Spotify Dataset Song Recommender in Python

I. Administrative

A. Team name: Group 48: Spotifython

B. Team Members:

Constantinos Barmpouris
Henry Harborne
Austin Spangler

C. Link to Github repo:

https://github.com/henryharborne/Cop35303B.git

D. Link to Video demo:

https://youtu.be/J_vAeg0kEYI

II. Extended and Refined Proposal

A. Problem and Motivation

Within the scope of our project we attempt to increase music listeners' enjoyment with a program that can cultivate a curated list of songs using a recommendation based off of similarity metrics in relation to user input. The extent of this problem is not as shallow as it may initially appear, as this application perhaps has an even stronger appeal to corporations and music providers, while also retaining its ability to improve and extend individuals' music enjoyment. For larger corporations, consumer retention is critical to earning revenue and remaining operational. Further, a necessary component to keeping purchasing consumers is constantly providing new and fresh music that said individuals would enjoy. This algorithm would, based on a single individual song, recommend up to 3 near identical songs per algorithm based on algorithmic similarity checkers using songs' quantitative features. Applied to a larger scale, corporations can solve perhaps the most critical aspect of their business, which is to keep the user engaged and constantly enjoying new music. For individuals, wanting new music that they would enjoy is something people, in general, are constantly feeling. Most people resort to random shuffling or asking friends, but we are providing a much more efficient path. For consumers, and perhaps even more importantly corporations, providing curated music is a necessity for not only enjoyment but business sustainability.

B. Features Implemented

Our project focused on the implementation of two algorithms whose goal was to recommend the closest songs to a user's inputted song based on a distance metric. Thus, the features we implemented were all based on this core idea of our project's goal. First, we implemented user friendly text with a short description on the goal of our project. Next, users were prompted to input a song. Based on the text the user provided, if the user's choice was invalid, our program prompted them to enter a valid song choice. If instead the user selected a valid song, we provided them with all the possible songs that included the string they inputted. Thus, based on which of the options they selected, we ran our two algorithms, KNN and Naive Bayes Classifier, to find the most similar songs to theirs. Each algorithm outputted 3 similar songs with user friendly text intended to improve user experience. Furthermore, we found the most similar song using Euclidean distance as our metric in order to give the user a starting point in their quest for new music. Finally, we plotted a plethora of graphs that compared

our two algorithms using a variety of distance metrics in order to produce a more holistic view of each algorithm's performance.

C. Description of Data

The data is an extensive and defined list of song names, authors, and a breakdown of said songs into the following categories. Songs are defined with a rank in each of : ‘danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness', 'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo', and 'time_signature'. Besides these more particular stats, songs are organized by their track id, author, duration, popularity, and of course the album and song name. There are over 125 genres in this dataset, and audio features such as loudness, instrumentalness, and valence are included as columns. The dataset has approximately 114,000 songs, each including columns with the aforementioned features.

D. Tools/Languages/APIs/Libraries used:

A multitude of libraries and tools were imported and used. For manipulation of dataframes, we used pandas. For specific functions relating to math and data analysis, we used numpy. For standardizing our data, we used sklearn’s preprocessing module, specifically StandardScaler. For plotting graphs that were used to display the distance comparisons of our algorithms, we used matplotlib. Since we don’t have targets, we used kMeans to create clusters and labels that enabled Naive Bayes Classifier to select recommended songs among the songs in their most similar cluster. Finally, for comparing our two algorithms, we needed various distance metrics, so we imported pairwise_distances, manhattan_distances, and euclidean_distances from sklearn’s metric module.

E. Algorithms implemented:

We began our implementation by performing standard data exploratory analysis and data preprocessing in order to gain an idea of how to craft the implementation of our algorithms and to clean our data in a way that allowed us to actually implement them. Some of these steps included manipulating our dataset so that it did not include duplicates, null values, or useless columns. Furthermore, we standardized our data to improve accuracy and reduce inherent bias for features that had higher ranges of values.

The algorithms utilized were the K Nearest Neighbors algorithm and the Naive Bayes Classifier algorithm. KNN was implemented using cosine similarity as its metric to locate the K closest feature vectors to the user inputted feature vector. Each feature vector is basically a Pandas series including information from each column that represents the features that make up each song. Using these feature vectors, we implemented our own cosine similarity function within our KNN algorithm in order to have a quantitative measure of the similarity of songs. Finally, we selected the first K songs after sorting them based off of their cosine similarity to the user input.

Our Naive Bayes algorithm is more unorthodox and more complex. We implemented an entire Naive Bayes class in order to make the implementation of our algorithm from ground zero. We used kMeans outside of our Naive Bayes algorithm in order to provide cluster centers that could be used by Naive Bayes to decide

which cluster the most similar songs should be placed. Then, we used these classifications to detect the most similar songs to the user input utilizing only the songs within the cluster of the user input presented to us by our Naive Bayes algorithm. The Naive Bayes class itself works by utilizing a probability metric, assuming the effects of each individual feature is independent of others, which could be a flaw in our project ideology that may be decreasing our accuracy. Finally, we used a variety of distance metrics from sklearn's metric module in order to conduct a comparison between the results of each algorithm. Our results were not entirely conclusive. Different distance metrics viewed the similarity of our recommendations to user input with varying levels of success. Certain distance metrics are unideal for high dimensional vectors, others primarily focus on classification, and thus there are a number of inconsistencies across these metrics. Thus, we decided to include a multitude of metrics to provide a more holistic display of the performance of our algorithms.

F. Additional Data Structures/Algorithms used:

In addition to the KNN and Naive Bayes algorithms, we use a wide variety of vector representations to store data temporarily. Furthermore, we used sorting algorithms to quickly sort data. KMeans was used to help train the Naive Bayes classifier. Distance metric algorithms were implemented through Python libraries.

G. Distribution of Responsibility and Roles: Who did what?

Defining a clear cut distribution of roles is difficult in this project, though some distinctions can be made. Every member participated heavily in all aspects of the project. From a general perspective, Henry had a larger role in overall architecture, implementation, and data preprocessing. Constantinos played a larger role in the theoretical aspects of the algorithms, such as the statistical methodologies behind the algorithms and data exploratory analysis.. Austin streamlined the comparison of algorithms by providing various distance metrics in order to display the efficiency of each algorithm.

III. Analysis

A. Any changes the group made after the proposal? The rationale behind the changes.

Our group made a few changes after the proposal. After discussing with TA Maximilan Meyer, we realized the implementation of algorithms such as tSNE were too complicated for us to be able to do. We decided to change our algorithm choices to KNN and Naive Bayes' classifier. During our implementation of these algorithms, we realized they were very different approaches and thus required a lot of time for implementation. Thus, we decided it was not feasible for us to also create the website we were planning to make for our code to have a nice front end display. We realized our ambitions may have been too high, and, in an effort to ground ourselves, decided to create simpler python executable functionality with output. We tried to make this functionality within our notebook as user friendly as possible.

B. Big O worst case time complexity analysis of the major functions/features you implemented

For the KNN algorithm, there are two functions to complete the algorithm. The cosineSim helper function has a time complexity of O(d * n), where d is the vector dimensionality and n is the number of songs in the dataset. Then the knnRecommend requires sorting, which is O(n log n). Overall, to implement the KNN algorithm, the final time complexity O(d * n + n log n).

For Naive Bayes, the time complexity is calculated by combining the distance calculation, which is O(n * d), where n is the number of songs and d is the vector dimensionality. Furthermore, sorting is involved in Naive Bayes, so the time complexity is O(n log n), where n is still the number of songs.

For running the entire program, it is dominated by the time complexity of the KNN and Naive Bayes algorithms. Importing libraries and data visualization can be disregarded when simplifying the time complexity, which is O(n * d + n log n), where n is the number of songs, and d is the vector dimensionality.

IV. Reflection

A. As a group, how was the overall experience for the project?

As a group, our overall experience for the project was wonderful. We had vastly different experience levels with Machine Learning algorithms, so it was really interesting for us to educate each other and learn from each other on advanced programming topics. Furthermore, we were very engaged in developing our code due to the impact of our problem statement. We are all avid music fans, and we recognize our problem statement is a big issue for all of us. Thus, we were very motivated to develop a program that can improve our listening experience. We hope that an advanced, better implementation of our idea can be implemented within Spotify one day as we believe in the potential of our idea to transform users' listening experience.

B. Did you have any challenges? If so, describe.

We had quite a few challenges. Firstly, we quickly realized that the implementation of Machine Learning algorithms is very difficult. Thus, we had to select algorithms that would be viable for our experience levels and coding abilities. Furthermore, it was challenging to keep everyone on the same page as we coded all together. Thus, our varying experience and knowledge levels meant we had to spend a lot of time in making sure we all maintain a similar level of understanding over every component of our project.

C. If you were to start once again as a group, any changes you would make to the project and/or workflow?

If we were to start once again as a group, we would certainly make changes to our project and our workflow. Firstly, we would probably focus on algorithms that were covered in this data structures class so that all members would have similar levels of familiarity with the algorithms. Also, we would probably slightly alter our workflow in order to use github more efficiently. Unfortunately, because we decided to code everything together, we ended up sending

code to each other through Discord often because it saved time. However, there were a couple moments where our commits were problematic and at one point our entire notebook would fail to open. At that point, the anxiety of having to restart was ever present, but, thankfully, we were able to override changes and our project remained safe.

D. Comment on what each of the members learned through this process.

Some of the most important unrealized parts of our team formation was the vast diversity in our academic backgrounds. Constantinos is a Math and Data Science major, Henry is a Computer Science major, and Austin is a Business Admin major and Computer Science minor. Thus, we all had different perspectives on our approach to solving our problem statement. This helped us tremendously when making tough decisions. For example, Austin had great insight into translating technical components of our project into deliverables that a board could appreciate. Henry was able to use his programming background to determine inventive ways to code our own implementations of algorithms that initially appeared extremely difficult. Constantinos was able to use his statistical knowledge to determine what statistical approaches were beneficial to our end goal of producing relevant song recommendations. Thus, with our diverse skill sets, we were able to educate each other and learn from one another in a way that was critical to our team's success. Austin's biggest takeaway was in relation to the underlying methods that make up Machine Learning Algorithms and how significant data preprocessing is to the success of a project. Henry learned about the statistics behind Machine Learning algorithms that he already had experience applying. Constantinos was educated on the structuring of code so that it is implemented in a way that ensures output that is digestible by users.

V. References

[1] https://pandas.pydata.org/

[2]https://scikit-learn.org/1.5/modules/generated/sklearn.cluster.KMeans.html

[3] https://numpy.org/

[4] https://matplotlib.org/

[5]https://scikit-learn.org/dev/modules/generated/sklearn.preprocessing.StandardScaler.html

[6]https://scikit-learn.org/1.5/modules/model_evaluation.html

[7] CIS4930 - Intro to Machine Learning

[8] EEL4930 - Applied Machine Learning Systems

[9] QMB3302 - Business Analytics & AI

[10] STA4273 - Statistical Computing in R