2018

Springboard Data Science

Rene Sanchez

[SPOTIFY USER ANALYSIS]

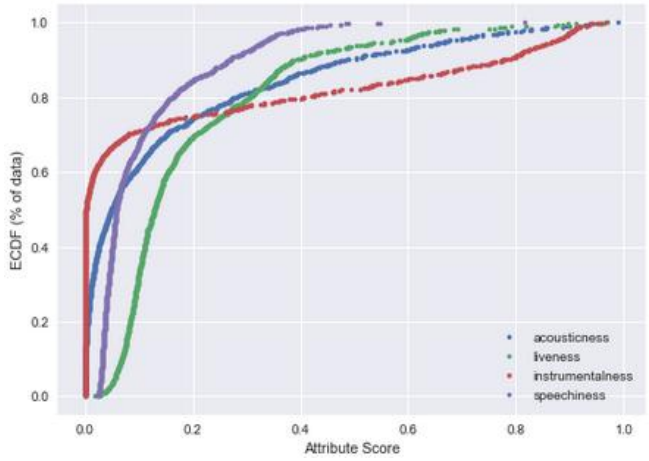
This report explores a single Spotify user's data to extract insights and define a user profile. A recommendation model is designed giving consideration to multiple algorithms and hypothesis'. The end result is a profile with which one can query song data to feed to the recommendation model to predict which songs the user will like. Model performance statistics are laded out on the last page.

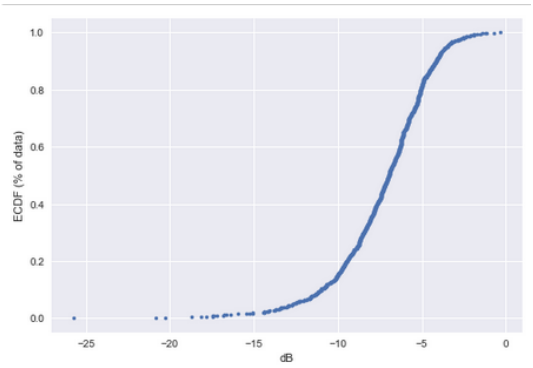
The aim of this report was to extract trends from Spotify user data to better understand their preferences and create a recommendation engine that can accurately predict what songs the user will like. Spotify, like many tech companies today, is constantly working to improve their recommendation systems to ensure user engagement and retention. Through traditional statistical analysis and machine learning techniques, I created a clear profile of attributes that defines the user's preferences.

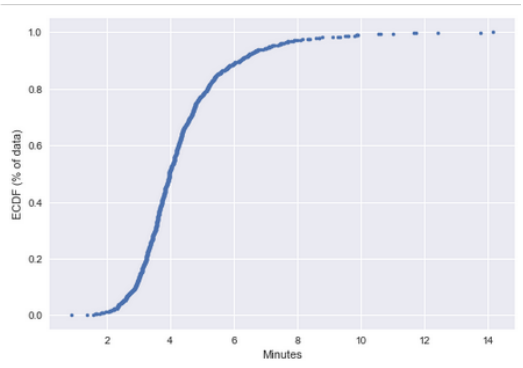
The dataset used here was obtained from Kaggle and contains 2017 songs with their attribute scores and whether or not they were liked by the user. Fortunately, this data was gathered from the Spotify API, therefore the data started off very clean with the columns properly formatted as 'int' and 'float' types. Regardless of this, I checked for missing values anyway and indeed there were none.

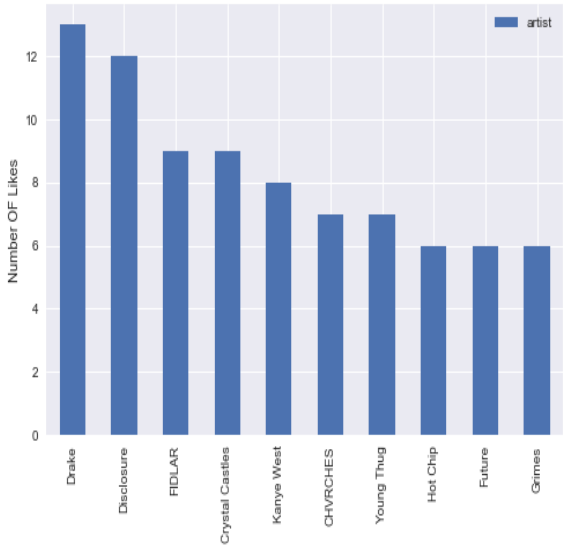
The Spotify API is freely available to generate the attribute scores of a single song or set of songs for analysis and prediction purposes. However, in order to get specific user data with which to train a recommendation engine the user must provide permission via their Spotify app in order for their playlists and liked songs to be accessible. This presents some limitations as to what a third party developer can do as the following recommendation engine may or may not generalize to sets of users.

I began by splitting the liked songs from the rest of the data and inspecting the numerical attributes visually to determine if there were any clear patterns present in the data. In inspecting the distribution graphs of the attribute scores, we see that some are normally distributed and others are heavily skewed (right graph). In context, what this means is that the attributes that are normally distributed are not concentrated at certain values but are instead relatively evenly spread out over the range. By contrast, the skewed attributes provide evidence for preferred values for the given attributes because of the concentration of songs around that value. The variables that exhibited skewness are Acousticness, Liveness, Instramentalness, Speechiness, Duration, and Loudness. The first four are plotted above because they share the same scale; Loudness and Duration are plotted below on their given scales. This cursory visual analysis gives us some indication as to what the user's focus is in determining what song they like and gives us criteria that can be used to selected songs from Spotify's database to run the recommendation algorithm on.









Next, I plotted the categorical attributes to check for differences in the frequency of categories. We find that the key denoted by 3 (D#) is the least popular key and the 4 (4/4) time signature is by far the most prevalent value. As it turns out though, these values are known throughout the music industry to be the most common value for these attributes; therefore it is likely that these patterns do not reflect user preference but rather the nature of music itself. Lastly, I generated a plot of the top 10 most liked artist which reveals that the user's favorite artist is likely Drake or Disclosure. With this information we can confidently present the user new releases from these 10 artists to keep them up to date with their favorite music.

With some variables of interest defined, I proceeded to conduct a statistical examination of the numerical attributes that exhibited skewness as well those that appeared to have mean differences between liked and disliked songs.

To do this, I split the dataset into the songs that were liked and those not liked to examine the mean differences between the sets. The variables that had considerable mean differences given their scale were Acousticness, Instramentalness, Duration, and Loudness which supports our visual analysis of the liked dataset.

	Z-test results	
Variable	**Statistic**	**P-value**
instrumentalness	-6.93091	4.18e-12
loudness	3.24042	0.001194
duration_min	-6.65947	2.75e-11
liveness	-1.18385	0.236471
speechiness	-6.99662	2.62e-12
acousticness	5.86831	4.40e-09

Z-tests were conducted to determine if these differences were significant and indeed acousticness, instramentalness, loudness, duration, and speechiness all had significantly different means suggesting they are prominent features that the user uses to determine whether or not they like a given song. We can now pull songs from the database with values that fit the visual profile above on these attributes with significant mean differences. This will allow us to filter Spotify's massive database down to a manageable size and recommend songs from a set that the user is likely to enjoy.

Before creating a predictive model for the data I explored the possibility of some features being more important than others. The hope was that this would reduce any noise in the predictive model and increase accuracy by removing unhelpful variables. I used Lasso regularization to determine if any variables should be dropped from the model due to their coefficient being reduced to 0 with an L1 constant of .01 (chosen based on max scale values).

Lasso reduced energy, key, and time signature coefficients to 0 indicating they are not well correlated with the target variable. Therefore for each model I would run one model with the recommended variables removed and one with all the variables included to compare the initial results before improving the model.

I began by using an Artificial Neural Network (ANN) to predict the target variable from the given song attributes. Two ANNs were designed using either the total number of variables or the recommended size given dropped variables. After several iterations that are not noted in the Jupyter notebook, 2 layers with 15 and 11 nodes was chosen; more layers significantly reduced accuracy while reducing variance slightly and more nodes improved accuracy less than the consequential increase in variance.

ANN Model	Variance	Mean Accuracy
Full Model result:	0.0257	70.652%
Lasso result:	0.0250	49.477%

With the structure determined, the data was split into training and test set and scaled for analysis. The initial results were determined using a batch size of 10 over 50 epochs since the training dataset is not particularly large. 5-fold cross validation revealed that following the Lasso recommendation significantly reduces accuracy of the ANN. Therefore all variables will be included in the final model which will have its hyper parameters tuned.

To tune the final model, I utilized GridSearchCV from the SKLearn library to test the batch and epoch hyper parameters. I chose not to test the optimizer here since the literature on the optimizers supported by Keras indicates that Adam is very likely the best option given the nature of the data and problem. Numerous values were tested through GridSearchCV for batch size and epochs based on the size of the dataset with batches size 1-15 and epochs 50-200 explored. There appeared to be convergence at batch size = 1 and epoch=99, and given the computational resources at hand, I will accept these as the optimal values. These parameters yielded a 72.987 % accuracy rate on the training data, so we will now feed in the test data for the model to predict with.

The final ANN model yielded a 74.917% accuracy rate on the test set, a better score than training but one to be challenged by other methods.

SVM Model	Variance	Mean Accuracy
Full Model result:	0.0287	73.686%
Lasso result:	0.0024	51.342%

Next, I used a Support Vector Machine (SVM) to create a predictive model through the same methodology. Using initial values of C = 1.5 for the tolerance value and gamma=.14 for the vector distance parameter, the model was tested with both the Lasso recommendations and the full model. Again we see here that the Lasso recommendation model underperforms and the full model performs slightly better than the ANN did in its training.

The final SVM model included all variables and was tuned in a similar manner by exploring in an exponential manner. The tolerance parameter ‘C’ was explored from 1-1000 and ‘gamma’ was explored from .01-100 with GridSearchCV converging at C = 1 and gamma=.136 with an accuracy rate of 74.562%. This figure already competes with the ANN so we can expect that it may outperform it.

In the final SVM model I found that it does indeed outperform the ANN with an accuracy rate of 77.227%. This is a noticeable improvement over the previous model but emphasizes the necessity for multiple methods to arrive at the optimal predictive power.

Given that the Lasso recommendation has thus far consistently underperformed I chose not cross validate the following models and instead proceed to parameter tuning for a Random Forest classifier (RFC). The RFC uses only a subset of the total amount of features to create a predefined number of decision trees and then average those trees out. The subset is conventionally chosen by taking the square root of the number of features, so this methodology will be used for the GridSearch. The number of trees was explored exponentially from 10 – 10,000 and converged at 90 trees. This RFC configuration yielded a 77.887% accuracy rate on the training data and yielded the highest accuracy thus far on the test data, 79.208%.

The final model I used is a Gradient Boosted Machine (GBM) which operates in a similar fashion as the RFC but instead of treating each decision tree as independent it trains the next tree based on the previous tree's residual. I predicted that this algorithm would perform as least slightly better than the RFC as the attributes of a song seem as those they would be related to each other to some degree.

There are several parameters to be specified for GBM: Number of Estimators, Learning Rate, and Max Depth. Feature selection will be handled using square root selection just as was done in the RFC. Just like the RFC, the Number of estimators determines the number of trees to utilize however unlike the RFC we define the learning rate here since the trees iterate based on the previous tree and adjust according to this learning rate. Finally, Max Depth defines how large each tree can be and, based on my experience training on the data here, is related to the number of features present in the dataset.

The GBM had 77.479% accuracy on the training set and converged at a Learning Rate of .2, Max Depth of 5, and Number of Estimators of 365. Values were explored for these parameters exponentially just as was done in the previous models to ensure that an earnest attempt was made to find a global minimum rather than a local minimum. This model specification yielded 80.198% accuracy on the test set, taking the top spot for performance on this dataset.

To implement these findings, we would query songs whose attributes of interest (defined in the statistical analysis) that fall within the concentrated range identified in the visual inspection of the data. Using these queried songs, we would then run the final GBM model and recommend the songs that are classified as '1' in the model results. We would expect this model to be able to fill a playlist with approximately 80% of songs that the user will like and add to their own library. Additionally, as the user goes through our predictions and likes or dislikes them, we would add those to our training data to further improve the models performance over time.

Final Model Results and Statistics

	precision	recall	f1-score	support
Disliked	0.81	0.81	0.81	158
Liked	0.79	0.79	0.79	145
avg/total	0.8	0.8	0.8	303
Gradient Boosted Tress Accuracy		**80.20%**		
Disliked	0.81	0.79	0.8	158
Liked	0.78	0.79	0.78	145
avg/total	0.79	0.79	0.79	303
Random Forest Accuracy		**79.21%**		
Disliked	0.77	0.81	0.79	158
Liked	0.78	0.73	0.75	145
avg/total	0.77	0.77	0.77	303
Support Vector Machine Accuracy		**77.23%**		
Disliked	0.76	0.76	0.76	158
Liked	0.74	0.74	0.74	145
avg/total	0.75	0.75	0.75	303
Artificial Neural Network Accuracy		**74.92%**		