Perfect Match – Final Project Report
CS 4262 (Spring 2022)

Keshvi Mahalingam [*1] **Zinnie Zhang** [*1]

Abstract

Currently, there exist many companies who have built software applications that offer music streaming and proprietary music recommendation services such as Spotify, Apple Music, and Amazon Music Unlimited. For example, Spotify is able to learn from a user's liked songs and curate multiple unique playlists that might suit the user's tastes. The aim of this project is to further explore the problem of music recommendation used by 450+ million paid music streaming subscribers globally. (MIDiA Research, 2021) In this project, the authors tested various models for classifying a song into a "Happy" or "Sad" category. They found that Logistic Regression, used with five audio features, yield the best prediction accuracy of 91.67%.

1. Introduction

1.1. The Dataset

The dataset used in this project was a Pandas Dataframe consisting of song metadata from Spotify. Spotify data was chosen as it is the most popular music streaming platform, thus implicating access to more data. Moreover, Spotify provides a robust API from developers to retrieve comprehensive metadata about playlists and songs. In the Spotify application, Mahalingam and Zhang found two playlists consisting of 100+ songs, titled "Happy Hits" and "Sad Songs" created by the Spotify application. The Spotify API allows developers to access information about multiple features about a song such as title, artist but 14 features were selected for further data exploration: *length, popularity, acousticness, danceability, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo, time-signature, and valence.*

[*]Equal contribution [1]School of Engineering, Vanderbilt University, Nashville, Tennessee, USA. Correspondence to: Keshvi Mahalingam <keshvi.j.mahalingam@vanderbilt.edu>, Zinnie Zhang <xin.yi.zhang@vanderbilt.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1.2. Literature Review

Mahalingam and Zhang reviewed several reference articles and noted three papers that influenced their project design:

1. In an article titled "How to Find New Songs on Spotify Using Machine Learning" by Logan Norman, Norman uses a dataset of nearly 10,000 songs and trains his data with three models: Logistic Regression, DecisionTreeClassifier, and RandomForestClassifier. Norman found that RandomForestClassifier had a 99.3% accuracy. Mahalingam and Zhang found this result to be abnormally high, however, Norman asserts that he checked for overfitting by adding more folds to the cross validation. Considering the extremely positive results of Norman, Mahalingam and Zhang also decided to test DecisionTreeClassifier as one of the models.

2. In an article titled "Organizing Spotify Playlist with Data Science" by Ankush Bharadwaj, the author tackles the problem of splitting a large Spotify playlist into smaller playlists of similar songs. Ankush Bharadwaj conducts data exploration using the Seaborn library to visualize the correlation and hierarchical clustering of the audio features. He also uses Principal Component Analysis to plot the variance explained by adding additional features. Bharadwaj hopes to find an *elbow* in the plot but the PCA plot is smooth. Finally, Bharadwaj uses Logistic Regression and K-Means Clustering to train his data. No performance results are discussed. Using Bharadwaj's article as a reference, Mahalingam and Zhang also conducted similar data exploration and incorporated the use of Logistic Regression and K-Means Clustering.

3. Lastly, in an article titled "Using KNN Algorithm to Classify Spotify Songs into Playlists" by Sameeha Afrulbasha, the author tackles the problem of classifying a new song into one of two playlists: a *happy* playlist and a *euphonious* playlist. To process her data, Afrulbasha graphs each music feature using matplotlib and notices distinct, considerable gaps between lines for *happy* and *euphonious* for three features: **energy, acousticness, and valence**. She then uses the three

main attributes to train her model using K-Means Clustering and reports an optimal performance of 87.0%. Mahalingam and Zhang also used the same feature selection method and incorporated the use of K-Means Clustering.

In conclusion, from reviewing prior literature, Mahalingam and Zhang were able to draw inspiration to design their methods to achieve the optimal recommendation model. They were aiming for results in the range of 87.0-99.7% based on performance of data scientists who conducted similar research.

2. Methods

As mentioned in the introduction, the dataset used for this project was a Pandas Dataframe composed of Spotify track metadata. The tracks included in the dataframe were extracted from two Spotify playlists, titled "Happy Hits" and "Sad Songs". Using the Spotify API, Mahalingam and Zhang were able to populate the dataframe with the 14 track audio features (or input features) described in Table 1 (see left).

** Note: This project validates previous classifications as some audio feature values (i.e. danceability) are the result of prior classifications.

2.1. Exploratory Data Analysis

Prior to any preprocessing or training, Mahalingam and Zhang familiarized themselves with the dataset through exploratory data analysis. The first step in this analysis was to utilize Seaborn's heatmap to visualize the correlations between the input features, as is seen in Figure 1.

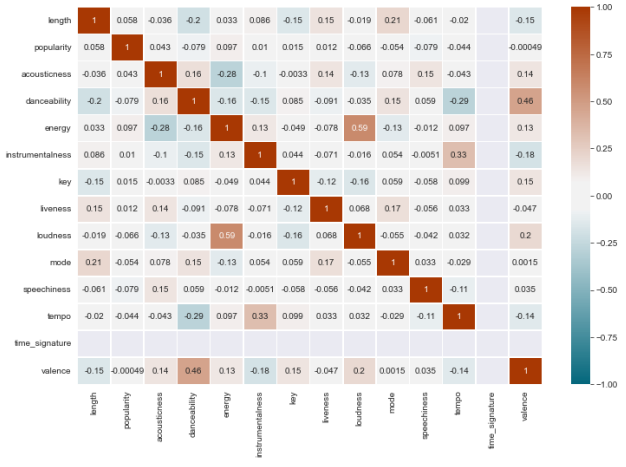

Figure 1. Seaborn heatmap depicting the correlation values between the Spotify track audio features in the dataset.

After analysis of these correlation values, Mahalingam and Zhang were able to conclude that 'energy' and 'loudness' were the only two input features in the dataset which displayed a somewhat strong correlation. Next, Mahalingam and Zhang used Matplotlib's plot function to create a 2D line plot of each input feature across all songs, using red and blue lines to denote songs from the happy and sad playlists, respectively. The purpose of creating these line plots was to identify the input features with the most observable separability between the happy and sad songs. After analyzing the generated line plots, it was concluded that the 'acousticness' and 'energy' line plots came closest to exhibiting separability between the happy and sad songs, as shown in Figures 2 and 3.

The happy and sad song datasets were then scaled independently and combined into a single dataset. Mahalingam and Zhang performed Principal Component Analysis on this scaled and combined dataset with the goal of determining which features could be eliminated. To do so, they plotted

Table 1. Audio features used for classifying between Spotify track humors.

FEATURE	DESCRIPTION	VALUE TYPE
LENGTH	LENGTH OF THE TRACK IN SECONDS.	INT
POPULARITY	A VALUE BETWEEN 0 AND 100 REPRESENTING THE POPULARITY OF THE TRACK'S ARTIST. CALCULATED FROM THE POPULARITY OF ALL THE ARTIST'S TRACKS. A VALUE OF 100 REPRESENTS THE HIGHEST ARTIST POPULARITY.	INT
ACOUSTICNESS	A CONFIDENCE MEASURE FROM 0.0 TO 1.0 OF WHETHER THE TRACK IS ACOUSTIC. 1.0 REPRESENTS HIGH CONFIDENCE THE TRACK IS ACOUSTIC.	FLOAT
DANCEABILITY	A VALUE BETWEEN 0.0 AND 1.0 REPRESENTING HOW SUITABLE THE TRACK IS FOR DANCING BASED ON A COMBINATION OF MUSICAL ELEMENTS INCLUDING TEMPO, RHYTHM STABILITY, BEAT STRENGTH, AND OVERALL REGULARITY. A VALUE OF 1.0 INDICATES THE HIGHEST DANCEABILITY.	FLOAT
ENERGY	A MEASURE FROM 0.0 TO 1.0 REPRESENTING A PERCEPTUAL MEASURE OF INTENSITY AND ACTIVITY. TYPICALLY, ENERGETIC TRACKS FEEL FAST, LOUD, AND NOISY.	FLOAT
INSTRUMENTALNESS	PREDICTS WHETHER THE TRACK CONTAINS NO VOCALS. "OOH" AND "AAH" SOUNDS ARE TREATED AS INSTRUMENTAL IN THIS CONTEXT. THE CLOSER THE INSTRUMENTALNESS VALUE IS TO 1.0, THE GREATER LIKELIHOOD THE TRACK CONTAINS NO VOCAL CONTENT.	FLOAT
KEY	ESTIMATED KEY OF THE TRACK.	INT
MODE	THE MODALITY (MAJOR OR MINOR) OF THE TRACK, THE TYPE OF SCALE FROM WHICH ITS MELODIC CONTENT IS DERIVED. MAJOR AND MINOR ARE REPRESENTED BY 1 AND 0, RESPECTIVELY.	INT
LIVENESS	DETECTS THE PRESENCE OF AN AUDIENCE IN THE RECORDING.	FLOAT
LOUDNESS	LOUDNESS OF THE TRACK IN DECIBELS (DB). VALUES TYPICALLY RANGE BETWEEN -60 AND 0 DB.	FLOAT
SPEECHINESS	DETECTS THE PRESENCE OF SPOKEN WORDS IN A TRACK.	FLOAT
TEMPO	ESTIMATED TEMPO OF THE TRACK IN BEATS PER MINUTE (BPM).	FLOAT
TIME SIGNATURE	AN ESTIMATED TIME SIGNATURE. THE TIME SIGNATURE (METER) IS A NOTATIONAL CONVENTION TO SPECIFY HOW MANY BEATS ARE IN EACH BAR (OR MEASURE). THE TIME SIGNATURE RANGES FROM 3 TO 7 INDICATING TIME SIGNATURES OF "3/4", TO "7/4".	INT
VALENCE	A MEASURE FROM 0.0 TO 1.0 DESCRIBING THE MUSICAL POSITIVENESS CONVEYED BY A TRACK.	FLOAT

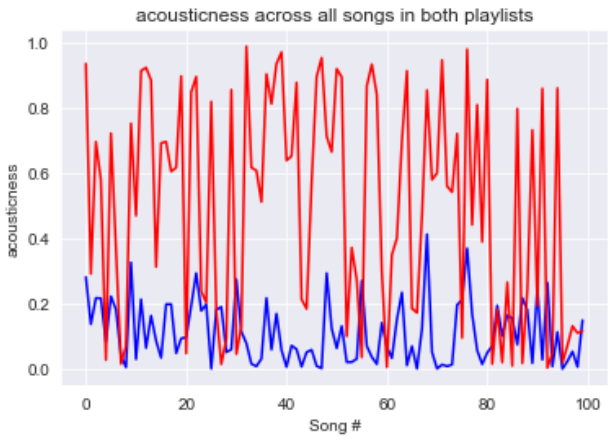

Figure 2. Acousticness values across all songs in the dataset. The red and blue lines correspond to happy and sad songs, respectively.

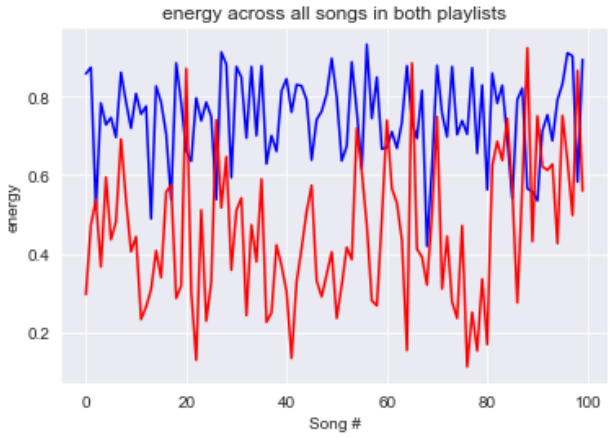

Figure 3. Energy values across all songs in the dataset. The red and blue lines correspond to happy and sad songs, respectively.

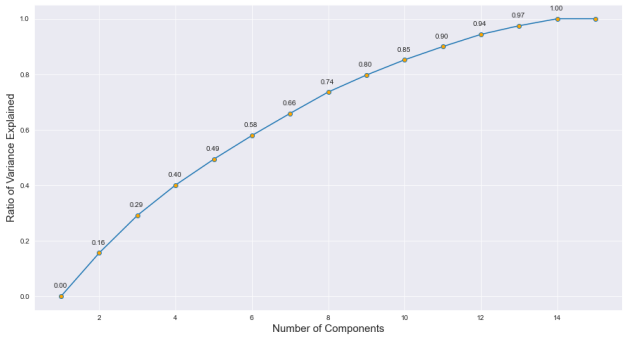

Figure 4. Total explained variance across all possible numbers of principal components (1-14).

the total variance explained by 1-14 principal components (shown in Figure 4) and looked for a notable decrease in slope, or an "elbow". However, after analyzing the plotted explained variances, it was found that there was no optimal number of principal components.

2.2. K-Nearest Neighbors

After fully exploring the dataset, Mahalingam and Zhang proceeded by using the original happy and sad song datasets to create combined datasets with reduced dimensionalities. Specifically, four datasets with dimensionalities of 2, 3, 4, and 5 were created. These reduced datasets were populated with the following input features: acousticness, energy, loudness, valence, and danceability. Each reduced dataset was split into a training and testing set (70:30).

Prior to training the K-Nearest Neighbors classifier, Mahalngam and Zhang selected four relevant hyperparameters for further tuning: *n_neighbors*, *p*, *weights*, and *metric*. The *n_neighbors* hyperparameter represents the k-value. Hyperparameter tuning was performed using k-fold cross-validation, with five folds.

Mahalingam and Zhang then trained a K-Nearest Neighbors classifier model using the optimal hyperparameters (with a k-value of 7) on the resulting training sets with 2, 3, 4, and 5 input features. The performance of the classifier was evaluated by comparing the model's predicted classification (denoted by a 0 or 1) to the actual classification in the test set and calculating the percent accuracy of these predictions.

2.3. Linear and Quadratic Discriminant Analysis

Mahalingam and Zhang proceeded by trying to use linear and quadratic discriminant analysis to classify the songs. Before training these respective models, hyperparameter tuning was executed for both models using k-fold cross-validation.

For the linear discriminant analysis model, the following parameters were calibrated: *solver*, *store_covariance*, *tol*, and *shrinkage*. It should be noted that the *store_covariance* and *tol* hyperparameters are only used for the SVD solver, and the *shrinkage* parameter is only used for the least-squares and eigen solvers.

For the quadratic discriminant analysis model, the following hyperparameters were tuned: *reg_param*, *store_covariance*, and *tol*.

After performing K-Nearest Neighbors classification, Mahalingam and Zhang used the same training and testing sets to perform linear and quadratic discriminant analysis, and the performances for these models were evaluated in the same manner as for the K-Nearest Neighbors model.

2.4. Logistic Regression

Prior to training the logistic regression model, Mahalingam and Zhang performed hyperparameter tuning using k-fold cross validation, with a k-value of 5. Mahalingam and Zhang selected the *penalty*, *C*, *solver*, and *max_iter* hyperparameters for tuning. Then, the logistic regression model was initialized with the selected hyperparameter values and trained on the reduced training sets and evaluated for accuracy using the same methods as prior models.

2.5. Decision Tree Classifier

Finally, Mahalingam and Zhang elected to use a decision tree classifier. Before any training was conducted, Mahalingam and Zhang performed five-fold cross-validation on the scaled dataset with all five input features to tune six important hyperparameters of the model. These selected hyperparameters were the following: *criterion*, *min_samples_leaf*, *min_impurity_decrease*, *splitter*, *max_depth*, and *ccp_alpha*. After tuning these parameters, Mahalingam and Zhang used them to initialize a new Decision Tree Classifier model. This model was then trained on each of the reduced and scaled training sets. Lastly, the performance was determined using the same previously established performing evaluation methods.

3. Results

3.1. K-Nearest Neighbors

Using a k-value of 7, the k-nearest neighbors model performed best on the datasets with reduced dimensionalities of 4 and 5, with a peak prediction accuracy of 90% (shown in Figure 5). As expected, the model performed worst on the dataset with a dimensonality of 2, with a performance of 85%.

3.2. Linear and Quadratic Discriminant Analysis

Both the linear and quadratic discriminant analysis models reached a performance of 90% on the dataset with 5 input features, as shown in Figures 6 and 7. The linear discriminant analysis model performed best on the dataset with all five audio features. Unlike all the other classification models, the quadratic discriminant analysis model did not perform best of the dataset with all five features. Instead, this model reached its peak prediction accuracy of 91.67% for the dataset with a dimensionality of 4. Both models had a minimum performance of 80%.

3.3. Logistic Regression

As can be observed from Figure 8, the logistic regression model performed best on the dataset with a dimensionality of 5, with a prediction accuracy of about 91.67%. When compared with the other models, the logistic regression model had the best performance on the dataset with all five audio features and came closest to our target performance of roughly 93%.

3.4. Decision Tree Classifier

After hyperparameter tuning, the decision tree classifier model was able to reach performances of 76.67%, 85%, 86.67%, and 86.67% for the reduced datasets with 2, 3, 4, and 5 audio features, respectively (shown in Figure 9). This model had noticeably lower prediction accuracies than the rest of the classification models used in this project. Even after hyperparameter tuning, Mahalingam and Zhang were unable to achieve a performance of 90% with the decision tree classifier.

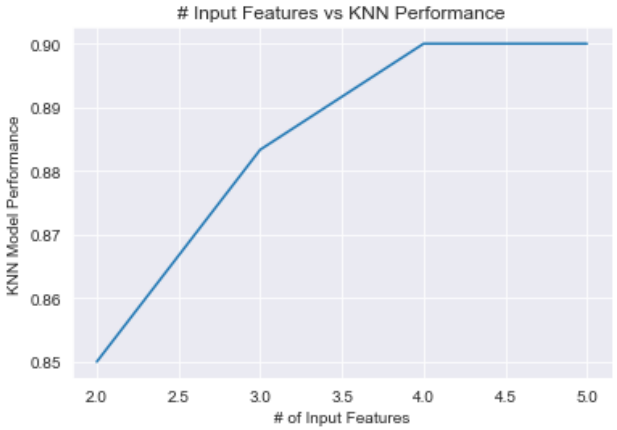

Figure 5. K-nearest neighbors model performance across all four scaled datasets.

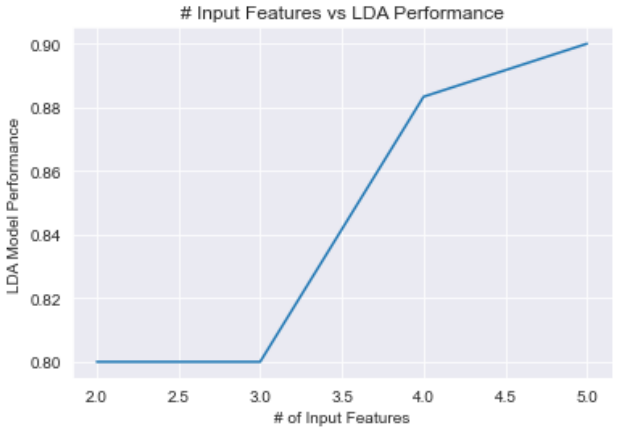

Figure 6. Linear discriminant analysis model performance across all four scaled datasets.

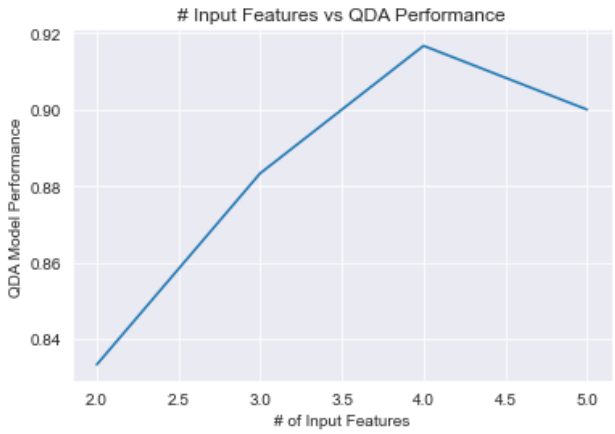

Figure 7. Quadratic discriminant analysis model performance across all four scaled datasets.

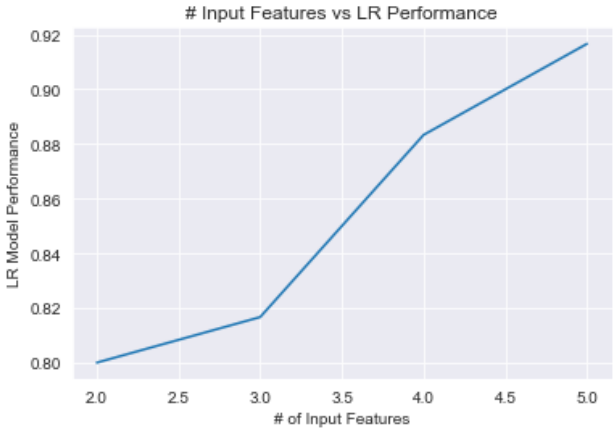

Figure 8. Logistic regression model performance across all four scaled datasets.

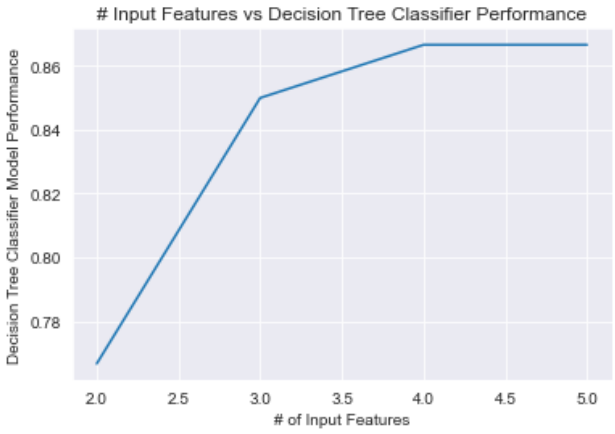

Figure 9. Decision tree classifier model performance across all four scaled datasets.

4. Discussion and Conclusions

4.1. Discussion and Conclusions

Three main challenges arose while completing this project. The first challenge that arose during this project was understanding past methods. The Spotify API is well developed, extensive, and frequently used. As such, there were many variations of similar past projects conducted and documented. Mahalingam and Zhang spent considerable time understanding successful past models, and ensuring the credibility of previous methods, to arrive at a target range of 87.0-99.7%.

The second challenge that arose during this project was reducing the dimensionality of the original combined dataset. The Spotify API provided more input features than were needed to perform binary classification. For example, song title and artist was a feature provided by the Spotify API. However, intuitively, this feature would not be useful in training a model as many artists produce both sad and happy songs. There were several features Mahalingam and Zhang eliminated using some human intuition. After this initial feature reduction process, the pair performed Principal Component Analysis to determine how many features would account for an acceptable amount of variance.

The last challenge encountered during this project was finding the optimal hyperparameter values for each classification model. Mahalingam and Zhang were able to address this issue using k-fold cross-validation. For each model, Mahalingam and Zhang loaded predefined hyperparameter values into a parameter grid, then cycled through and evaluated all possible value combinations in the grid using k-fold cross-validation. The combination of hyperparameter values with the best performance was determined to be the ”optimal” set of hyperparameter values and was then used to initialize each model. In terms of possible future work, Mahalingam and Zhang plan to add additional playlists (with different song moods) to the dataset.

As was mentioned in Section 2.1, the dataset used in this project does not exhibit perfect separability. As such, Mahalingam and Zhang were initially skeptical that logistic regression would be able to compete performance-wise with the k-nearest neighbors model. However, the logistic regression model had the best performance on the scaled dataset with all five audio features, with a prediction accuracy of about 91.67%. As was briefly discussed in the introduction, one main concern that Mahalingam and Zhang had when beginning this project was the potential for overfitting. During their review of similar projects, Mahalingam and Zhang often came across performances of 99% or higher, which they found to be unusually high. Logistic regression is generally very resistant to overfitting as a classification method and is less sensitive to modeling assumptions than methods like

Gaussian Discriminant Analysis. These qualities, coupled with its reliably high performance, have led Mahalingam and Zhang to select the logistic regression model as their ideal choice of classifier for future work on this project.

4.2. Next Steps

Possible avenues for future work include:

- Use more robust cross validation to tune model hyperparameters.
- Include additional playlists (with different "song moods") to the dataset.
- Explore additional machine learning methods.
- Training on more features and more data.

One practical application of this research would be to create a front-end interface that allows Spotify users to extend playlists with a small number of songs.

Acknowledgements

The contributions of each team member were as follows:

Team:

- Read research, define project, and train models
- Create presentation
- Write final paper

Mahalingam:

- Data exploration
- Create plots for presentation and paper

Zhang:

- Set up Spotify configurations
- Preprocess data
- Define ML methods

References

Afrulbasha, S. Ml step-by-step: Using knn algorithm to classify spotify songs into playlists, Dec 2020. URL https://towardsdatascience.com/ml-step-by-step-using-knn-algorithm-to-classify-spotify-songs-into-playlists-8c7892428371.

Bharadwaj, A. Organizing my spotify playlists with data science, Jul 2020. URL https://towardsdatascience.com/organizing-my-spotify-playlists-with-data-science-9a528110319.

Brownlee, J. Linear discriminant analysis with python, Sep 2020. URL https://machinelearningmastery.com/linear-discriminant-analysis-with-python/.

MIDiA Research. Global streaming music subscribers worldwide from 2015 to 1st quarter 2021 (in millions), Jun 2021. URL https://www.statista.com/statistics/669113/number-music-streaming-subscribers/.

Norman, L. How to find new songs on spotify using machine learning, Apr 2021. URL https://medium.com/geekculture/how-to-find-new-songs-on-spotify-using-machine-learning-d99bd8855a18.

A. Navigating the Source Code

This section is analogous to the README for a git repo. Follow the filepath below to see the main code in attached repository:

ml-project/ml-project.ipynb

Main Sections in Final Code:

1. Imports and Configurations
2. Exploratory Data Analysis
3. Performance

The code is well commented to aid the comprehension of the code reviewer. Necessary installs are commented in the main code file.

Follow the below filepath to see the code file prior to the presentation checkpoint:

ml-project/.ipynb_checkpoints/ml-project-checkpoint.ipynb

All other files comprise the data used for this project.