Predicting Music Trends Across Countries*

Hamsa Gouda Veerendra
Computer Science
University of California, Riverside
hveer003@ucr.edu

Priyanka Jadli
Computer Science
University of California, Riverside
pjadl001@ucr.edu

Sahil Gami
Computer Science
University of California, Riverside
sgami003@ucr.edu

Samrita Sen Bakshi
Computer Science
University of California, Riverside
sbaks005@ucr.edu

Sourav Singha
Computer Science
University of California, Riverside
ssing263@ucr.edu

ABSTRACT

In this report we investigate how unsupervised learning (soft clustering algorithms) enables us to identify the songs into various clusters based on the attributes or the audio features. The audio features taken into account are loudness, liveliness, valence, pitch, acousticness, danceability, energy and tempo. As part of preprocessing we have been able to analyse the data after doing a dimensionality reduction via PCA. More specifically we extensively evaluate the dataset, emphasizing influence of the important design choices such as dimensionality reduction to enable the model to perform better clustering and providing for better clustering analysis.

Furthermore, we have implemented 4 soft clustering algorithms with a classification model which will enhance the understanding of the data and the underlying pattern even better.

CCS CONCEPTS

• Unsupervised learning • Weighted K means • Fuzzy C means • DBSCAN • Bisecting K means • Soft Clustering • songs • popular songs • clustering

KEYWORDS

Unsupervised learning, Weighted K means, Fuzzy C means, DBSCAN, Bisecting K means, Bisecting K means, Soft Clustering, Popular songs, Clustering, Music Similarity

1 Introduction

Increased diversity in music and with raging music streaming platforms such as Spotify, Apple Music the genres have been very hard to distinguish and often the lines separating various genres are blurred in current music recommendations

With this in mind we try to derive more meaningful insights for such music. The audio titles hosted on Spotify, each have their own popularity and huge following which also varies from region to region across the globe.

The Spotify Developer API allows developers/data scientists to get a list of popular songs worldwide and provide for their audio features. We are currently using the same data for our analysis.

Since, clustering is a task of grouping data based on similarity. Popular algorithms group data by firstly assigning all data points to the closest clusters, then determining the cluster. By implementing soft clustering algorithms we intend to, knowing the fact it is slower than hard clustering algorithms, map a single data point to multiple groups.

1.1 Data Collection

Our dataset was procured from the Spotify Web API using the /audiofeature API. However the data could only be procured via web-api using authentication which eventually led us to writing a wrapper for data-extraction API to pull data from Spotify using a token via API credentials. The following audio features were taken via the API

1. Acousticness: [0-1] Confidence measure of whether a track is acoustic
2. Danceability: [0-1] How suitable a track is for dancing

3. Energy: [0-1] Perceptual measure of intensity and activity. Energetic tracks feel fast,loud, and noisy.
4. Instrumentalness: [0-1] Predicts if the track contains no vocals
5. Liveness: [0-1] Predicts the presence of the audience in the recording
6. Loudness: [dB] Average volume across an entire track
7. Speechiness: [0-1] Detects the presence of spoken words in a track.
8. Valence: [0-1] Potential positiveness conveyed by the track
9. Tempo [0-300 BPM] The speed or pace of the given song

The track url was procured first and based on that the other audio features were pulled in to.

1.2 Data Processing

The following have been done for data preprocessing

1. Data Cleaning - Rows having null or missing values for a feature attributes for all column values are dropped

2. Data Imputation- Rows those had missing values were replaced with the mean of the values for that attribute. At present no rows have been observed with few missing feature values, rows that have at least single feature value present only for those rows this operation would be performed.

3. Data Normalisation - Following the visualization of data, three normalization techniques have been successfully implemented on the data, which includes z-normalization, min-max normalization, and standardization.

1.3 Dimensionality Reduction

First we tried to plot each of the data points in 2D graphs taking 2 features at a time. For example, the bottom left graph plots the points based on valence and acousticness features.

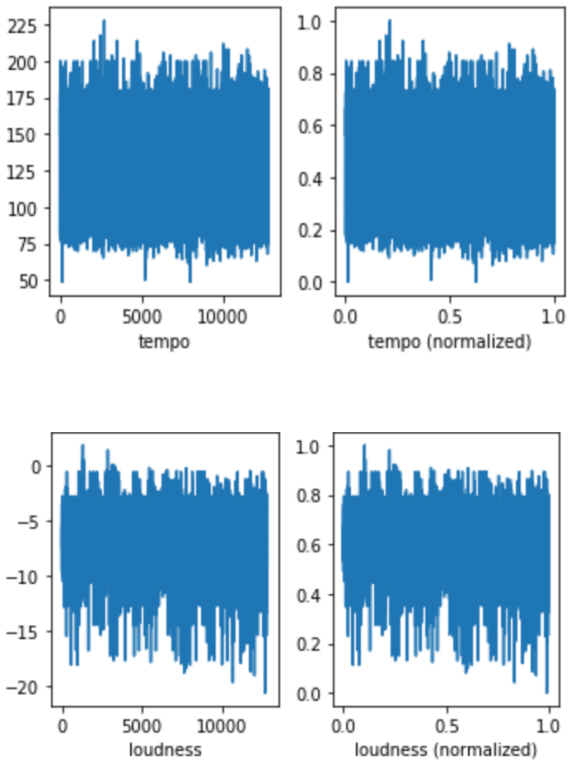


Figure 1: Scaled values for the attributes.

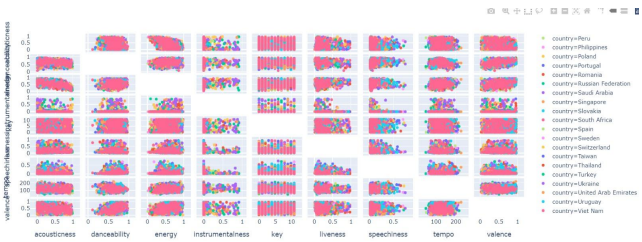


Figure 2: In all the graphs above, the points have been color-coded by country

We implemented 2 D PCA on the data. In our opinion there is not so much of a stark observation. However, there are some general observations we have made visually like- the majority of the data for Austria is mostly skewed towards the left while United States data is more spread out across the axes and a lot of the data for Germany coincides with data from Austria.

Figure 3: In all the graphs above, the points have been color-coded by country

Next a 3D PCA was implemented. As per the plot on the right again many data points of Austria and Germany match. They are more densely populated towards the bottom of the graph. Red points are mostly since the purple points are plotted first and then red ones.

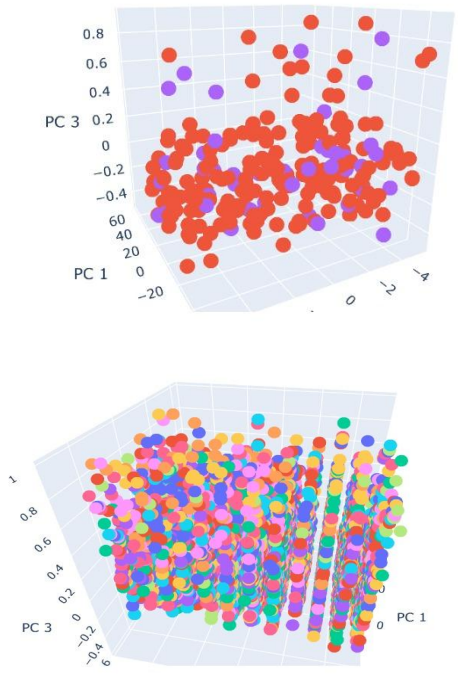


Figure 4: 3D PCA visualization. The data is seen in 12 horizontal lines since the data contains an attribute “key” which represents the pitches of songs and has integer values ranging from 0 to 11.

1.4 Model Implementation

The team has managed to implement 4 soft clustering algorithms and a classification algorithm. Every member of the group has implemented an algorithm to identify and observe the hidden patterns in audio features.

The algorithms have been implemented by the following members :

1. Fuzzy C-Means - Sahil Gami
2. DBSCAN - Hamsa Gouda Veerendra
3. KNN - Priyanka Jadli
4. Weighted K-means - Samrita Sen Bakshi
5. Bisecting K-means - Sourav Singha

All the algorithms are further detailed in section 4.

2. Related Work

It was seen that the songs attributes were used from spotify api for general trends in music in some papers focused on predicting trends in the future. The reason for using attributes is to analyze the model based on ‘the way songs are’ instead of the qualities of the song. [1] .

[3]The paper describes classification methods that recognize the genres of music using both supervised and unsupervised learning techniques. For unsupervised learning the K-mean method was implemented with a to evaluate the performance of our classification, the purity and Rand Index(RI) were used as opposed to the supervised learning method. The purity for a three genre classification was 0.822 which is significant and shows the performance is almost same as the supervised learning. For supervised learning, CART showed a recognition rate of 86.7% for 3 genres and 60.7% for 5 genres at maximum. Hence the results are comparable for both of them.

The KNN classification algorithm is also implemented in this project. [5] Some papers recommend initializing k with sqrt(number of observations in the training set) which is in the range of k tested in our implementation
While KNN has the advantages of being a simple and scalable algorithm, its biggest con is its lazy learning nature. Unlike algorithms like backpropagation, k-NN does not create a model of the input data before taking the test data as input. It holds out on any computations until the test data is actually encountered. This makes it quick to train but slow to classify test data. Also, this method is very sensitive to outlier data points. If a test sample has sparsely distributed known samples around it, the algorithm will cover a larger area for the same number of k, making it susceptible to outliers. [6]

This paper [7] briefly discusses modified algorithms that implement slight variations in the standard KNN algorithm.

Some of them are Weight adjusted KNN (includes assigning weights to the neighbors based on their distances) and KNN with K-Means (eliminates the requirement for parameter k). While these work faster, they give lower accuracies.

3. **Proposed Method**

The clustering stage receives multivariate features and these proposed soft clustering methods help us group the songs based on their attributes like dancelibility, loudness, energy, pitch (key) liveliness etc.

4 **Experimental Evaluation**

The algorithms implemented by the team have been evaluated on various parameters. To begin with the comparison amongst the silhouette scores of all the models are comparable with their respective library function. The custom algorithms like Weighted K-means, Fuzzy C-means, Bisecting K-means and DBSCAN have the following scores 0.315, 0.30, 0.38, 0.39 respectively and the respective library functions score 0.359, 0.38, 0.39, 0.262. This silhouette score comparison shows that the custom made algorithms are at par with the library functions and hence project accurate results.

The elbow method has been used to identify the optimum clusters for the respective algorithms and bolstered the algorithm's proper clustering analysis and a strong correlation has been seen in the music attributes. The clustering algorithm has been implemented using euclidean distances. Also the clustered groups show a similarity in the countries where the music is popular.

4.1 **Weighted K-means**

K Means is an algorithm for partitioning the data into k distinct clusters. Given an initial K-cluster the distance between all data points are computed and the existing K centroids and reassigned each point to its nearest centroid accordingly. Followed by calculating the new K centroids by taking the data points' mean based on the new cluster assignment. However by using the weight K means we add a flavour to the data. In the first place the centroids initially as mentioned above that get created now take into account the "weight" factor, which in our case we have taken to be a danceability attribute as the same. Thinking of these K centroids as anchors for each cluster. To give an interpretation of the influence of the weights on the algorithm basis this weight choses the gravity of the weight and pulls the centroid nearer to it. Hence the clusters now formed are the ones that have taken into account the component of danceability in the music.

To decide the optimal numbers of clusters for the given data the plot between SSE and n_clusters helps determine the optimal K. Typically the graph would form an elbow shape and the very knee determines the number of clusters.

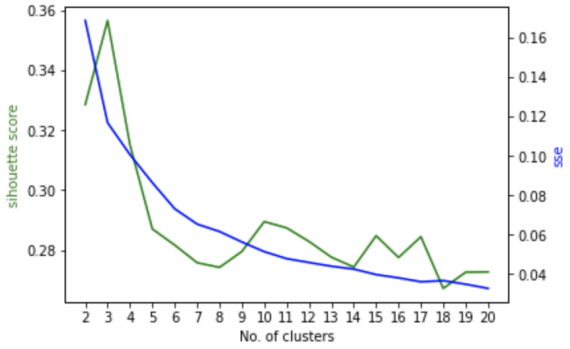


Figure 5: SSE, Silhouette Score v/s number of clusters. SSE decreases as the number of clusters increases. There a various optimal points here K=3,5,7

Further insights on clustering the data for K=3.

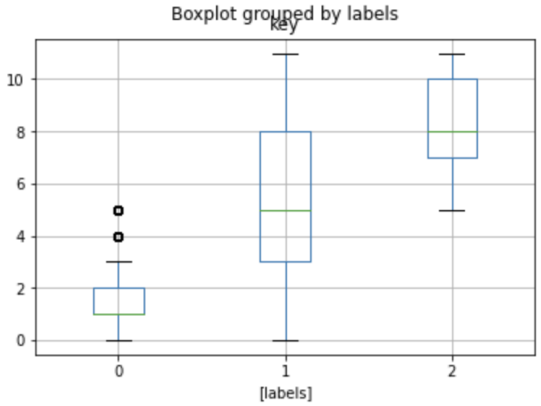
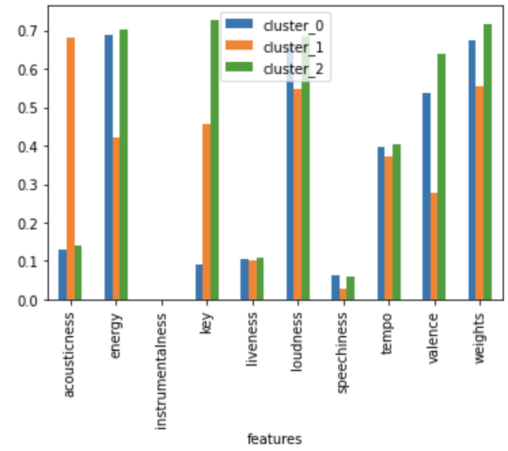
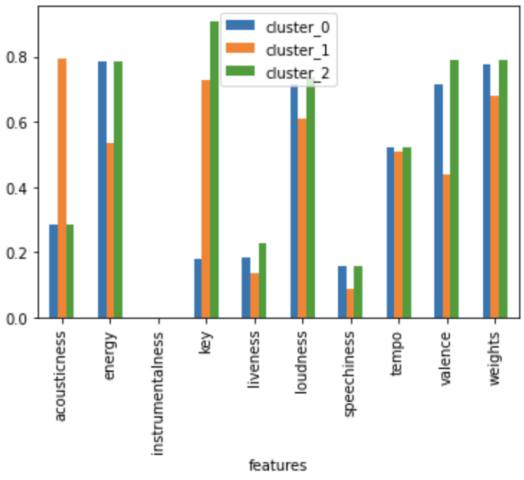


Figure 6: the 'Key' (Pitch) is evaluated for all the 3 clustered observed as the primary differentiating factor amongst the clusters is 'key' attribute

4.1.1 Comparison - Silhouette Score To gauge the efficacy of the model custom built a comparison of the Silhouette Score was done and it was observed that the *Sklearn* library function computed a score of 0.359 and the custom function computed a score of 0.315. This observation helps us derive a conclusion that the custom built function produces almost the same results as the pre-built library function which ensures the clustering accuracy is maintained.

4.1.2 Clustering Analysis To gauge and understand the key differentiating features in the clusters an analysis of the audio features clustered together gives us insights into the groups. These observations have been made based on the mean, median and the std of the features.





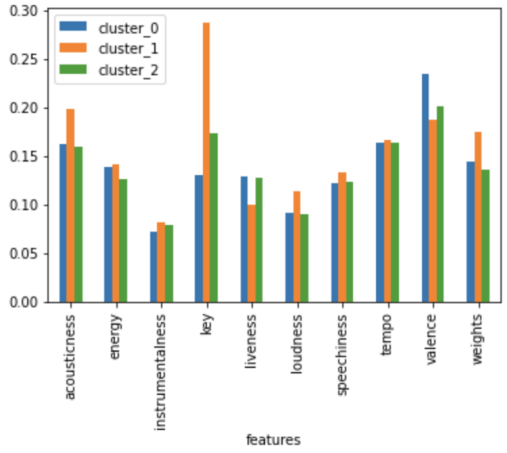


Figure 7: Describes data from cluster 1 , 2 and 3 based on mean, median and std of the attributes.

There are significant similarities noticed between the first and the third clusters. As per the observation the mean of the primary lies in the same range whereas the key (pitch) is the differentiating factor. Whereas cluster 2 has differentiating features like acousticness, valence from the rest. These variations in the attributes helped us in understanding the data further and after some observation we could see a pattern as the songs from cluster 1 primarily belong to popular songs in the North American subcontinent and European ones who are majorly english speaking. Cluster 2 comprises songs prevalent in the south asian countries and cluster 3 in south american countries.

Figure 8: Prevalence of Song from Cluster 1,2 and 3 in the respective countries.

4.2 Fuzzy C Means

Fuzzy C Means clustering is a clustering which has a data point in all the clusters with an index of similarity with each cluster. Working of Fuzzy C Means is very similar to that of K Means clustering but, instead of assigning each point to a cluster with respect to its cluster centroid and changing the cluster centroid after each epoch, we assign all the points a weight or closeness percentage with each cluster centroid and then change the centroid accordingly. A new matrix of size N x k has the similarity or closeness index of each data point with respect to each cluster where N is the number of data points and k is the number of clusters.

Fuzzy C Means was used because the data had many repeating points as the dataset consists of songs and some songs might be popular and trending at multiple countries at the same time period. The clusters obtained after applying Fuzzy C Means gives us the similarity of songs amongst different countries and what type of songs are most listened to in various countries, and the trend of similar songs with similar attributes.

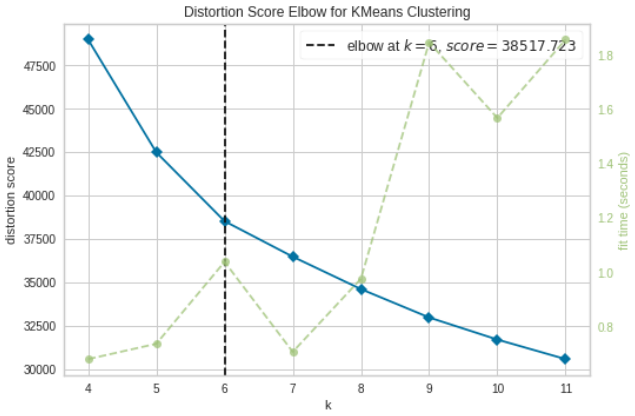


Figure 9: Fit time vs number of clusters, elbow at k = 6

The graph of number of clusters ‘k’ vs fit time // distribution score has an elbow at k = 6 giving us the optimal number of clusters as 6, nonetheless the algorithm has low fit time at k = 7 with no significant drop in runtime and better cluster distribution at 7 clusters. (Runtime for 12,812 data points with 6 attributes and 7 clusters is around 35 mins.)

After some feature selection it was found that the attributes ‘speechiness’ and ‘tempo’ gave a very vague set of clusters and a very low total variance amongst the clusters and data points. Dropping the said two attributes resulted in cleaner and well distributed clusters with a relatively high variance amongst clusters. After clustering, the representation of the obtained clusters was done using PCA of scikit learn library of python as seen in figure 10 below.

Figure 10: PCA visualization of the obtained clusters (2D)

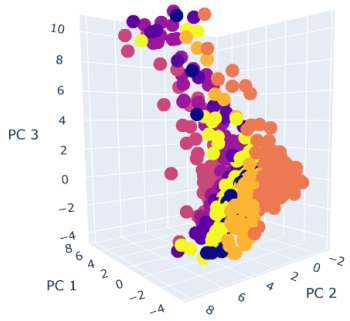


Figure 11: PCA visualization of clusters in 3D

The clustering was done on the original dataset with 7 attributes (dropping Tempo and speechiness) and then PCA was used for visualization, the data points were colored based on the cluster number they were clustered in. Attempting Clustering on attributes obtained from PCA resulted in inconsistent results hence the usage of PCA was limited to visualization.

We can see each cluster’s similarity for different points. As seen in the figure 12 the data points colored yellow are closer to the first cluster and data points having blue-er tone have low similarity index for the first cluster. Here we can see how different clusters have different cluster similarity index and also each data point belongs to all the clusters, with different similarity index, the figure 10 is based on the highest similarity index of a point with a cluster.

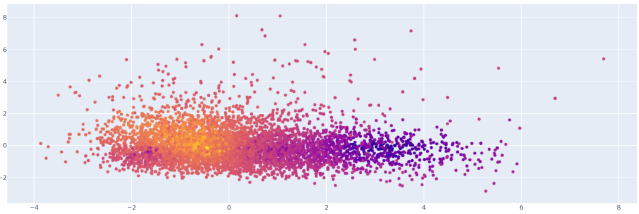


Figure 12: PCA of cluster number 0 (first cluster)

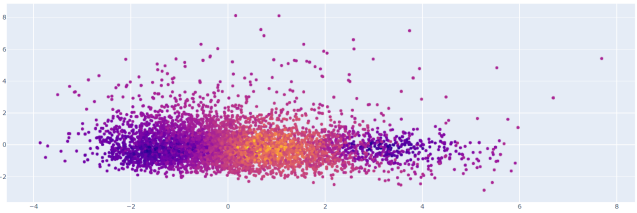


Figure 13: PCA of cluster number 1 (second cluster)

Figure 14: Prevalence of Songs of Cluster 0 (first cluster)

From the figure 14 we can infer that any song that fits in cluster number 0 has higher similarity to the songs from countries 'Norway', 'Greece' and 'Canada' and lower similarity to other countries as shown in the word cloud. Similarly the word cloud of all the clusters can be obtained.

4.3 **DBSCAN**

Density based spatial clustering algorithm with noise is the clustering algorithm that requires only one input parameter and supports the user in determining an appropriate value for it. This particular algorithm relies on the density based notion of the clusters.DBSCAN expects two input parameters which are MinPts and Eps. With an arbitrary point p which is a core point, the algorithm yields a cluster with respect to the two input parameters. It also considers noise as a set of points As global values are used for MinPts and Eps, DBSCAN may merge two clusters to one cluster according to the input which are non - empty subset of the database satisfying Maximality and Connectivity. Usually, MinPts as a parameter is eliminated further while computing the DBSCAN by setting it to 4 for all 2-D data. The parameter Eps and MinPts can be determined by considering the distance of a point to its k-th nearest neighbor, then the neighborhood of the point would consist exactly of the k+1 exactly otherwise it is quite unlikely.

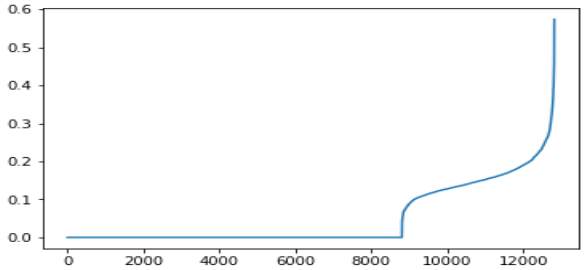


Figure 15: Distances plotted to find the Epsilon using library function

Hence the clusters now formed are the ones that have taken into account the component of liveness in the music.

The clustering was done on the original dataset and then PCA was used for visualization, the data points were majorly colored to one based on the cluster number they were clustered in. Attempting Clustering on attributes obtained from PCA resulted in inconsistent results hence the usage of PCA was limited to visualization.

4.3.1 Comparison - Silhouette Score

To measure the efficacy of the algorithm that is custom built, the comparison of Silhouette Score was done and it was observed that the Sklearn library function computed a score of 0.262 and the algorithm implemented computed a score of 0.356 and can be further inferred from the same that for the dataset, the density was clearly accumulated to one feature.

4.3.2 Clustering Analysis

For implementing the custom built algorithm, first the labels were assigned(noise, core point, edge or the border point), and then the neighbor points were found and by finding the core, border points and assigning the cluster, the Eps and MinPts were assigned for which the Silhouette score was found in order to compare.

4.4 **Bisecting K-Means**

Bisecting K-Means is a centroid based clustering algorithm that combines divisive hierarchical clustering with K-means, which not only gives the clusters but also the hierarchical structure of the data points. First it considers all the points as a single cluster and breaks it into two sub clusters and for each next step it breaks a sub cluster with the highest SSE (using Squared Euclidean distance in this implementation) observed into another two sub clusters by applying K-Means (K= 2) in successive manner until the desired number of clusters has been obtained. Due to this binary nature of breaking down a cluster, Bisecting K-Means works much more efficiently over normal K-Means and beats it in entropy measurement. For the dataset Bisecting K-Means is a good fit as the number of features are more and the various features like- country, acousticness, danceability, energy, instrumentalness, key, liveness have strong dependence on each other and may exhibit a hierarchical or segmented relation with each other, and also many popular tracks may have similar values across features in a repeating nature and most importantly as Bisecting K-Means divided the clusters with highest SSE at every step hence the each cluster can be expected to have somewhat equal number of points it. The quality of the clusters, cohesion and the pattern observed from the

generated clusters is shown further into the report.

4.4.1 Comparison - Silhouette Score

The implemented Bisecting K-Means and K-Means module showed on par Silhouette Score against the library implementation as shown in Figure 16. The library implementation of K-Means algorithm reached a score around 0.39 whereas the implementation version reached around 0.38. The Bisecting K-Means had a slightly lower score than the K-Means but was comparable at 0.34 when cluster number was around 4 to 6.

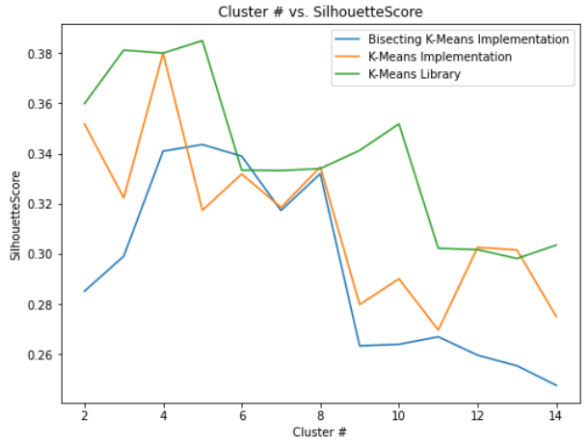


Figure 16: SSE, Silhouette Score v/s number of clusters for Bisecting K-means. Optimal number of cluster points are at K = 3 to 8

4.4.2 Cluster Analysis To group the tracks into different clusters the Bisecting K-Means algorithm was applied following the application principle component analysis using developed module specifically for this dataset on the following features: acousticness, danceability, energy, instrumentalness, key, liveness, loudness, speechiness, tempo, valence with the PCA value 3.

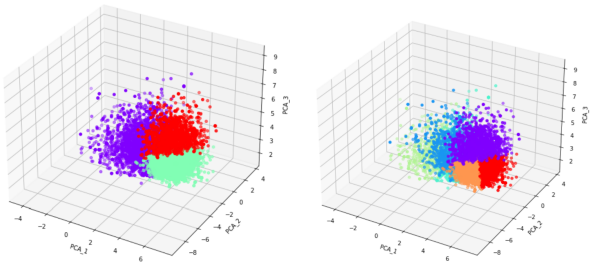


Figure 17: Cluster Visualization 3D along all features for K = 3, 6

k-NN is a supervis Figure 17 shows that the tracks can be effectively segmented into minimum 3 segments that shows clear distinction between them and also highlights the similarity between them. On further deep dive into the individual features various sets of features showed strong correlation between them using which they could further segments on smaller sets of attributes as shown below.

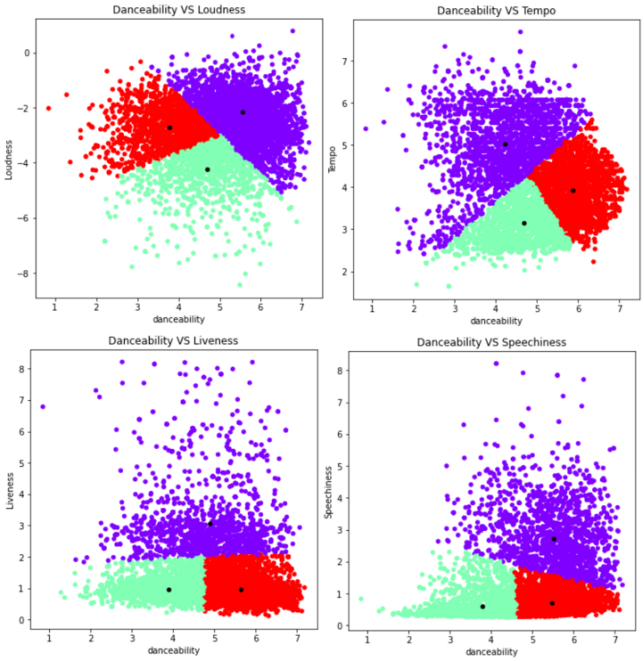


Figure 18: Cluster Visualization for the feature danceability against Loudness, Tempo, Liveness, Speechiness

The feature danceability showed positive correlation with features like loudness, tempo etc. but showed slight negative dependence on features like liveness, speechiness. Figure 18 shows that danceability can be clearly segmented into 3 clusters, where tracks having high magnitudes are clustered together or low magnitudes of features are clustered together.

For example using this understanding we can identify tracks with high danceability and loudness in one segment and or vice versa tracks with high danceability and low speachiness into one.

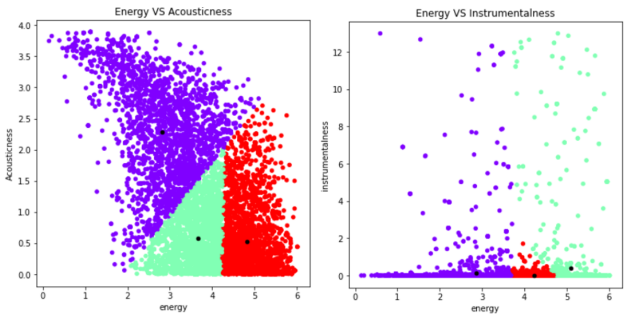


Figure 19: Cluster Visualization for the feature energy against acousticness and instrumentalness

However all features can not be strongly clustered together but can only be grouped together such as energy and instrumentalness.
The trends can also be displayed using the bisecting k-means and identify for other combinations of features. By identifying the choice of features from one cluster similar songs clustered with different features can be identified.

4.5 KNN

k-NN is a supervised algorithm used to solve classification problems with categorical data. In it, the data point of interest (unseen, unlabelled data) is assigned a class based on the classes of the k closest known data points around it. The algorithm is:
Step 1: Select a value of k
Step 2: Find the distance of the current test point with all its neighbors
Step 3: Find the k neighbors with the least distances from the test point
Step 4: Find the class labels of these closest neighbors and assign the most frequently found label to the unknown point
The main crux of this algorithm is to find the optimal value of k. Too large or too small of a value will produce incorrect results since the former may bias the data against classes with fewer samples while the latter will make the model susceptible to outliers.

The k-NN algorithm works with many different distance functions. The most commonly used are Euclidean distance and Manhattan distance which are both parts of the Minkowski distance family. Others include cosine distance (mainly used in test analysis applications), hamming distance (used with binary data strings), etc.

In our implementation, we have used the Euclidean and Manhattan distances to find the accuracy of KNN. First, the ”track_url” and ”Unnamed:0” columns are dropped and the rest of the columns form the X feature matrix. The ‘country’ column in the df forms the Y matrix. The two rows in the dataset with NaN values are populated with data from the previous row. The PCA implementation with the desired number of components is carried out. We then split the data into train and test subsets. The training set is standardized to have a centered mean and unit variance using StandardScaler. Then the KNN model is implemented with accuracies recorded as a plot against k in the range tuned by the user as given in Figure 20.

It is observed that the model provides low accuracy on our dataset. The primary reason could be since KNN is a hard clustering algorithm, while our data contains multiple duplicate song entries for different countries. For instance, a certain song ‘song 1’ could be in the Top 200 charts of ‘country1’ and ‘country2’ but when provided as a test point, our model would only classify it as one of these country labels (let's say country 1 for instance) each time it is found in the test set. This could be causing a decrease in accuracy.
However, as shown in the above table, our model does provide accuracy comparable to accuracy computed by the sklearn library implementation of KNN.

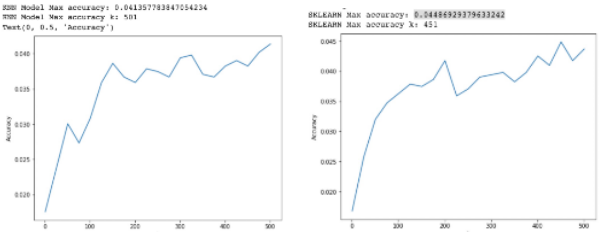
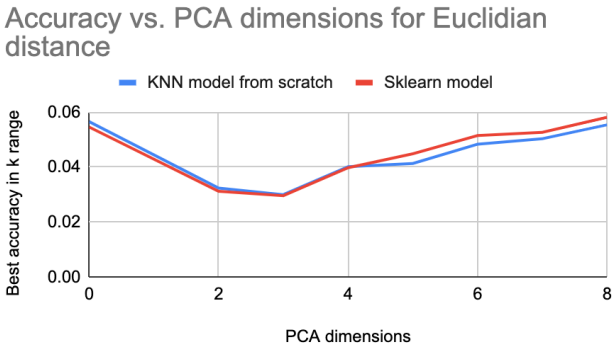


Fig 20: Accuracy vs. k plot for PCA components=5 and distance function = Euclidean
Our model from scratch (on left) and sklearn model performance (on right)
All the experiments were recorded considering k in range (1, 502, 25) and run on the entire data. Each case consisted of the features (mentioned as default features in Table 1) : acousticness, danceability, energy, instrumentalness, key, liveness, loudness, speechiness, tempo, and valence. The features instrumentalism and streams were considered in some cases mentioned in the table.



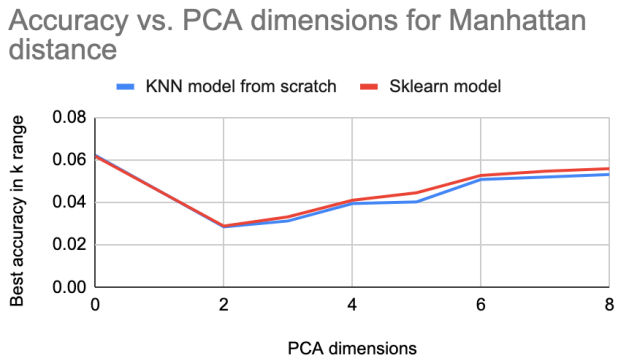


Fig 21: Best accuracy vs. PCA dimensions with different distance functions. As seen, the model from scratch closely follows trends of the Sklearn model implementation.

PCA _Dim	Distance used	Best accuracy in k range	At k =	Best accuracy in k range	At k =	Features
0	Euclidian	5.66	401	5.46	401	Default features
0	Euclidian	7.8	326	8.12	126	Streams added Instrumentalness removed
0	Manhattan	6.2	476	6.16	476	Streams added
0	Manhattan	9.99	301	10.46	301	Streams added Instrumentalness removed
2	Euclidian	3.24	401	3.12	426	Instrumentalness removed
2	Euclidian	10.46	26	9.91	26	Streams added Instrumentalness removed
2	Manhattan	2.85	451	2.89	401	Instrumentalness removed
2	Manhattan	10.03	201	9.83	76	Streams added Instrumentalness removed

Table 1: As observed, the accuracy usually increases as the 'streams' column is added to default features and as instrumentalness column is removed.

As shown in table 1, It is observed that when the streams feature is added, the accuracy of the model increases since this eliminates to some extent the duplicacy issue since the number of streams will show variation across countries as well. However, in a scaled-up version the streams feature may not be very helpful with a song that has recently been released (as it would have 0 streams) but will work if the song test point has had some time to gather a decent number of streams.
We have considered removing the instrumentalness feature in some experiments since its value is 0 in the data set for 6742 entries out of the total of 12814 entries.

5 Discussion & Conclusion

While working on unsupervised learning algorithms learnt that although it identifies previously unknown patterns and does not require manual effort , however there is uncertainty about the accuracy of the unsupervised learning outputs and it's difficult to check the accuracy as there is no labelled data. We also found out that we can predict other attributes like danceability from remaining song attributes of the songs with some higher accuracy. [1]

While collecting data from spotify we learned that data rate collection was very inconvenient and random. Sometimes the api request method allowed us to pull 50 data frames at once and sometimes we were able to pull hundreds of song attributes at once. We also understood the importance of feature selection and dimensionality reduction for improving performance as well as reducing runtime, removing unwanted data points, managing incomplete data points, and how it can affect the model performance and model prediction. We used graphs, data visualization using scatterplot and wordcloud to analyze the data and decide what algorithms to use to make a model. We also saw that using a classification model could be used to predict countries by looking at data and point distributions.

5.1 Future Work

It was observed that not many people like the songs recommended to the users via the recommendation systems on these streaming platforms , hence by exploring and designing these clustering algorithms has helped us understand how every audio feature of the song can help us identify commonality in music for every user. The comparison of the above implemented methods have led the team to believe that the clustering could be further improved clustering and the audio feature could enable to develop a good recommendation system based on the artist and track popularity based on the regions. Later, the clustering and classification algorithms can be modified to predict other attributes in the dataset as well.

REFERENCES

[*1] Menten, Matthew & Ng, Kieren & O'Rourke, Tanner & Holmes, Rourke. (2018). Temporal Trends in Music Popularity - A Quantitative analysis of Spotify API data. 10.13140/RG.2.2.11551.71843.*

[2] Grover, Nidhi. (2014). A study of various Fuzzy Clustering Algorithms. International Journal of Engineering Research. 3. 177-181. 10.17950/ijer/v3s3/310.

[3] Kim, Kyuwon, Wonjin Yun, and Rick Kim. Clustering music by genres using supervised and unsupervised algorithms. Technical report, Stanford University, 2015.

[4] Ester, M., Kriegel, H.P., Sander, J. and Xu, X., 1996, August. A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd (Vol. 96, No. 34, pp. 226-231).

[5] Zhang Z. (2016). Introduction to machine learning: k-nearest neighbors. Annals of translational medicine, 4(11), 218. https://doi.org/10.21037/atm.2016.03.37

[6] Wang, Lishan. (2019). Research and Implementation of Machine Learning Classifier Based on KNN. IOP Conference Series: Materials Science and Engineering. 677. 052038. 10.1088/1757-899X/677/5/052038.

[7] K. Taunk, S. De, S. Verma and A. Swetapadma, "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 1255-1260, doi: 10.1109/ICCS45141.2019.9065747.