Song Selection Based on Your Favorite Artist

***Pranav Prajapati*[1]**

1Stevens Institute of Technology, Hoboken, 07030, United States of America

A R T I C L E I N F O

Keywords:
Web Scraping
PCA
KNN
Clustering
K-means
K-medoid
Python
R

A B S T R A C T

A recommendation system for songs is a common feature on any music platform. Numerous sophisticated expert systems are available to carry out music recommendation tasks to enhance user experience on platforms like Apple Music, YouTube, etc. However, we often come across recommendations based on our recent activity on the said platform. Our favourite artists release music albums generally once every 2 years. Therefore, such recommendation systems are suboptimal for the best music experience. In this paper, we combine both supervised and unsupervised learning to create a recommendation system that uses audio features as independent variables. Our project aims to primarily provide listeners with playlist recommendations based on their favourite artists. We have used KNN classification to determine the songs that match our likeness and perform K-means and K-medoid clustering to find similar songs from the classified playlist and cluster them into smaller playlists.

1. Introduction

Digital music distribution has taken over all traditional distribution methods and streaming services like Apple music, Spotify, Tidal have become increasingly popular. Everyone has their personal favorites when it comes to music. We often find it difficult or less confident when exploring new songs. A few years back individuals used to ask their friends, colleagues or relatives to suggest them good music. But in today's growing technology and fast paced world people don't that have much time. That's why there are automated song recommendation systems.
The noun recommendation stems from Middle Latin word recommend meaning "present as worthy". Good recommendations reduce user's efforts and valuable time in making decisions. There has been a lot of study about music recommendation systems but the problem of such kind of recommendation continues to remain complicated because of multiple factors that influence the listener's preferences. We feel that a very important factor is being neglected by the current song recommender systems.

Every individual has their own personal favorite artist that they follow diligently thought their musical career. But since those favorite artists cannot put out music frequently, the listeners would surely like to listen to songs by artists that are similar to their styles or have the same voice. Some recommender systems do provide such results based on favorite artists, but they rely heavily on genre classification. But with the recent digital transformation in music production styles, there isn't a specific genre of a song. Tools like Ableton Live, FL Studio etc. enable producers to mix, match, auto-tune and distort audio. Hence there is a need to study each song's scientific attributes to find the right match. This project focuses on those methods to recommend 3 playlists to a listener from more than 1000 songs.

2. Background

2.1. Problem Description

The currently used music recommended systems rely on the number of views, ratings, likes, genres and recent popularity of songs. This confines the listener's exploration range, thus exposing only popular artists and song titles. Hence to improve the recommendations the songs' scientific attributes must be taken into account. This project aims to solve the problem of recommendation of songs not only according to the current trend of music but also by finding the songs that are heavily related to their favourite artists.

2.2. Web Scraping and Dataset Evaluation

To obtain our datasets we decided to scrape our own data. To do this we used Spotify's web API that provides something called audio features, which are, as the name implies, features or characteristics of a song. We used Spotipy, a lightweight python library for the Spotify Web API to make authorized token requests and run our scripts for extracting these features. So according to our preferences 10 features we extracted for each song of that particular artist. There were total 219 songs and 1000 more random playlists songs data extracted from 2018. Once the songs were scraped, they were combined the data and split them into training and testing datasets.

In order to perform classification on the dataset, we must have a target variable that classifies the data. However, since we scraped our own dataset, we had to create a target variable and assign each data row to its corresponding class. For our dataset, we had 2 classes, 0 and 1, where 0 indicated that the song was not similar to our favorite artist's style, and 1 indicated that it was.

The challenge remained as to how the classification into 0 and 1 be performed and on what basis. We generated density plots for each music feature (independent variables of our dataset) and analyzed the mode and assigned the classes based on the mode value.

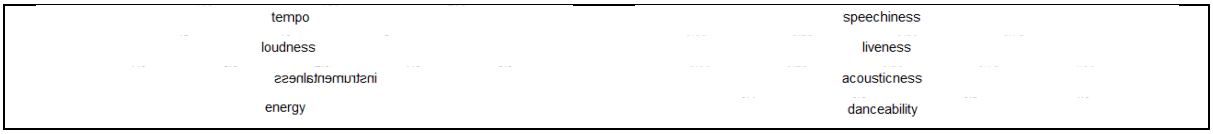


Figure 2.2.a

3. Methodology

3.1. Data Cleaning and Preparation

Since our Data was scraped from the web, the dataset did not contain any outliers or missing values. However, to clean our data we removed the following unnecessary variables:

1) Album name
2) Artist name
3) Type
4) Release date
5) Year
6) Minutes
7) Time signature
8) Feature

from the dataset, since they would have not added value to our classification model. We also deleted the column Danceability.1, because it was listed twice. Before cleaning, the dataset had 20 independent variables. After deletion of the unnecessary columns, we had 12 independent variables and one target variable present in our dataset.

3.2. Dimension Reduction and Pre-processing

In classification problems, there are many factors which are the basis of classification These factors are fundamentally variables termed as features. If there are too many features, it becomes difficult for the algorithm to perform classification optimally. Sometimes, most of these features are correlated, and therefore are redundant. This is where dimensionality reduction algorithms come into play.

Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables

The most common dimensionality reduction methods are:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)

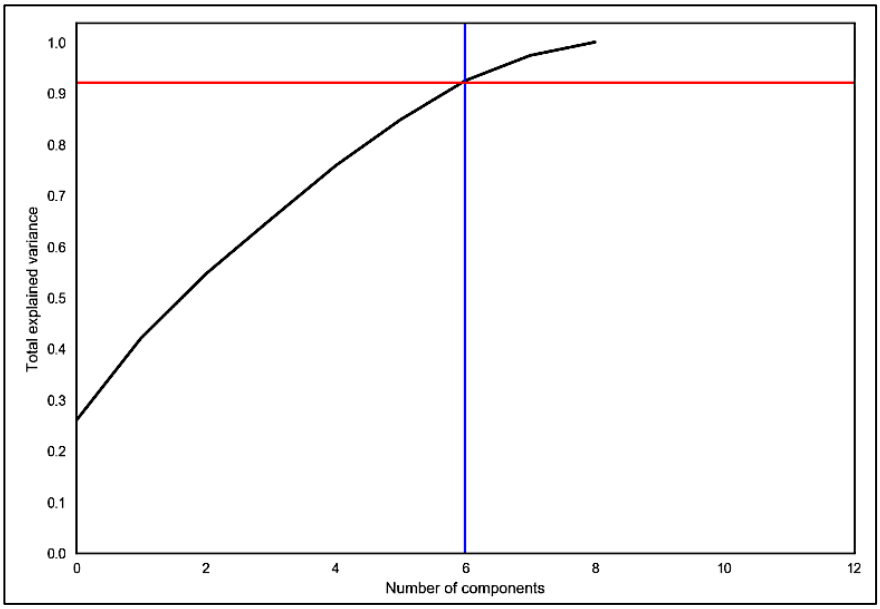
For this project, we have used Principal Component Analysis to perform dimensionality reduction. PCA works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.
To analyse the number of Principal Components to be considered to have a relatively high variance, we first standardized the features by removing the mean and scaling to unit variance.

The standard score of a sample x is calculated as:
$z = (x - u) / s$

where: u = mean of the training samples or zero if with_mean=False
s = standard deviation of the training samples or one if with_std=False.

This was achieved by the sklearn.preprocessing.StandardScaler() class. We further performed Linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.

By plotting the PCA, we got the following results:



Figure

Therefore, as we can notice from the above plot, having six principal components will give us a 92% variance, which is good enough for music recommendation. So, we move forward with number of components = 6.

3.3. Performance Prediction

The problem of predictive modelling is to create models that have good performance making predictions on new unseen data.
Therefore, it is critically important to use robust techniques to train and evaluate your models on your available training data. The more reliable the estimate of the performance on your model, the further you can push the performance and be confident it will translate to the operational use of your model.

Hyper-parameters are parameters that are not directly learnt within estimators. In scikit-learn they are passed as arguments to the constructor of the estimator classes. Typical examples include C, Kernel and gamma for Support Vector Classifier, alpha for Lasso, etc.
It is possible and recommended to search the hyper-parameter space for the best cross validation score.

Cross-validation is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of *known data* on which training is run (*training dataset*), and a dataset of *unknown data* (or *first seen* data) against which

the model is tested (called the validation dataset or *testing set*). The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

In summary, cross-validation combines (averages) measures of fitness in prediction to derive a more accurate estimate of model prediction performance.

To predict which classification algorithm will perform the best on our dataset, we performed a GridSearchCV for the following classification algorithms:

1) KNN Classification
2) Random Forest Classification
3) Decision Trees Classification

The GridSearchCV instance implements the usual estimator API: when "fitting" it on a dataset all the possible combinations of parameter values are evaluated and the best combination is retained.

Algorithm	GridSearchCV Score
KNN	0.8032786885245902
Random Forest	0.7180327868852459
Decision Trees	0.740983606557377

Table 3.3.1

On the analysis of the results obtained from the GridSearchCV, we noticed that out of the tree algorithms, the best performance score was predicted for KNN, with a score of 0.803278. The lowest score was predicted for Random forest at 0.718032, and a relatively better score was obtained for Decision trees at 0.740983.

3.4. Supervised Learning

3.4.1. KNN Classification

From the scores obtained by GridSearchCV, we determined the best algorithm for the classification of our dataset was the KNN classification algorithm.
The first step in KNN classification is to determine the value of 'K'. The GridSearchCV also predicts the number of neighbours optimal for our dataset, which can be seen in the image below:

```
Out[12]: ({'n_neighbors': 3}, 0.8032786885245902)
```

Figure 3.4.a

Therefore, the best value of k for KNN classification of our dataset is set to 3. The next step is to split the target variable 'y' from our training and testing dataset and fit the model on the training data.

We then transform our scaled testing data into principal components and run the transformed

data to the KNN function. The KNN algorithm provides us with a new playlist with 77 songs classifies as 1, i.e. similar to our favourite artist's style. The image below shows the songs that have been classified to new_class 1.

name	Unnamed: 0	popularity	danceability	acousticness	energy	instrumentalness	liveness	loudness	speechiness	tempo	class	new_class
Drip Too Hard (Lil Baby & Gunna)	1	86	0.896	0.10300	0.671	0.000000	0.5520	-6.977	0.2890	112.502	1	1
Without Me	2	94	0.752	0.29700	0.488	0.000009	0.0936	-7.050	0.0705	136.041	0	1
Taki Taki (with Selena Gomez, Ozuna & Cardi B)	4	100	0.841	0.15300	0.798	0.000003	0.0618	-4.206	0.2290	95.948	0	1
Happier	5	99	0.687	0.19100	0.792	0.000000	0.1670	-2.749	0.0452	100.015	0	1
Moonlight	6	95	0.921	0.55600	0.537	0.004040	0.1020	-5.723	0.0804	128.009	0	1
Nonstop	7	93	0.912	0.01640	0.412	0.013000	0.1040	-8.074	0.1240	154.983	1	1
FEFE (feat. Nicki Minaj & Murda Beatz)	10	94	0.931	0.08800	0.387	0.000000	0.1360	-9.127	0.4120	125.978	0	1
Better Now	11	95	0.680	0.35400	0.563	0.000000	0.1360	-5.843	0.0454	145.028	1	1
Never Recover (Lil Baby & Gunna, Drake)	12	90	0.757	0.07160	0.690	0.000000	0.1920	-5.327	0.2820	132.064	1	1
I Love It (& Lil Pump)	13	98	0.901	0.01140	0.522	0.000000	0.2590	-8.304	0.3300	104.053	0	1
In My Feelings	14	97	0.835	0.05890	0.626	0.000060	0.3960	-5.833	0.1250	91.030	1	1
SICKO MODE	16	97	0.834	0.00513	0.730	0.000000	0.1240	-3.714	0.2220	155.008	1	1
ZEZE (feat. Travis Scott & Offset)	17	95	0.826	0.07100	0.615	0.000000	0.0965	-7.979	0.2190	98.056	1	1
MIA (feat. Drake)	18	97	0.818	0.01430	0.540	0.000512	0.0990	-6.350	0.0544	97.064	1	1
Arms Around You (feat. Maluma & Swae Lee)	19	84	0.694	0.03020	0.724	0.000002	0.1260	-5.450	0.0424	104.943	1	1
Better	20	62	0.551	0.08030	0.563	0.234000	0.1060	-10.271	0.0865	97.846	0	1
SAD!	21	95	0.740	0.25800	0.613	0.003720	0.1230	-4.880	0.1450	75.023	0	1

Figure3.4.b

However, if we print the list of songs that have been classified as 0, we can see a few songs that were previously classified as 1, but the KNN algorithm changed them to 0 in the new_class column.

name	Unnamed: 0	popularity	danceability	acousticness	energy	instrumentalness	liveness	loudness	speechiness	tempo	class	new_class
Sunflower - Spider-Man: Into the Spider-Verse	0	92	0.757	0.5410	0.501	0.000000	0.0718	-5.593	0.0466	89.931	0	0
Lucid Dreams	3	96	0.511	0.3490	0.566	0.000000	0.3400	-7.230	0.2000	83.903	0	0
Shallow	8	91	0.572	0.3710	0.385	0.000000	0.2310	-6.362	0.0308	95.799	0	0
when the party's over	9	89	0.498	0.9790	0.104	0.000079	0.0895	-14.080	0.0621	124.001	0	0
Mo Bamba	15	91	0.729	0.1940	0.625	0.009860	0.2480	-5.266	0.0315	146.034	0	0
Falling Down	25	96	0.669	0.0175	0.574	0.002940	0.1460	-6.442	0.0286	120.013	0	0
Noticed	27	80	0.702	0.4560	0.571	0.000000	0.1160	-7.540	0.0536	80.012	0	0
changes	41	91	0.669	0.8830	0.308	0.000000	0.0984	-10.068	0.0290	64.934	1	0
SLOW DANCING IN THE DARK	47	66	0.515	0.5440	0.479	0.005980	0.1910	-7.458	0.0261	88.964	0	0
Trip	51	84	0.477	0.2250	0.610	0.000000	0.1070	-5.628	0.1440	79.882	1	0
I Fall Apart	52	88	0.556	0.0689	0.538	0.000000	0.1960	-5.408	0.0382	143.950	1	0
Beautiful (feat. Camila Cabello)	59	93	0.638	0.3460	0.717	0.000000	0.1050	-4.722	0.0337	100.027	0	0
Girls Like You (feat. Cardi B)	61	87	0.851	0.5680	0.541	0.000000	0.1300	-6.825	0.0505	124.959	0	0
lovely (with Khalid)	62	90	0.351	0.9340	0.296	0.000000	0.0950	-10.109	0.0333	115.284	0	0
Nice For What	63	89	0.586	0.0891	0.909	0.000109	0.1190	-6.474	0.0705	93.394	1	0
This Feeling	65	92	0.575	0.0558	0.571	0.000000	0.0912	-7.906	0.0439	105.049	0	0
Natural	73	94	0.448	0.2160	0.612	0.000000	0.0800	-6.106	0.0708	97.773	0	0
Waste It On Me (feat. BTS)	79	82	0.669	0.2520	0.684	0.000000	0.7200	-4.595	0.0585	96.097	0	0
Always Remember Us This Way	81	83	0.553	0.2990	0.502	0.000000	0.7640	-5.972	0.0409	129.976	0	0

Figure 3.4.c

3.5. Unsupervised Learning

Once the classified playlist of 77 songs was created, we performed Clustering, an unsupervised learning method to create different genre playlists. Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). Hence, we used clustering methods to match similar songs to create playlists.

3.5.1. K-means Clustering

The initial method we used was K-means clustering. The first step in creation of clusters is to determine the value of k that will tell help us to create the optimal amount of clusters. The first method we used was the Elbow method to determine k. The elbow method looks at the percentage of variance explained as a function of the number of clusters. One should choose a number of clusters so that adding another cluster doesn't give much better modelling of the data. More precisely, if one plots the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph. From figure 3.5.a we can see the elbow plot of sum of squared distances to cluster centres for k-means method created for this set of data. As you can see from the plot there was no significant point for the elbow, but the closest we could get was for k = 4.

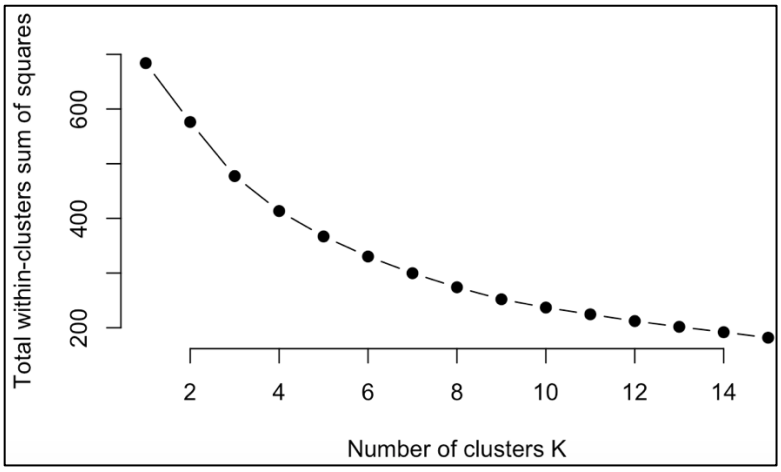


Figure 3.5.a

Thus K-means clustering with k = 4 was performed on the song features(scaled). We can see from TABLE X the centers created. In k-means clustering, each cluster is represented by its center which corresponds to the mean of points assigned to the cluster.

popularity	dance	acoustic	energy	instrumental	liveness	loud	speech	tempo
- 0.064058	-0.311	- 0.3885	0.83298	- 0.1701502	0.0903	0.8127	-0.1022	0.2564
0.0971104	0.615	0.03776	- 0.5066	- 0.1424046	- 0.1424	- 0.3485	-0.2886	-0.550
- 0.3244078	- 0.98	1.14019	- 0.7317	0.7925862	0.7925	- 0.7833	-0.0408	0.6867
0.48399603	1.165	- 0.2671	- 0.9075	- 0.1697797	1.5953	- 1.1253	1.1188	-0.677

Table 3.5

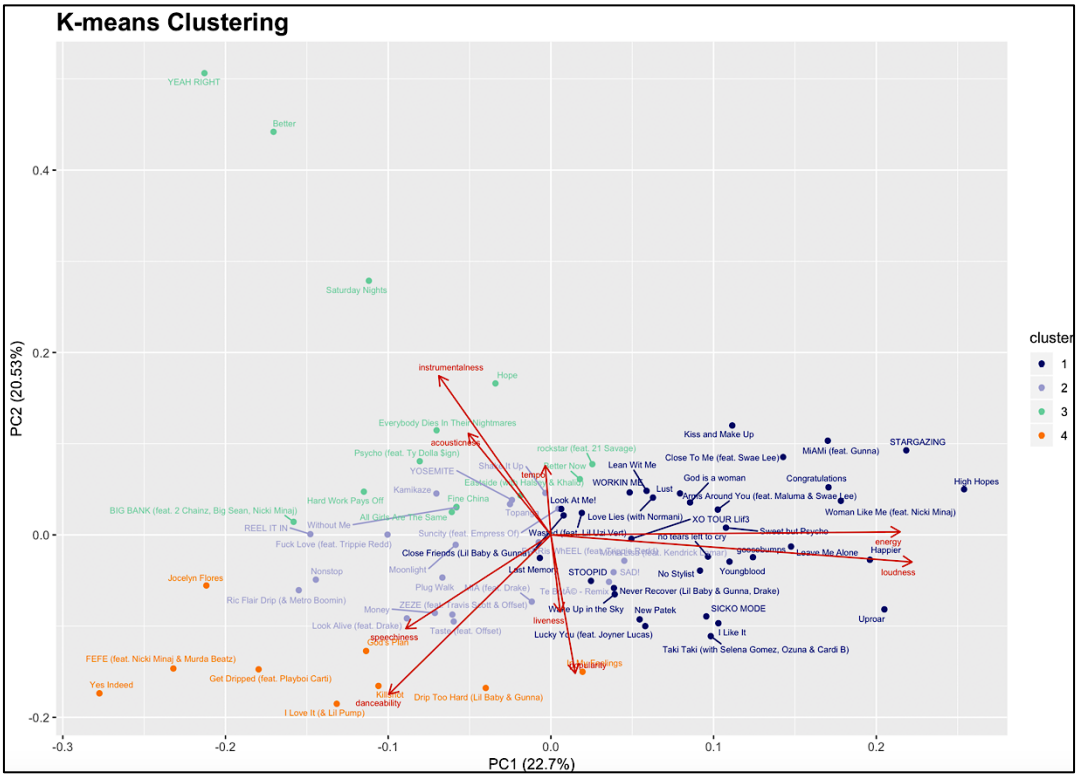


Figure 3.5.b

To describe more about how the results have been clustered, Figure 3.5.b shows the plot of the first 2 PCA components. PCA components are the directions where there is the most variance, the directions where the data is most spread out. The red arrows represent eigenvectors and values that exist in pairs: every eigenvector has a corresponding eigenvalue. An eigenvector is a direction, in the figure 3.5.b the eigenvector is the direction of the line (vertical, horizontal, 45 degrees etc.) . An eigenvalue is a number, telling you how much variance there is in the data in that direction, in the figure X the eigenvalue is a number telling us how spread out the data is on the line. The eigenvector with the highest eigenvalue is therefore the principal component. On close observation of the plot we can deduce that the first principal component accounts for as much of the variability in the data as possible.

As we can see in the figure 3.5.b, the x-axis is PC1 (22.7%) and the y-axis is PC2 (20.53%). These are the first two principal components. The PCA graph shows that PC1 separates songs by loudness/energy vs danceability/speechiness, while PC2 appears to separate songs on a liveliness/popularity vs tempo, acoustics and instrumentalness.

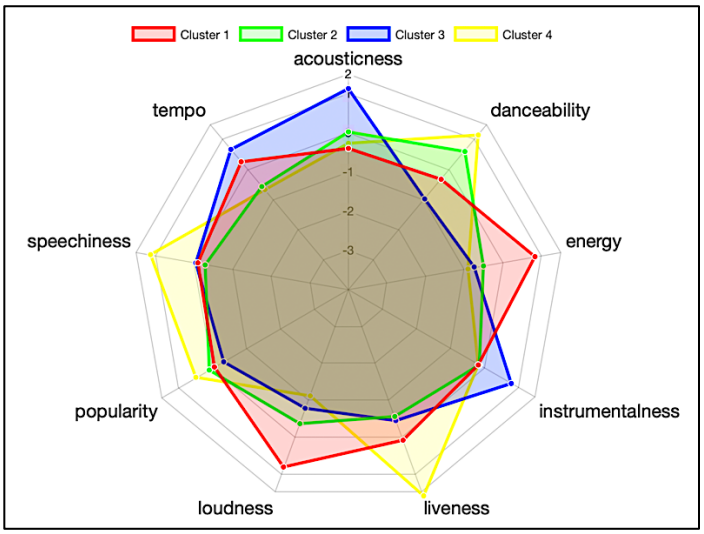


Figure 3.5.c

To get a summarized view about the type of songs clustered we created a radar chart to understand which cluster belongs to what kind of genre. From figure 3.5.c we can see that the k-means clustering gives us excellent results to create 3 significant playlists. But it does not define cluster 2 shown in green to be part of a specific category of music. Hence there arose a need to try out another clustering method.

3.5.2. K-medoids clustering

For the second cluster analysis we chose this method because k-medoids is more robust to noise and outliers as compared to k -means because it minimizes a sum of general pairwise dissimilarities instead of a sum of squared distances.

Similar procedure was followed to create clusters using k-medoids. Since in the previous method the optimal value of k was not accurately found, we used 2 different methods. The first method was evaluating the silhouette coefficient. The silhouette coefficient measures how similar a data observation is to those on its own clusters (cohesion), compared with the observations in the other clusters (separation). The value of the silhouette ranges from -1 to 1, where 1 indicates a good cohesion, and -1 indicates that the object is probably in the wrong cluster. From figure 3.5.d we can see the range of values of silhouette coefficient for different values of k. The highest values 0.55 and 0.44 are for k = 2 and k = 3 respectively. Since we wanted to create different genre playlists, k = 2 would not be as helpful. Hence k = 3 was chosen.

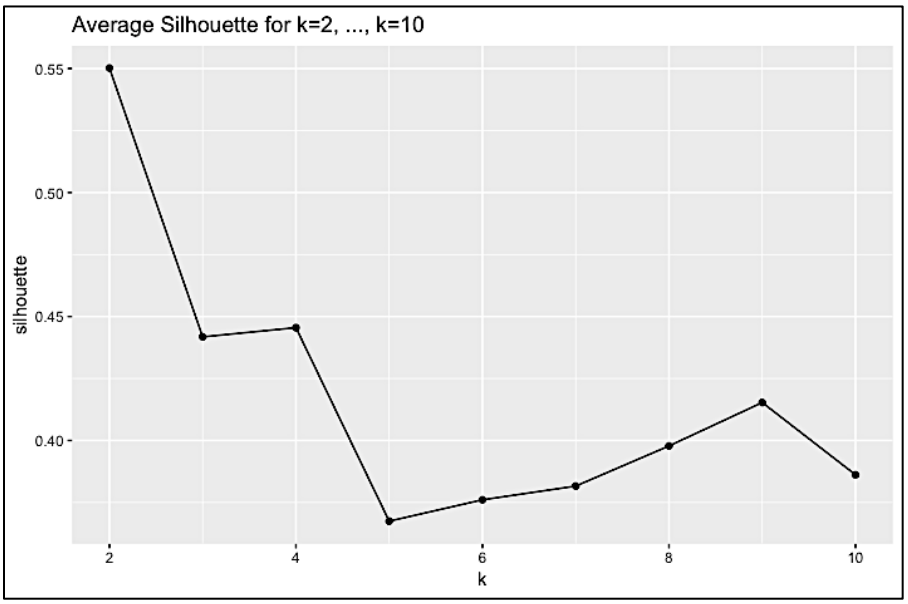


Figure 3.5.d

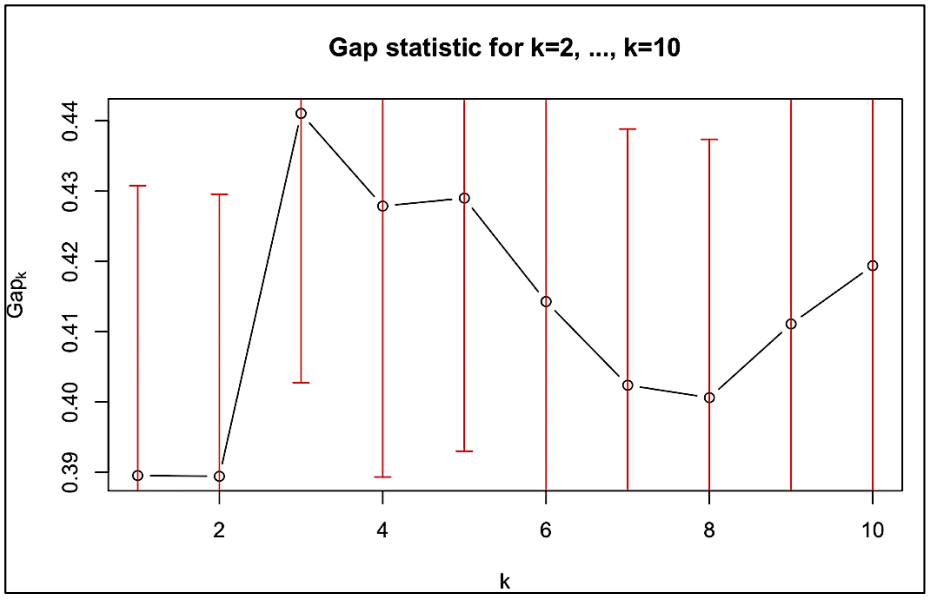


Figure 3.5.e

The second method used was evaluating the gap statistic. The gap statistic compares the total within intra-cluster variation for different values of k with their expected values. To correctly identify the value of k we see the correct elbow point like we did in the elbow plot. From the figure 3.5.e we can see that k=3 and k=4 are the suitable values. After evaluating both the methods we decided k = 3 as the ideal choice for the clusters.

From figure 3.5.f we can see the Silhouette plot of the clusters created by k-medoids clustering technique. The numbers on the right of the image are the average silhouette of each cluster. Cluster 3 has the lower silhouette (0.27), while cluster 2 has the best (0.55). Cluster 1 is also the one with the most observations with a silhouette lower than 0.
According to all the metrics presented in this section, cluster 2 seems to be the best one, because it has the best average intra-cluster distance and diameter, plus the highest silhouette. The points with negative silhouette are songs that are most probably assigned to the wrong cluster. The figure 3.5.g shows the 3 clusters created with the count of songs in each playlist.

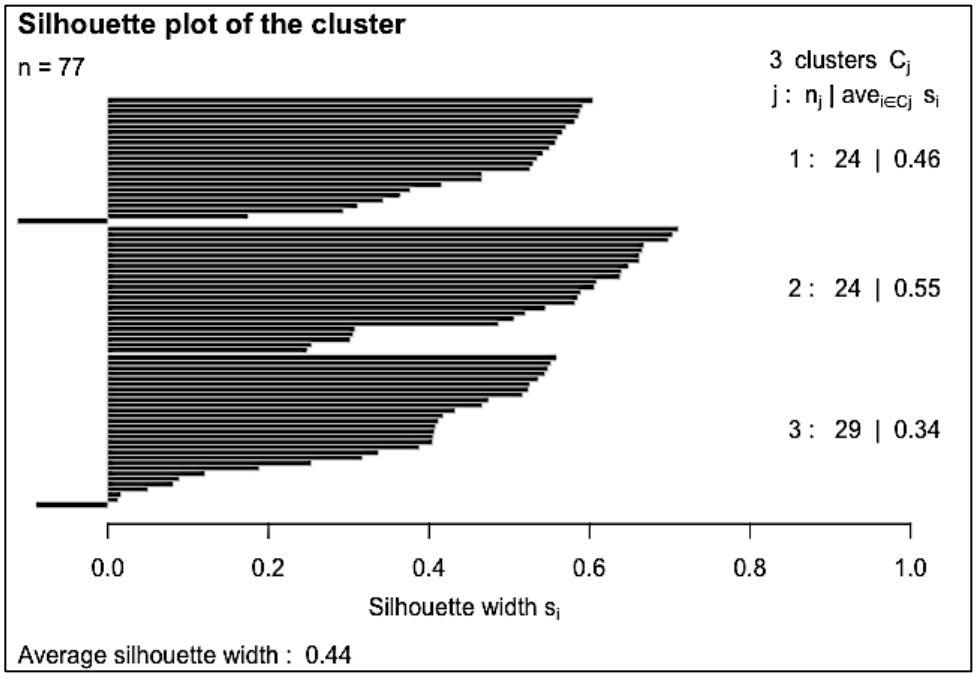


Figure 3.5.f

size	max_diss	av_diss	diameter	separation
24	38.38980	13.481818	53.18065	8.259486
24	20.81241	7.757454	29.22367	4.117094
29	48.84549	11.886105	61.08781	4.117094

Figure 3.5.g

3.5.3. Dimension Reduction

As we could see that K-medoids was successful in creating the clusters. But there were still few outliers that had to be removed. To take a closer look at the clusters we performed dimension reduction techniques and studied their plots to get exact count of the outliers (songs not belonging to a certain playlist).

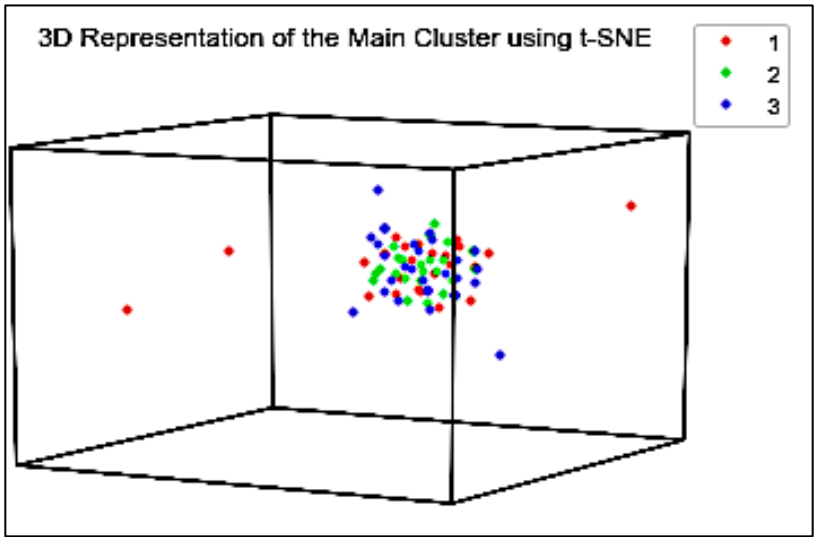


Fig 3.5.h

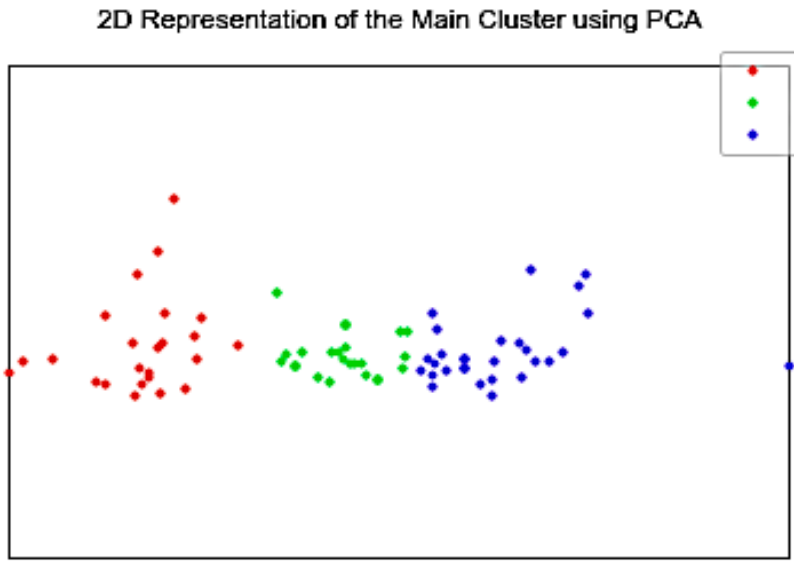


Fig 3.5.i

The two plots above represent the clusters using t-SNE and PCA methods. The t-Distributed stochastic neighbour embedding (t-SNE) minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding. By looking at both the plots we can see that cluster 1 has 3 songs that are outliers and cluster 3 has 2 outliers.

The next phase of naming the genre of playlists was done manually by listening and comparing those songs to their original genre. As a result, we 3 playlists were created with the names HIPHOP, RNB, TRAP. From the first playlist 3 songs - Happier, Taki Taki, Eastside were the outliers and God is a woman, Woman Like me were the outliers from cluster 3. However, a close look at cluster 2 gave us one song - no tears left to cry as an outlier although the clustering showed that cluster 2 was perfect.

4. Results

- As we can see in figures 3.4.b and 3.4.c, the songs have been classified as 0 and 1, where the column 'class' indicates the manual pre-assigned class and 'new class' indicates the class assigned by KNN algorithm.
- Firstly, the main analysis that we can make is that new class 0 has been assigned to all dance/pop songs present in the testing dataset.
- Another interesting result that we found was that the song 'Nice for what' by our favourite artist (Drake), which was initially assigned class = 1, was classified into new_class = 0 by the algorithm. This is correct since the song was very unlike the artist's style, and essentially is very different from all his other songs.
- The tables below depict the playlists generated by K-medoids clustering technique: -

HIPHOP

name
Drip Too Hard (Lil Baby & Gunna)
Taki Taki (with Selena Gomez, Ozuna & Cardi B)
Happier
I Love It (& Lil Pump)
In My Feelings
ZEZE (feat. Travis Scott & Offset)
MIA (feat. Drake)
Arms Around You (feat. Maluma & Swae Lee)
Better
SAD!
Taste (feat. Offset)
Eastside (with Halsey & Khalid)
God's Plan
Suncity (feat. Empress Of)
Topanga
Kiss and Make Up
Uproar
High Hopes
Ric Flair Drip (& Metro Boomin)
Te BotÃ© - Remix
Plug Walk
Kamikaze
Killshot
FeRRis WhEEL (feat. Trippie Redd)

RNB

name
Without Me
Moonlight
FEFE (feat. Nicki Minaj & Murda Beatz)
Never Recover (Lil Baby & Gunna, Drake)
Money
Yes Indeed
I Like It
Jocelyn Flores
Fuck Love (feat. Trippie Redd)
Mona Lisa (feat. Kendrick Lamar)
Psycho (feat. Ty Dolla $ign)
Hard Work Pays Off
Last Memory
REEL IT IN
Youngblood
Look Alive (feat. Drake)
Everybody Dies In Their Nightmares
YOSEMITE
Look At Me!
Congratulations
Sweet but Psycho
no tears left to cry
Get Dripped (feat. Playboi Carti)
goosebumps

TRAP

name
Nonstop
Better Now
SICKO MODE
Fine China
Wake Up in the Sky
No Stylist
Leave Me Alone
New Patek
BIG BANK (feat. 2 Chainz, Big Sean
STOOPID
All Girls Are The Same
rockstar (feat. 21 Savage)
Lean Wit Me
Lucky You (feat. Joyner Lucas)
WORKIN ME
Woman Like Me (feat. Nicki Minaj)
STARGAZING
Saturday Nights
Wasted (feat. Lil Uzi Vert)
XO TOUR Llif3
God is a woman
Love Lies (with Normani)
Hope
Close To Me (feat. Swae Lee)

5. Conclusion

In this project we obtained our dataset of our favourite artists' songs by scraping data from Spotify. We further used classification techniques to determine KNN as the best classifier to find similar songs to our favourite artists. To create playlists various clustering techniques were used and K-medoids proved to be the most apt method to give us the 3 clusters. These 3 clusters were then manually assessed to remove outliers created in the clustering process. As a result, 3 playlists were created that belonged to different genres which keeping the style of songs similar to our favourite artists. Such recommendation systems will be highly appreciated by those audience that love a particular artist's music and will benefit from listening to their style of songs.

6. Future Research

The analysis presented in this project can be used and developed further in many ways. In legal cases where intellectual property disputes take place, such kind of a study can be useful to understand the origin of a style of a song. Moreover, streaming services can find new talent, which is often difficult to find as these days' songs popularity is due to artists' popularity rather than the quality of their music.

7. References

[1] https://www.sas.com/en_us/insights/analytics/machine-learning.html
[2] https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d
[3] https://github.com/plamere/spotipy/blob/master/docs/index.rst
[4] https://www.geeksforgeeks.org/dimensionality-reduction/
[5] https://scikitlearn.org/stable/modules/generated/sklearn.decomposition.PCA.html
[6] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[7] https://machinelearningmastery.com/estimate-performance-machine-learning-algorithms-weka/
[8] https://scikit-learn.org/stable/modules/grid_search.html#grid-search
[9] https://en.wikipedia.org/wiki/Cross-validation_(statistics)
[10] Feng, Shanshan, et al. "Personalized Ranking Metric Embedding for Next New POI Recommendation." *IJCAI*. Vol. 15. 2015.
[11] Song, Yang, Lu Zhang, and C. Lee Giles. "Automatic tag recommendation algorithms for social recommender systems." *ACM Transactions on the Web (TWEB)* 5.1 (2011): 4.
[12] Song, Yang, Lu Zhang, and C. Lee Giles. "Automatic tag recommendation algorithms for social recommender systems." *ACM Transactions on the Web (TWEB)* 5.1 (2011): 4.
[13] Chen, Hung-Chen, and Arbee LP Chen. "A music recommendation system based on music data grouping and user interests." *Proceedings of the tenth international conference on Information and knowledge management*. ACM, 2001.
[14]https://medium.com/s/story/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe