# CAPSTONE PROJECT NO.2

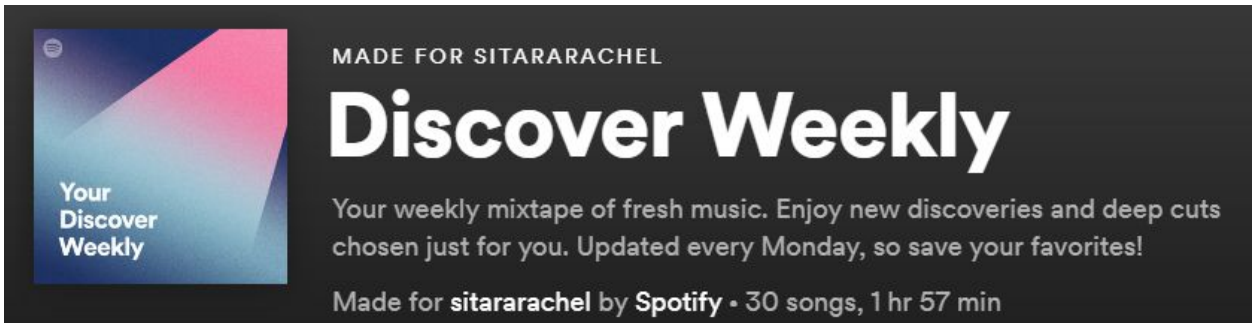
# SPOTIFY SONG RECOMMENDATION SYSTEM

# HIGH LEVEL REPORT

**SITARA ABRAHAM**
**SPRINGBOARD 2019**

## OVERVIEW

The aim of this project is to build a model that utilizes a user's current musical tastes to predict whether s/he will enjoy songs in new playlists. Currently, Spotify has a very intelligent recommendation system; it's "Discover Weekly" playlists are tailored to each user and provides new songs which align with the user's historic tastes.




## PROBLEM

One of the most dynamic interests, throughout the ages and geographic location, is music. From being the voice behind the lyrics, to writing the lyrics, to producing the sounds, to blending songs together in a mix for energetic crowds, to simply listening--there is rarely a person you will meet who completely dislikes music altogether.

That being said, we all love the feeling of discovering a new song that we end up playing on repeat. But inevitably, we grow tired. Bored. And yearn for the novelty of a new song that aligns with our moods and tastes. For some, this search for new songs is thrilling, the backbone of their love of music even. For others, it's a time consuming process with usually little to yield as a result.

## DATA

A major barrier to this being a usable model amongst multiple Spotify users is the fact that I would not have access to anyone else's account data except my own. For the purposes of this project, these limitations are understandable.

The first hurdle was retrieving and wrangling the data of interest. Using Spotify's well-documented API and a Python library package made specifically for this purpose (Spotipy),

I was able to use my API credentials to pull in my music data under the scope of modifying public playlists.

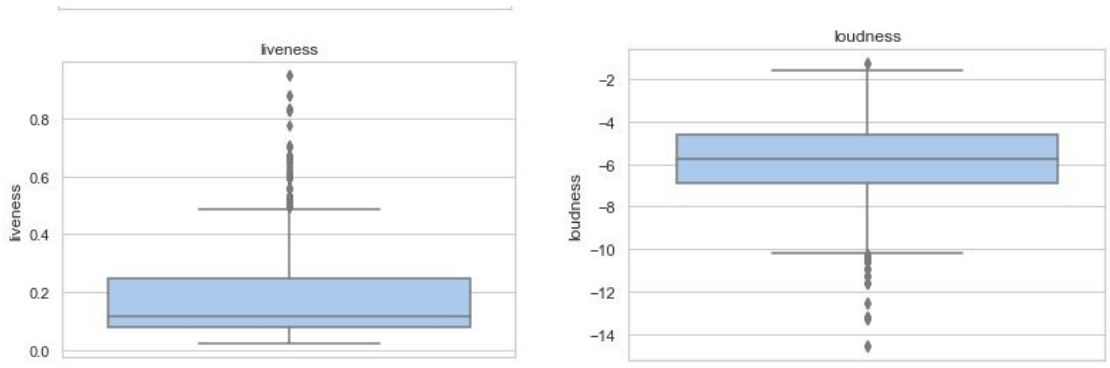
After creating two playlists in my account, one for songs I like and one for songs I dislike, I was able to extract the audio features of these songs and put all this data into a new dataframe.

They key players in this conflict of whether I will like a song did, surprisingly, not have anything to do with genre or artist. Rather, they were:

- Acousticness
- Danceability
- Duration in milliseconds
- Energy
- Instrumentalness
- Key
- Liveness
- Loudness
- Speechiness
- Tempo
- Time signature
- Valence

In terms of an initial hypothesis, it looked as though I seem to enjoy more lively songs. The majority of values of certain features such as danceability, tempo, and valence were on the higher end of the individual spectrums, indicating the majority of the liked songs had high values in these areas.

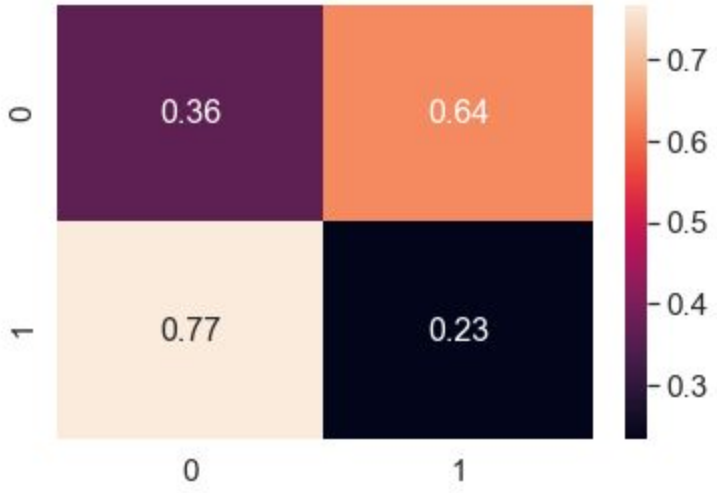
I then visualized these ranges for each of these key features. While most of the features had a minimal or reasonable amount of outliers, some of them were very skewed by outliers. Because the validity of the recommendation might be adversely affected by these skews, the features “time signature” and “instrumentalness” were dropped from the dataframe.




Next, I visualized the distribution of the remaining features. Again, while some features had reasonably normal distributions, others had blatant skews or even bimodal distributions. To achieve the maximum validity of the model, the features with the least normal distributions ("speechiness","liveness","key","acousticness") were dropped.

Ideally, the end product for this project would be a comprehensive recommendation system, one that uses a mix of machine learning models to aid each other. For the sake of this project, I chose three ML models: a decision tree classifier, logistic regression, and KNeighbors classifier.

I wanted to include a decision tree in this mix because of relatively high number of dimensions; even though it is still separating the data set into two classes this model has its advantages because the boundary between songs I like and songs I dislike is not necessarily linear (as seen by the hypothesis testing).




A normalized confusion matrix for a decision tree with no parameter tuning. This model accurately predicted liked songs

23% of the time.

I also utilized logistic regression as part of the mix of models. Using a Python package called statsmodels, I was able to see which, if any, of the logistic features were significant and to what effect:

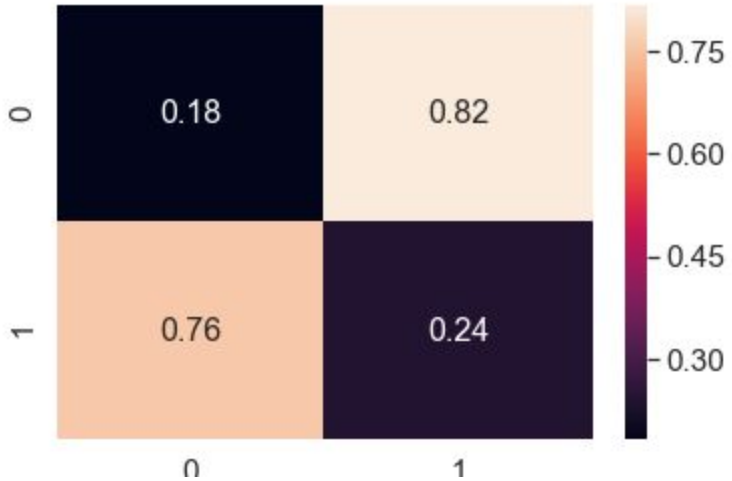
Logit Regression Results

| | | | |
|---|---|---|---|
| Dep. Variable: | target | No. Observations: | 1291 |
| Model: | Logit | Df Residuals: | 1284 |
| Method: | MLE | Df Model: | 6 |
| Date: | Tue, 20 Aug 2019 | Pseudo R-squ.: | 0.003171 |
| Time: | 20:00:10 | Log-Likelihood: | -890.67 |
| converged: | True | LL-Null: | -893.50 |
| | | LLR p-value: | 0.4615 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| danceability | 0.5241 | 0.389 | 1.347 | 0.178 | -0.238 | 1.286 |
| duration_ms | 4.289e-07 | 1.2e-06 | 0.357 | 0.721 | -1.93e-06 | 2.79e-06 |
| energy | -0.1287 | 0.365 | -0.353 | 0.724 | -0.843 | 0.586 |
| loudness | 0.0029 | 0.031 | 0.093 | 0.926 | -0.058 | 0.063 |
| mode | 0.0355 | 0.113 | 0.315 | 0.753 | -0.186 | 0.257 |
| tempo | -0.0023 | 0.002 | -1.395 | 0.163 | -0.006 | 0.001 |
| valence | -0.0187 | 0.329 | -0.057 | 0.955 | -0.663 | 0.626 |

Unfortunately upon initial glance, according to the summary output, none of the features were significant and the model did abysmal job explaining the variability in liked and disliked songs.

Finally, I wanted to round out the mix by using another supervised model that clusters data points: K Nearest Neighbors.




A K Nearest Neighbors normalized confusion matrix. This model, with no specified optional parameters, did not fare too much better than the decision tree, objectively.

For each model, I started by instantiating a model with no specified optional parameters, and then looked at the accuracy score and classification report.

## EVALUATION

For the sake of the context of predicting songs I like, using a metric as simple as accuracy score would suffice; the percent of songs I accurately enjoy out of a playlist recommended for me would be indicative of how successful the models are.

Before any hyperparameter tuning, the accuracy scores were as follows:

- Decision Tree: 29.7%
- Logistic Regression: 49%
- K-Nearest Neighbors: 21.1%

Looking at these results, the logistic regression model would have the most predictive weight.

After tuning the models, the accuracy scores were as follows:

- Decision Tree: 45.2%
- Logistic Regression: 49.2%
- K-Nearest Neighbors: 31%

The decision tree gained the most improvement with the parameter tuning, whereas the logistic regression model more or less stayed the same and continued to have the highest score.

## CONCLUSION

While these scores can seem to be pretty low, in the context of Spotify's current recommendation system-fueled playlists I think these models are on par.

The most volatile variable in this model was not any audio feature or a Spotify component--but rather me. The accuracy and any baseline comparisons of the model to Discover Weekly were not based on an objective ground truth, but rather my tastes. Even when listening to the albums and playlists Spotify recommends for me, I will enjoy maybe 30-40% of the songs--and that is with them using all of the audio features as a basis for recommending the songs.

That being said, I think the scores of these models when used in conjunction would perform decently in terms of predicting songs I'd probably enjoy. The low accuracy scores were to be expected, even understandable. If Spotify's system, using all the audio metadata, could make a playlist in which I only liked 40% of the songs, my system using less features would have predictable score of under 40% for some of the models.

In terms of next steps, the whole exploration of the dataset and the resulting model has produced valuable insights in terms of my taste in music in the context of how lively or positive a song is. As it turns out, there is no significant distinction in these traits for songs I like and dislike. I can obviously confirm this conclusion, I enjoy both lively and more mellow songs, happy and perhaps more emotional.

While the problem of making the process of finding new songs seamless is not completely solved with my system, I am in a better place of finding a wide variety of songs that could suit my tastes. I of course can utilize Spotify's recommendation; however, using the API I now have more control of curating training data for which to base a recommendation system. To take this model further, I can break out songs into further classifications in which I could not do in Spotify and then make predictive models for those breakouts.