre6', 'std_timbre6', 'min_timbre6',
'max_timbre6', 'range_timbre6', '80Percentile_timbre6', 'kurtosis_timbre6', 'skewness_timbre6', 'mean_timbre7', 'median_timbre
7', 'std_timbre7', 'min_timbre7', 'max_timbre7', 'range_timbre7', '80Percentile_timbre7', 'kurtosis_timbre7', 'skewness_timbre
7', 'mean_timbre8', 'median_timbre8', 'std_timbre8', 'min_timbre8', 'max_timbre8', 'range_timbre8', '80Percentile_timbre8', 'ku
rtosis_timbre8', 'skewness_timbre8', 'mean_timbre9', 'median_timbre9', 'std_timbre9', 'min_timbre9', 'max_timbre9', 'range_timb
re9', '80Percentile_timbre9', 'kurtosis_timbre9', 'skewness_timbre9', 'mean_timbre10', 'median_timbre10', 'std_timbre10', 'min_
timbre10', 'max_timbre10', 'range_timbre10', '80Percentile_timbre10', 'kurtosis_timbre10', 'skewness_timbre10', 'mean_timbre1
1', 'median_timbre11', 'std_timbre11', 'min_timbre11', 'max_timbre11', 'range_timbre11', '80Percentile_timbre11', 'kurtosis_tim
bre11', 'skewness_timbre11', 'mean_timbre12', 'median_timbre12', 'std_timbre12', 'min_timbre12', 'max_timbre12', 'range_timbre1
2', '80Percentile_timbre12', 'kurtosis_timbre12', 'skewness_timbre12']
```

Figure 3: Feature List

c) Popularity Prediction

Using the described output and features, various machine learning models were trained and their accuracy improved using specific techniques learned in class.

Before making predictions on the regional outputs, we tried our methods on the overall popularity prediction and compared our results to the University of Antwerpen benchmark (*see section I*). The popularity score is defined through spotify and lies within the scale from 0 to 100 (very successful). Attempts to predict this score using regression models failed and we thus decided to transform it into a classification problem. For that purpose, we created 3 quantile bins, of which the first and third are defined as “0-not successful” and “1-successful”, while all the data in the middle bin was thrown away, in order to create a gap between those two classes. The gap size has been varied, but no significant improvements were determined. At the end we chose to use a 35-30-35 percent split.

For our convenience, we built ourselves a function, which automatically trains the following models and compare their train/test accuracy, confusion matrices and auc scores with each other: SVM, Random Forest, Gradient Boosting, Logistic Regression, KNN, Perceptron. Note, that we chose to use the AUC score as a performance metrics as well. As the area under the Receiver Operating Characteristics (ROC) curve, it can give a better metrics, especially for unbalanced datasets. It turned out that the Random Forest Classifier is the best classifier for our use case.

In the next step we performed feature selection using the forward stepwise selection method. We first ranked all our 120 features by their importance using a Random Forest Classifier. Figure 4 shows the top 10 features for our model.

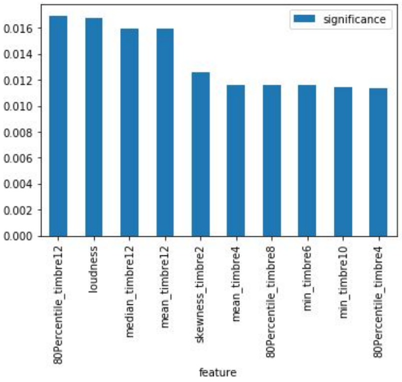


Figure 4: Top 10 model Features

Note, that almost all of the top ranked features (except loudness) are somehow related to the timbre vectors. This can be explained, that other features, especially the spotify tags, can be extracted through those timbre vectors and are thus less important for our model. In order to prevent the model from overfitting, we tried to find the optimal number of features for our model. For that purpose, we started off, by only training the model with the top 5 most important features and increased the number of features by 5 until we hit 120.

[2]

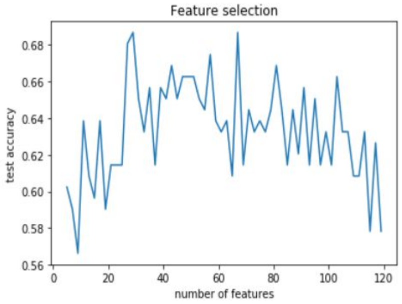


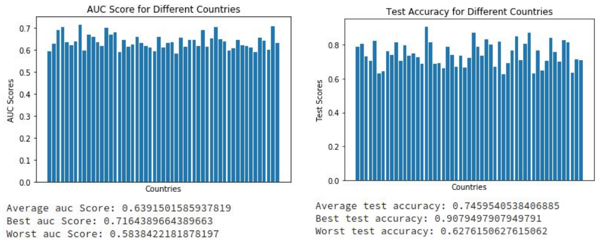
Figure 5: Test accuracy over number of features

Figure 5 shows the test accuracy over the number of features. It is clear, that our model performs the best for features between 30 and 80 features. We thus decided to limit our model to the top 60 most important features.

Using this framework we were finally able to reach a test accuracy of 67.50%, which is an 2.5% improvement in comparison to the University of Antwerpen benchmark.

d) Predicting Regional Data

After verifying the performance on the overall popularity, we trained 53 models for each country to make regional predictions.



```
Average auc Score: 0.6391501585937819
Best auc Score: 0.7164389664389663
Worst auc Score: 0.5838422181878197
```

```
Average test accuracy: 0.7459540538406885
Best test accuracy: 0.9079497907949791
Worst test accuracy: 0.6276150627615062
```

Figure 6: AUC and Test Accuracy results for regional data

As can be seen from these results (Figure 6), an average AUC score of 0.64 was obtained ranging from a highest value of 0.72 to lowest of 0.58. Looking at the test accuracy of the regional data performance, an average test accuracy of 0.75 was obtained for the model, ranging from highest to lowest values of 0.91 to 0.63 respectively. The high variance between the countries can be explained through the unbalanceness of the dataset.

V. DATA EXPLORATORY

Given the big dataset, we were also able to perform a few interesting data analysis on music in general. For a better interpretability, we focused on the spotify tags (see Section IV b)) and created the following analysis: 1) average score for each country, 2) average score for each genre 3) average score on each day.

As an example, two of those plots are displayed for "Accousticness" in the following figures.

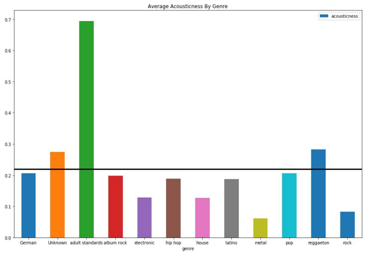


Figure 7: Acousticness grouped for differenc genres

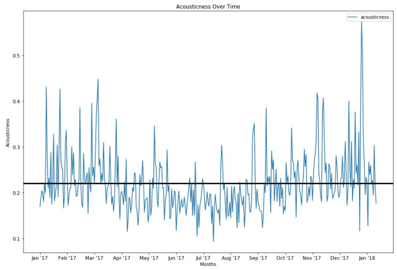


Figure 8: Acousticness over time

Adult Standards songs for example have a significant higher accousticness than the other genres.

The time series over a period of one year (Figure 8) shows that the accousticness does not really deviate much over the year, except of a christmas peak and a slightly higher value in the winter months.

Those plots are available in our GitHub for all of the 10 spotify tags.

VI. Conclusion and Future Steps

Using Machine Learning models learned in class, we were able to beat the current benchmark of the University of Antwerpen with our model and create a clean frontend to interact with potential users.

However, there are still a few steps, we could take to improve our performance. First, we can use the whole dataset of 17,000 songs to train our model. Expanding the training set by 15 times, will definitely improve our performance.

Second, we can include metadata to our feature lists. Those would include metrics like artist hotness or advertisements and lyrics, which could be scraped through the million song database on Kaggle. While this might improve our model performance, it would also limit our use case, since those meta-data is usually not available for new or arising artists.

Lastly, we could feed our raw timbre vector into a convolutional neural network, which would perform automatic feature selection for us. Instead of extracting features by manually taking statistical descriptions of those vectors, the neural network would do it automatically. Through the nonlinearities it would also be able to capture nuances going beyond our current model. Especially through the size of our big dataset, it could be a major improvement to our accuracies.