Recommending in Style: Comparative Study of Machine Learning Techniques for a Music Recommendation System

Arya Chethikattil[1], **Greeshma Challaram**[1], **Micah Ban**[1], **Nhut Nguyen**[1], **and Asish Addanki**[1]

[1]Department of Computer Science, Arizona State University, Tempe, AZ
[*]*{aaddanki, nmnguye5, achethi1, mtban, gchallar}@asu.edu*

Abstract

Music recommendation system has become an increasingly prevalent topic in the recent years because technology has advanced and people prefer to listen to music through applications like Spotify, Apple Music, Youtube, etc. With online music applications, users can discover new music that matches their taste, discover new songs based on what they like, and music applications can target their audience better. In this project, we are attempting to build a model that serves as a music recommendation system using different machine learning techniques to categorize music genres and curate a personalized playlist. We will be utilizing the Spotify Tracks dataset to analyze features such acousticness, danceability, instrumentalness, loudness, tempo, energy, and more to give recommendations based on a user-generated playlist. We will be implementing an assortment of clustering algorithms and the k-nearest neighbor machine learning algorithm in this project. To measure the benchmarks of our model, various performance metrics will be utilized. This report gives an overview of the project introduction, description, approaches, results, and conclusion.

1 Introduction

The problem of improving current music recommendation systems is one that has grown in popularity in recent years. In this section, we discuss the motivation behind the problem, the benefits of improving music recommendation systems, and the scope of the problem. We also explore in this section technical concepts relevant to our solution and surrounding research in this field including existing best methods and the most difficult challenges faced by others.

1.1 Motivation

In recent years, music platforms have become increasingly more popular due to music being mostly digital. Many machine learning algorithms that recommend songs have been a crucial aspect of using the most popular music platforms, such as Spotify [1]. There are several motivations for our project, first of all, personalization allows users to explore different artists similar to their current preferences and acts as a filter. On top of that, it allows users to listen in the background as they do not always have to queue songs or actively keep queuing songs. Although users are content with current methods, there are still existing challenges that if overcome, would improve these algorithms and user satisfaction greatly [1]. Additionally, recommendation systems have shown increase in engagement in the application and boost user retention. Lastly, a good recommendation system can attract new user, increase in advertisements, and ultimately enhance revenue. It's important that we

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

build a robust recommendation system that uses state-of-the-art algorithms on a big dataset to provide good recommendations.

1.2 Problem Statement

The evolution of music streaming platforms has intensified the quest for optimal song recommendation systems fueled by the desire to cater to diverse user preferences and elevate user experience. This project aims to implement multiple machine learning models to efficiently recommend songs based on a user-created playlist through strategic feature engineering maneuvers, such as principal component analysis (PCA) and recursive feature elimination (RFE). Additionally, exploring the application of multiple clustering algorithms to integrate with the k-nearest neighbors (kNN) machine learning model will enable the generation of tailored song suggestions, effectively addressing existing challenges in music recommendation systems.

1.3 Technical Background

This project incorporates an unsupervised learning approach because attributes of songs in a given playlist are immensely diverse such that generating a comprehensive labeled dataset based on specific features for a supervised learning approach can lead to a poor performance in a manner of recommending songs the user would not prefer to include in that playlist. Most notably, clustering algorithms are prominently utilized in this project to cluster songs into groups so that songs within each group are more similar to each other and different from songs in other groups. This similarity could be based on factors like genre, artist, tempo, mood, etc. Thus, simplifying the process of song recommendation to recommend songs that are the closest to songs in the provided playlist. There were two primary clustering algorithms implemented: K-means, and Balanced Iterative Reduced and Clustering using Hierarchies (BIRCH).

K-means clustering is an unsupervised machine learning algorithm that groups unlabeled datasets into distinct clusters. K-means clustering operates by iteratively assigning data points to the nearest cluster centroid and updating centroids based on the mean of the assigned points [2]. K-means clustering can effectively handle large datasets and is computationally efficient, making it suitable for real-time recommendation systems.

BIRCH, or Balanced Iterative Reducing and Clustering using Hierarchies, is a clustering algorithm that clusters a reduced dataset that encompasses the majority of information from the original dataset. There are two key characteristics in the BIRCH algorithm: Clustering Feature (CF), and the CF tree. A clustering feature is a three-dimensional vector that summarizes information about clusters of objects from the original dataset. A CF tree is a height-balanced tree that is used to store clustering features for hierarchical clustering where each leaf node stores clustering features for a fixed number of clusters. Additionally, there are two primary stages in the BIRCH clustering algorithm. The first stage consists of analyzing the dataset to construct an initial CF tree which can be viewed as a multi-level compression of the dataset that attempts to retain the inherent clustering structure of the original dataset. The algorithm then incorporates a selected clustering method to cluster the leaf nodes of the CF tree [3]. The BIRCH clustering algorithm was selected as a potential clustering algorithm because the BIRCH clustering algorithm can process large datasets such as the provided dataset more efficiently and utilizes even less computational resources than the K-means clustering algorithm.

In addition to clustering algorithms, our project incorporated the use of K-nearest Neighbors (KNN) classifier as part of our recommendation system. The KNN Classifier is a supervised machine-learning method employed to tackle classification and regression problems [4]. Unlike clustering algorithms, which group data points into clusters based on similarity, the KNN classifier predicts the class or label of a data point based on the majority class of its nearest neighbors in the feature space. By leveraging features of songs within a given playlist, such as genre, artist, mood, and tempo, the KNN classifier can identify the most similar songs in the dataset based on these attributes. This allows the classifier to recommend songs that align closely with the user's preferences and characteristics of the songs in the original playlist. KNN classifier also adapts dynamically to the local structure of the data, making it well-suited for scenarios where the boundaries between classes may be fluid or nonlinear. The combination of unsupervised clustering for grouping similar songs and supervised

classification for fine-grained recommendation enhances the accuracy, relevance, and, ultimately, user satisfaction of the music recommendations generated by our application.

1.4 Related Work

There are 2 main methods for filtering: content-based and collaborative filtering [1]. The recommended and most favorable method is a hybrid approach of both [1].

1.4.1 Content Based Filtering

Content-based filtering refers to recommending songs based off existing data of user's preferences for songs [5]. Using feature extraction and cluster analysis, we can base our preferences from these [5]. The Cold Start Problem is an issue with this method however to a higher degree [5].

There are two subcategories of music data used with filtering: metadata and audio content [5]. The first refers to the labels and lyrics associated with a song while the latter refers to audio features of the songs [1]. An example of metadata could be genre, reviews, or artist while audio content could refer to pace, instruments, and rhythm [5].

1.4.2 Collaborative Filtering

On the other hand, collaborative filtering matches users to other users based on histories [5]. By analyzing historical interactions between users and items, collaborative filtering identifies patterns and similarities among users' preferences. As a result, recommendations for songs are based off what the matched user listens to as well [5].

1.4.3 Hybrid Filtering

The hybrid method, which utilizes both methods of filtering, is preferred as it addresses the cold start problem, which will be discussed in the next section, by using collaborative filtering for users with no data. It also has the benefits of using content based filtering for users with data [5].

1.4.4 Challenges and Solutions

In related work, there were a few common challenges discussed. One problem discussed is the "Cold Start Problem", referring to the lack of data initially, and what to recommend in that scenario [1]. One solution is to allow the user to choose a few genres, artists, or songs when they make an account, and start recommending based off those selections [1].

Another common challenge is the "automatic playlist continuation" problem which describes the issues related with continuously having to add songs that are similar to the original songs the user liked [1]. Common solutions discussed are using other playlists created by the user to base future recommendations on [1]. The drawbacks of this method and the main issue to address here are the importance of sequences of songs and user moods, and how best to transition to a different mood [1]. Songs, compared to books or movies, are unique in this way and require a different method of approach [5].

2 Exploratory Data Analysis

2.1 Dataset and Pre-Processing

With great consideration, our group has decided to use another dataset from Kaggle that consists of audio features retrieved from the Spotify Web API of Spotify tracks instead of the previous partitioned dataset from the Spotify Million Playlist Challenge [6]. Primarily, this dataset contained an additional attribute of the music genre of each track which was utilized as a supplementary proxy that verifies the quality of the clusters produced from the clustering algorithms and the accuracy of the KNN classifier.

The dataset comprises 114,000 Spotify tracks that span over a diverse range of 125 unique genres [7]. There are 20 attributes in the dataset, the first four attributes detailing fundamental information of the song such as 'track_id', 'artists', 'album_name', and 'track_name'. The remaining 16 attributes are

predominantly utilized in the clustering algorithms and the KNN classifier to precisely recommend songs from a given playlist. The attributes are: 'popularity', an integer value from 0 to 100 that determines the popularity of the track based on the amount of occurrences the track has been listened to and how recent the track was listened to; 'duration_ms', the length of the track in milliseconds; 'explicit', a binary value that indicates if the track contains explicit lyrics; 'danceability', a float value ranging from 0.0 to 1.0 that describes how suitable a track is for dancing based on multiple factors such as tempo, rhythm stability, and beat stability; 'energy', a float value ranging from 0.0 to 1.0 that measures the intensity of a track according to several factors like dynamic range, onset rate, and timbre, which is the perceived quality of a music note; 'key', an integer value that corresponds to the key of the track using standard Pitch class notation; 'loudness', the overall volume of the track measured in decibels; 'mode', a binary value indicating the modality (major or minor) of the track which is the type of scale the track's melodic content is derived from, 'speechiness', a float value ranging from 0.0 to 1.0 that determines the presence of spoken words within a track; 'acousticness', a float value ranging from 0.0 to 1.0 that determines whether the track is acoustic; 'instrumentalness', a float value ranging from 0.0 to 1.0 that determines if the track contains any vocals; 'liveness', a float value that determines the presence of a live audience in the track; 'valence', a float value ranging from 0.0 to 1.0 that measures the overall positiveness of a track; 'tempo', the estimated pace of a track, which derives from the average beat duration measured in beats per minute (BPM); 'time_signature', an integer value from three to seven that entails the approximate time signature that specifies the number of beats are in each measure; and 'track_genre', the music genre of the track.

In the data pre-processing phase, the dataset was initially accessed, and a duplicate copy was generated to ensure data integrity. Subsequent to this, a meticulous examination ensued, aimed at identifying and resolving any instances of null values and duplicate records within the dataset. This rigorous process was undertaken to refine the dataset, thereby laying a robust foundation for subsequent feature engineering endeavors. The elimination of identified null values and duplicate entries served to enhance the dataset's quality, preparing it for integration into the K-means clustering algorithm. Notably, this strategic refinement process led to a reduction in dataset size from 114,000 to 89,379 rows, without compromising data integrity. This reduction further underscores the efficiency and effectiveness of the pre-processing efforts.

In the dataset, the datatype of the 'explicit' column was modified from 'truefalse' to '1' and '0' to enable numeric comparison. Subsequently, the Min-Max Scaler was applied to ensure comparability across various features essential for clustering. Specifically, the columns 'popularity', 'duration_ms', 'danceability', 'energy', 'key', 'loudness', 'speechiness', 'acousticness', 'instrumentalness', 'valence', and 'tempo' were normalized through fitting and transformation processes, enhancing their suitability for clustering analyses.

Additionally, outliers were identified for each column that was normalized and subsequently removed using the Interquartile Range (IQR) method and the 'remove_outliers_iqr' function. The index was then reset for the existing columns, resulting in a dataset size reduction to 55,082 rows. Furthermore, for the existing 'track_genre' column, a label encoder function was employed to assign labels to all existing genres. Subsequently, upon creation of the 'genre_label' column, a copy of the dataset was made, and the 'track_genre' column was dropped, ensuring streamlined data preparation for further analysis.

2.2 Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE)

In this project, we are looking at machine learning techniques such as principal component analysis and recurisve feature elimination to do data preprocessing and reduce the dimensionality of features. Since our spotify dataset has many features, it's important to simplify the model, speed up computation, and avoid causing the performance of our model to decrease. How PCA works can be broken down into 5 steps: standardization, covariance matrix computation, eigenvalue decomposition, create a feature vector, and recast the data along principal components axes. In machine learning, PCA is often used as a data preprocessing step to enhance the performance of models, so we looked to incorporate it into our project. However, there are some limitations such that PCA only captures the linear combination of features, and does not effectively capture the non-linear features.

Alternatively, we also opt for recursive feature eliminations (RFE) as a data preprocessing technique that fits a model and removes the weakest feature (or features) until the specified number of features is satisfied. Since our dataset is big and has a lot of features, RFE is very versatile because we can select

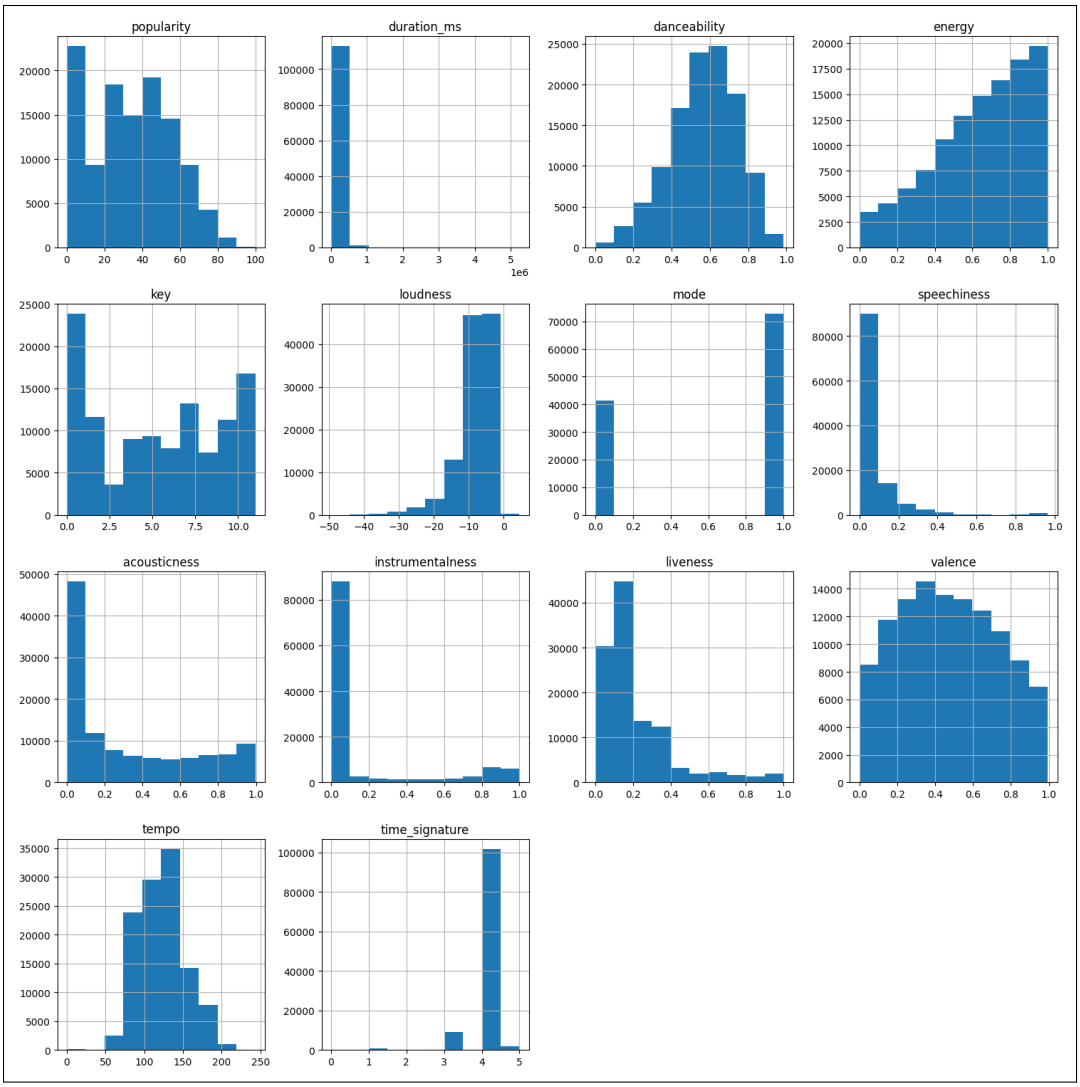

Figure 1: Histograms illustrating the distribution of the audio features for the tracks.

how many features we want based on our dataset. This is a big part of our recommendation system because we want to select only the relevant features so that our model can perform well and mitigate over fitting. RFE works in the follwing steps: Initial model fitting, ranking of features, elimination of least important features, repeat, and select the best features. RFE offers many benefits such as reducing the model complexity and increasing performance by removing irrelevant features. Another benefit of RFE is preventing overfitting. Overall, RFE is easy to use, implement, and it is a powerful technique in machine learning, thus we took advantage of this feature in our recommendation system.

3 Approach

Our team decided to use a content based filtering approach for this model given the dataset contents. Statistics surrounding the user's song choice were present, but we were not given much information about user's information to incorporate collaborative filtering.

3.1 Clustering Songs by Genre

After constructing a reduced dataset through data pre-processing and RFE, our group was able to implement multiple clustering algorithms to cluster each of the songs by an approximate genre

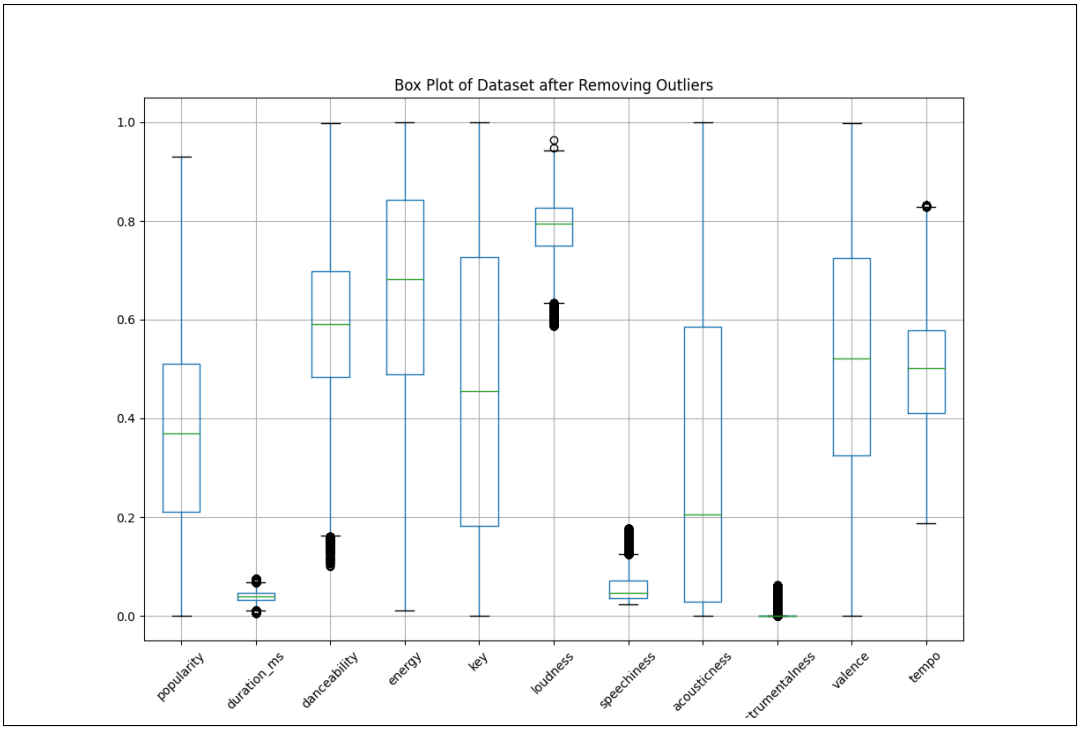

Figure 2: A box plot to illustrate the distribution among features in the dataset after removing outliers.

category. Firstly, the sum of squared distances between the assigned cluster centroid and its respective points which is known as within-cluster sum of squares or WCSS were calculated for a number of clusters ranging from one to eleven. After presenting the WCSS and the number of clusters on a line graph, the elbow method is performed by finding the number of clusters where the line sharply decreases to determine the optimal number of clusters [8]. According to the line graph illustrated in Figure 3, we concluded that the optimal number of clusters is four.

Subsequently, each of the clustering algorithms were then evaluated by a silhouette score after being deployed on the dataset. A silhouette score is a metric to measure the quality of clusters produced by clustering algorithms. With a range from -1 to 1, silhouette scores closer to 1 indicate that data points are classified in the correct clusters and the clusters are distinctly separated, scores near 0 indicate overlapping clusters or data points corresponding to multiple clusters, and scores near -1 indicate that the data points are classified in incorrect clusters [8]. As a result, our group selected the K-means clustering algorithm as the primary clustering algorithm because the K-means clustering algorithm generated the highest silhouette score. Further discussion of the silhouette score and the scores of the clustering algorithms are discussed in the 'Results' section.

From the K-means clustering algorithm, our group analyzed the features of the cluster centroids to classify each centroid as a comprehensive music genre category. There were noticeable insights from the dataset that enabled the creation of the overall genre categories such as a centroid receiving significantly higher or lower values in specific features. The examination of the features of the cluster centroids prompted the construction of approximate genre categories: 'Acoustic/Classical'; 'EDM', or Electronic Dance Music; 'Pop'; and 'Rock/Metal'. Through PCA, or principal component analysis, the dataset's dimensions were effectively reduced to three which enabled the creation of the 'x', 'y', and 'z' axes for the scatter plot visualization. This reduction not only simplified the data representation but also demonstrated a clearer understanding of the underlying patterns and relationships within the dataset. Successively, our group was able to visualize the clusters of tracks according to their respective genre category using the K-means clustering algorithm as illustrated in Figure 3.

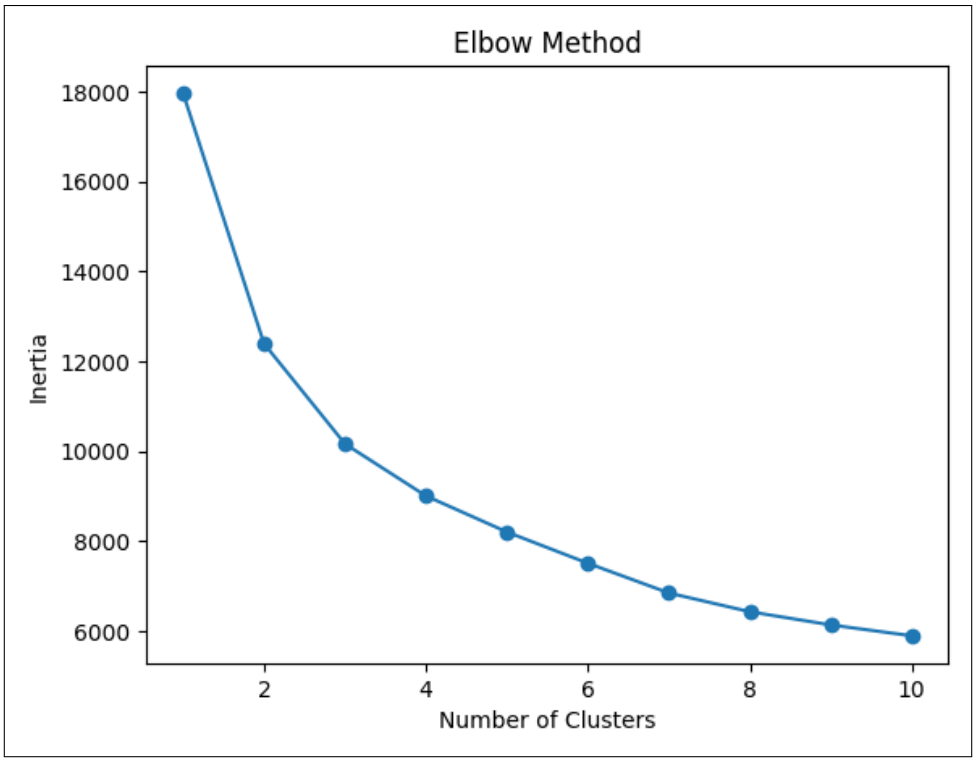

Figure 3: A line graph to find the optimal number of clusters using the elbow method.

3.2 KNN Classification

After analyzing the performance of the clustering algorithms and selecting the most optimal algorithm, the K-means clustering algorithm was then incorporated into the KNN classification algorithm to not only verify the accuracy of the clustering algorithm, but also check that recommended songs are within the same cluster as songs from the given playlist. The reduced dataset that was applied to the two clustering algorithms is split into training and testing datasets for the KNN classifier with a 80/20 ratio. The KNN classifier was evaluated using cross-validation across different values of k (5, 10, and 25), where k represents the number of nearest neighbors used to determine the classification of a given track. Subsequently, the variations of the KNN classifier yielded similar accuracy scores: 0.759, 0.756, and 0.737 respectively. Based on the training results, our group noticed that the KNN classifier with lower values of k performed greater than the classifier with higher values of k. As a result, the KNN classifier with a value of k equaling to 5 resulted in an accuracy of 0.767 from the testing dataset.

Table 1: Accuracy Scores of KNN Classifiers

k	Accuracy
5	75.9%
10	75.6%
25	73.7%

3.3 Song Recommendation from Playlist

Lastly, the integrated machine learning model of the K-means clustering and KNN classification algorithms were implemented to recommend new songs based on an input playlist of songs. The selected attributes for the K-means clustering algorithm were the input playlist of songs were first retrieved and calculated to derive a singular data point that comprised the average values of the songs within the playlist. The data point was processed by the integrated model, which involved computing

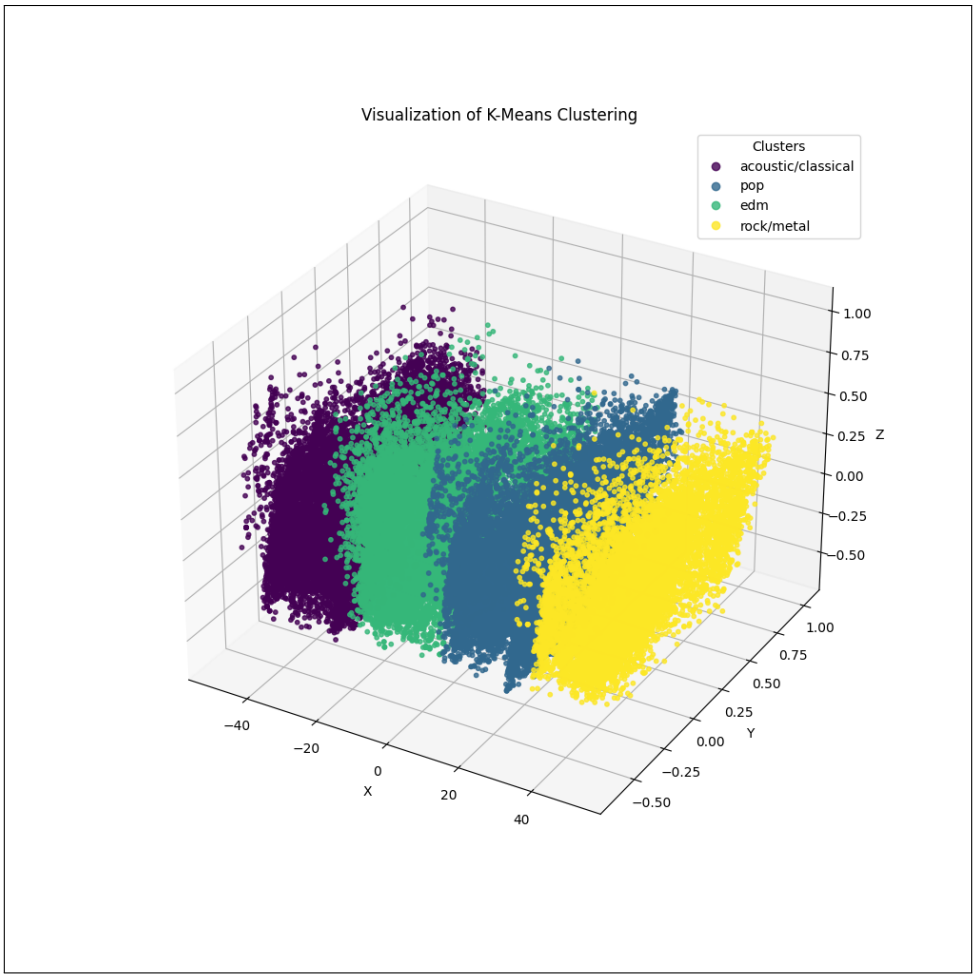

Figure 4: A three-dimensional scatter plot displaying the clusters of tracks to their approximate genre.

cosine similarities and Euclidean distances between the inputted data point and others in the dataset. As a result, the integrated model generated five recommended songs that are the closest to the inputted data point.

4 Results

As mentioned previously, we employed two different clustering algorithms, namely k-means clustering and BIRCH clustering, to categorize songs into distinct genres. To evaluate the effectiveness of the algorithms, we utilized silhouette scores. Our analysis revealed that the k-means clustering algorithms yielded a silhouette score of 0.59, indicating well-defined and distinct clusters. This suggests that the data points were appropriately classified into cohesive clusters with minimal overlap and incorrect clustering. In contrast, the BIRCH clustering algorithm produced a slightly lower silhouette score of 0.57 while still indicating relatively well-separated clusters, suggesting a lower level of clustering quality compared to k-means clustering.

Table 2: Silhouette Scores of Clustering Algorithms

K-means	BIRCH
59%	57%

Furthermore, the integrated machine learning model of the K-means clustering and KNN classification algorithms resulted in accuracy score of 0.767. This accuracy score indicates that the integrated model

is generally effective in recommending songs that align well with the characteristics of the derived data point of a given playlist. There are potential factors that can be further implemented to enhance the model's accuracy like including additional audio features that analyze intricate characteristics of the song such as Zero Crossing Rate and Mel-Frequency Cepstral Coefficients (MFCC). Nonetheless, an accuracy of 76.7% demonstrates that the integrated model performed sufficiently in the task of song recommendation where user preferences of recommended songs can be subjective can vary significantly.

5 Limitations

A hybrid filtering approach, which uses a combination of methods from collaborative filtering and content based filtering, is ideal in problems such as the one we aim to tackle. However, the data provided in this dataset does not provide user information. As such, we resorted to using content based filtering methods. This refers to recommending songs off user's preferences which matches with the statistics we were given in the data set. If given more history and information about users, incorporating collaborative filtering methods, such as pairing users based off traits and similar songs, would added depth to the model and increased the accuracy of the model.

Another issue we encountered regarding the dataset and problem defined was that this was partly an unsupervised learning task. The first stage, which obtains a silhouette score, is unsupervised and there is not way to verify this. However, we use genre as a proxy in order to make sure similar songs are in the same cluster during KNN classification. Using genre in this way adds a check to ensure higher quality results, however, gaining user responses through surveys and other research methods would have been more ideal if we had the means on a larger scale project. Most larger music recommendation systems and platforms have measures to actively trace and learn from user's behavior as users listen, and as a result models learn gradually over time.

6 Conclusion

Overall, this project successfully demonstrated the implementation of a combined machine learning model incorporating K-means clustering and K-nearest neighbor classification algorithms for song recommendation based on a given playlist. By analyzing key audio features from the Spotify Web API, our group applied K-means clustering and BIRCH clustering algorithms to categorize songs into distinct clusters of music genres. Furthermore, the integrated K-means clustering and KNN classification model achieved a relatively high accuracy score, indicating the model's effectiveness in recommending songs that align with user preferences. Moving forward, analyzing the waveform of songs from their audio files can generate specialized audio features like Spectral Rolloff that can potentially enhance the performance and accuracy of the integrated. Nonetheless, the integrated model was effective in the task of song recommendation, catering to diverse user preferences within the realm of music applications and online streaming platforms.

7 Contributions

7.1 Arya Chethikattil

My contributions to this project include background research, collecting best methods to share, and researching common weaknesses. Additionally, I contributed heavily to the report in those relevant sections in the introduction, technical background, related work, and limitations. During pair programming sessions, I contributed by helping navigate and brainstorming ideas and potential fixes to issues as they arose. I believe most of my contributions though were related to research and the provided recommendations following this.

7.2 Greeshma Challaram

My contributions include preliminary background research on different clustering algorithms, existing recommendation systems, and user preferences and behavior. Additionally, I also heavily contributed to the report on the technical work, approach, and results. During our pair programming sessions, I

actively participated by offering suggestions and solutions to address any issues that arose during the coding process.

7.3 Micah Ban

My contributions to the project are reviewing an assortment of research articles that present potential methods and applications that can be applied to our project. I also conducted exploratory data analysis by applying principal component analysis (PCA) and recursive feature elimination (RFE) to reduce the dataset to contain the most influential features. Additionally, I also implemented the BIRCH clustering algorithm to the reduced dataset and evaluated the algorithm's performance with the silhouette score. I also trained the integrated model that comprised the K-means clustering and K-nearest neighbor classifier. Furthermore, I implemented the integrated model to recommend songs based on an input playlist of songs. Lastly, I actively participated in weekly meetings and contributed to the project proposal, milestone report, and the final report.

7.4 Nhut Nguyen

My contributions to this project were the doing research on recommendation systems, looking at past research papers, writing down the motivations behind our project, and coming up with an evaluation plan. I also worked on the K-nearest neighbor classifier on my local machine. During our meetings, I actively contributed to collaboration of the project and make sure we are able to meet the deadlines. Finally, I worked on writing the milestone report and final report.

7.5 Asish Addanki

I actively contributed to the report generation during the project proposal with others and collaborated closely with Micah in the data pre-processing of the older dataset in the first Milestone; in which we made substantial progress, particularly in feature engineering for the initial dataset. Now, as we approach the final milestone, I've played a significant role in feature engineering for the new dataset and implemented K-means clustering and oversaw the meticulous pre-processing of the dataset for analysis and assisted in the process of integrating K-nearest neighbour model to recommend songs.

References

[1] Schedl, M., Zamani, H., Chen, CW. et al. Current challenges and visions in music recommender systems research. Int J Multimed Info Retr 7, 95–116 (2018). https://doi.org/10.1007/s13735-018-0154-

[2] GeeksforGeeks. K means clustering - introduction. GeeksforGeeks. https://www.geeksforgeeks.org/k-means-clustering-introduction/

[3] V. Dalal, "Birch algorithm with working example," Medium, https://medium.com/@vipulddalal/birch-algorithm-with-working-example-a7b8fe047bd4 (accessed Apr. 28, 2024).

[4] GeeksforGeeks, "K-nearest Neighbours (KNN)", [Online]. Available: https://www.geeksforgeeks.org/k-nearest-neighbours/. [Accessed: April 28, 2024].

[5] Huang, E. (2024). Recommender system using cosine similarity: An honors research project on music recommendation. Medium. Retrieved from https://medium.com/@edhuang392/recommender-system-using-cosine-similarity-an-honors-research-project-on-music-recommendation-120bfc8806ba

[6] "Get track's audio features," Web API Reference | Spotify for Developers, https://developer.spotify.com/documentation/web-api/reference/get-audio-features (accessed Apr. 28, 2024).

[7] M. Pandya, "spotify tracks dataset," Kaggle, https://www.kaggle.com/datasets/maharshipandya/-spotify-tracks-dataset (accessed Apr. 28, 2024).

[8] Srebrenik, B. (2018, December 4) Introduction to Music Recommendation and Machine Learning. Medium. Available: https://medium.com/@briansrebrenik/introduction-to-music-recommendation-and-machine-learning-310c4841b01d

[9] Verma, N. (2023, December 14). Optimizing K-means clustering: A guide to using the elbow method for determining the number of... Medium. https://blog.gopenai.com/optimizing-k-means-clustering-a-guide-to-using-the-elbow-method-for-determining-the-number-of-877c09b2c174

[10] H. Gultekin, “What is silhouette score?,” Medium, https://medium.com/@hazallgultekin/what-is-silhouette-score-f428fb39bf9a (accessed Apr. 28, 2024).

[11] Chang, E. (2021, December 17). Building a song recommendation system with Spotify. Medium. https://towardsdatascience.com/part-iii-building-a-song-recommendation-system-with-spotify-cf76b52705e7

[12] Kim, D., Kim, K., Park, K.-H., Lee, J.-H.,& Lee, K. M. (2007). A music recommendation system with a dynamic K-means clustering algorithm. Sixth International Conference on Machine Learning and Applications (ICMLA 2007). https://doi.org/10.1109/icmla.2007.97