Music Recommendation System Using Clustering

Aicha Slaitane[*], Chaelin Lee[*]

Abstract

Recommending songs for the users aroused attention among researchers, as the technology for personalization has developed. In this paper, we implement a music recommendation system using clustering methods. We first evaluate some clustering algorithms that do not take k (the number of clusters) as an input hyperparameter and then evaluate those that build clusters based on a given number of k clusters. We use three evaluation metrics: Davies-Bouldin Score, Silhouette Score, and Calinski-Harabasz index. We conclude that the Davies-Bouldin Score is the best evaluation metric for our problem and K-Means is the optimal clustering algorithm. Finally, we present a Music Recommendation System based on the trained optimal algorithm.

1 Introduction

Music plays a pivotal role in people's lives, accompanying them during various activities such as work, study, or exercise. According to Rentfrow et al.[8], music often takes precedence over other pastimes like watching television, reading books, or watching movies.

In the wake of technological advancements, personalized music recommendation systems have gained prominence. Many companies harness the abundance of data and rapid information growth to deliver tailored recommendations, thereby gaining a competitive edge and optimizing their services. For instance, Spotify leverages user music preferences and listening habits to provide individualized music suggestions through recommendation systems [7].

Markus Schedl categorizes recommendation systems into three main types: collaborative filtering, content-based filtering, and hybrid recommendation systems [9]. Collaborative filtering approaches exploit user-item interactions like clicks or ratings to offer recommendations based on similarities with other users. There are two primary types of collaborative filtering: user-user collaborative filtering and item-item collaborative filtering. In contrast, content-based filtering primarily relies on textual descriptors of content, such as metadata, user-generated tags, or reviews, to generate recommendations. Hybrid recommendation systems strike a balance between collaborative and content-based filtering methods to provide users with well-rounded recommendations.

Our work aims to provide users with highly personalized and contextually relevant song recommendations through a content-based music recommendation system. Through extensive model development and rigorous testing, we aspire to identify the optimal solution that offers superior recommendation performance, enhancing the user's music listening experience.

[*]Duke Kunshan University, Email: {aicha.slaitane, chaelin.lee}@dukekunshan.edu.cn

2 Background

Recent trends in recommendation systems have led to an emphasis on real-time updates and the incorporation of various variable types. In the collaborative-based recommendation system, one approach to gathering user data is the utilization of questionnaires[2]. These questionnaires employ clustering methods to group users with similar musical preferences. An alternative methodology involves the application of recurrent neural networks (RNN) for comparing songs based on their similarity [5]. Rather than clustering users, this research focuses on evaluating the likeness between songs. To further enhance the recommendations, a fast content-based music information retrieval (CBMIR) approach is employed, which draws from online sources to provide song similarities [4]. A recent noteworthy contribution in the field is the Tunes Recommendation System [3], which combines collaborative and content-based filtering within a deep learning classification model. This novel approach solicits input from listeners, consisting of 9 songs they have heard and a 10th song for which they express a likelihood of preference. The model computes a score from 0 to 1, signifying the listener's probability of liking the 10th song based on the initial 9. Given its content-based filtering approach, this research represents a promising avenue, particularly when detailed song content is available.

3 Problem

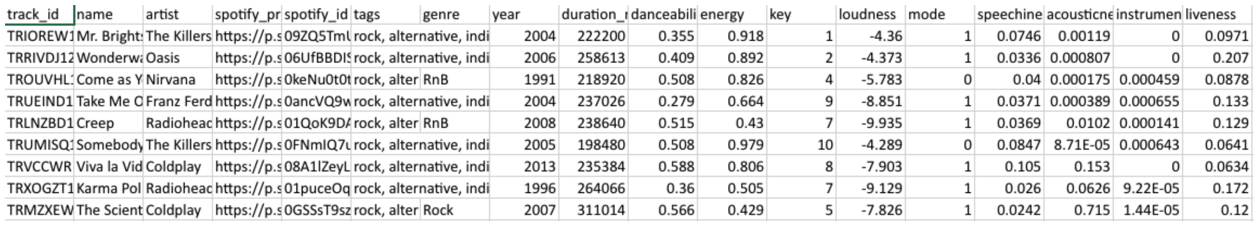
Collaborative filtering is the most widely adopted technique in music recommendation systems, mainly due to its ability to provide recommendations without necessitating explicit content descriptions, making it highly versatile [6]. Additionally, it effectively distributes the workload associated with evaluating and annotating items within its database. Nevertheless, this approach encounters several challenges including data sparsity and issues related to identifying similar items.

This paper sets forth a clear research objective - to use content-based filtering to develop a recommendation system with several clustering techniques and to find the best technique for the model. Clustering offers a promising avenue to address the limitations of current models by enhancing the grouping of songs with similar characteristics, thus leading to more refined and accurate recommendations.

4 Method

4.1 Dataset

As mentioned above, our work focuses on building a Music Recommendation System using clustering methods. This paper evaluates both clustering algorithms that take the number of clusters 'k' as an input hyperparameter and those that do not. We train our models on a subset of the Million Song Dataset, a freely available collection of data for one million contemporary songs. This dataset contains a collection of audio features and metadata for the songs. The features include the duration of the song, danceability, mode, tempo, energy, etc, and some additional meta-data such as Spotify URLs and the playlist's number of followers. The dataset looks like the following:



track_id	name	artist	spotify_pr	spotify_id	tags	genre	year	duration_i	danceabili	energy	key	loudness	mode	speechine	acousticne	instrumen	liveness
TRIOREW1	Mr. Bright	The Killers	https://p.s	09ZQ5Tml	rock, alternative, indi		2004	222200	0.355	0.918	1	-4.36	1	0.0746	0.00119	0	0.0971
TRRIVDJ12	Wonderwa	Oasis	https://p.s	06UfBBDIS	rock, alternative, indi		2006	258613	0.409	0.892	2	-4.373	1	0.0336	0.000807	0	0.207
TROUVHL1	Come as Y	Nirvana	https://p.s	0keNu0t0t	rock, alter	RnB	1991	218920	0.508	0.826	4	-5.783	0	0.04	0.000175	0.000459	0.0878
TRUEIND1	Take Me O	Franz Ferd	https://p.s	0ancVQ9w	rock, alternative, indi		2004	237026	0.279	0.664	9	-8.851	1	0.0371	0.000389	0.000655	0.133
TRLNZBD1	Creep	Radiohead	https://p.s	01QoK9DA	rock, alter	RnB	2008	238640	0.515	0.43	7	-9.935	1	0.0369	0.0102	0.000141	0.129
TRUMISQ1	Somebody	The Killers	https://p.s	0FNmIQ7u	rock, alternative, indi		2005	198480	0.508	0.979	10	-4.289	0	0.0847	8.71E-05	0.000643	0.0641
TRVCCWR	Viva la Vid	Coldplay	https://p.s	08A1lZeyL	rock, alternative, indi		2013	235384	0.588	0.806	8	-7.903	1	0.105	0.153	0	0.0634
TRXOGZT1	Karma Pol	Radiohead	https://p.s	01puceOq	rock, alternative, indi		1996	264066	0.36	0.505	7	-9.129	1	0.026	0.0626	9.22E-05	0.172
TRMZXEW	The Scient	Coldplay	https://p.s	0GSSsT9sz	rock, alter	Rock	2007	311014	0.566	0.429	5	-7.826	1	0.0242	0.715	1.44E-05	0.12

Figure 1: Overview of the dataset

4.2 Data Preprocessing

The data was originally stored in .h5 files. To enhance accessibility, a custom function was developed to extract the required columns from these files and transform them into a CSV format. This CSV file contains a subset of the original dataset as illustrated in Figure 1 which is used in our project. Figure 2 is a visualization of the dataset in two dimensions using Principal Component Analysis (PCA) as the dimensionality reduction method.

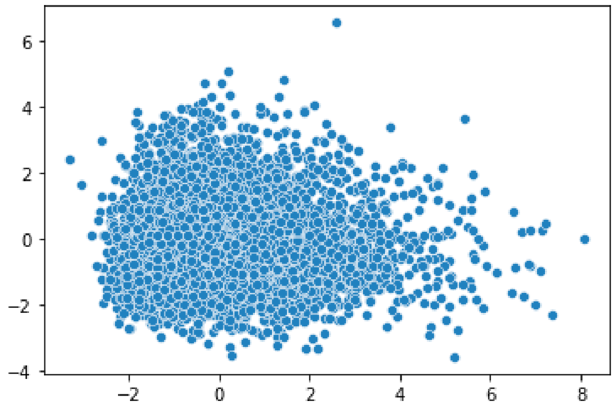


Figure 2: Visualization of the dataset in 2D

Before delving into modeling, we explored the dataset and conducted feature engineering. We calculated the correlation matrix of the features to determine which ones to choose during modeling. Only one feature was highly correlated with two others and therefore is not used in the clustering. The benefit of eliminating highly correlated features is to improve model performance by simplifying the relationships between features and allowing for faster model training.

4.3 Clustering with algorithms that do not take the number of clusters k as a hyper-parameter

First, we evaluated some models that do not take the number of clusters k as a hyperparameter: Affinity Propagation, DBSCAN, and OPTICS.

Affinity Propagation is a clustering algorithm that identifies a set of examples among the data points and forms clusters around them. These exemplars are specific data points that serve as representatives or prototypes for each cluster and best summarize or characterize them. Without tuning any of the algorithm's hyperparameters, we can visualize the following clusters.

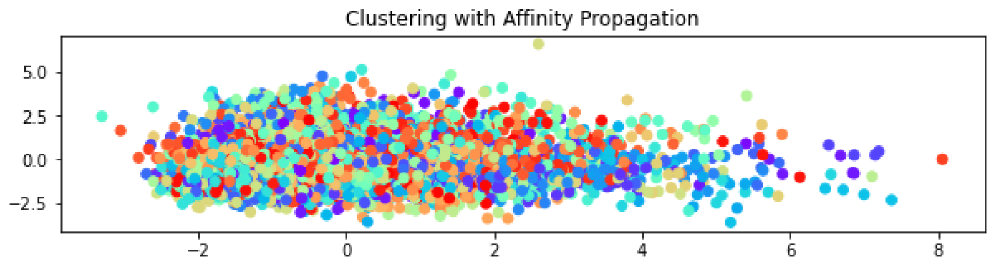


Figure 3: Clustering with Affinity Propagation

As it is not clear whether the algorithm performs well based on Figure 3, we tried to tune the main hyperparameters of Affinity Propagation (damping and preference) and evaluate the resulting models using three evaluation metrics: Silhouette Score, Davies-Bouldin Score, and Calsinki-Harabasz Score. However, we were not able to evaluate them at all as we kept getting errors because the number of the resulting clusters was equal to the number of data points in the training dataset.

The next two clustering algorithms we tested are DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and OPTICS (Ordering Points To Identify the Clustering Structure). Both are density-based algorithms, meaning that they group the points together in areas of high density separated by areas of low density. The following graphs show the resulting clusters with the default hyperparameters.

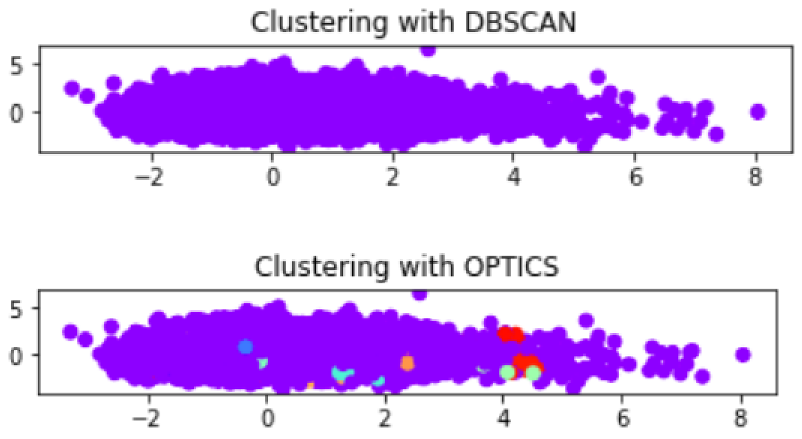


Figure 4: Clustering with OPTICS and DBSCAN

We then tried to tune their hyperparameters but that was not successful as well as we were getting convergence errors. Overall, clustering algorithms without the number of clusters as a hyperparameter are not promising in our project. In the future, we want to work on how to fix the errors we encountered.

4.4 Clustering with algorithms that take the number of clusters k as a hyper-parameter

Next, we evaluated some algorithms that take the number of clusters as a hyperparameter. We chose the following: KMeans, Birch, Agglomerative Clustering, and Gaussian Mixture. Birch and Agglomerative are from the Hierarchical Clustering family. Gaussian Mixture is based on probability analysis. It allows for some data points to be in different clusters at the same time which could be the case for songs. For all of these algorithms, we run experiments to determine the optimal k value and evaluate the resulting models using Silhouette Score, Davies-Bouldin Score, and Calinski-Harabasz Score. We also determined which evaluation metric best evaluates the models.

5 Computational Results

There are no results to be shown for the first set of experiments conducted for algorithms without the number of clusters as an input hyperparameter because of the errors we got as explained above.

The results for the other set of algorithms are shown in Figure 5.

For both the Silhouette Score and Calinski-Harabasz Score, a high value indicates that the data points are well-matched to their own cluster and poorly matched to neighboring clusters. If we try to choose the optimal k value based on them, we would choose k to be 1 (see Figure 5), that is, all data points belong to one cluster, but that is obviously not good. So, both of these evaluation metrics are not going to be chosen for our project. On the other hand, the lower the value of the Davies-BouldinScore the better. We can clearly identify what are the lowest Davies-Bouldin Scores for KMeans, Birch, and Agglomerative, which are:

Algorithm	**Score**
KMeans	1.689589901210743
Birch	2.076127283238973
Agglomerative	1.8674594798769

Table 1: Clustering Algorithm Davies Scores

Their corresponding k values are 11, 24, and 10 respectively. Figures 6-8 show 2D visualization plots of each resulting optimal model.

Given the Davies-Bouldin Scores of the models and the visualization of their optimal representations, we conclude that KMeans is the best model for our problem. The number of clusters depends on the training dataset. In our case, it was 11, but it can change if we are training on a different dataset.

Cluster	Precision Score
0	0.06150717872403289
1	0.04791169237585684
2	0.06237910317577342
3	0.0566110133962638
4	0.06804992027015072
5	0.07191290052984407
6	0.06226570542083079
7	0.06479491207473576
8	0.0665046690928914
9	0.0643030166295834
10	0.06559703531109644

Table 2: Precision score for each Cluster

6 Evaluation

In recommendation system, without A/B testing, it is difficult to estimate the accuracy of the system. Since it's unable to apply a negative score on the recommendation, Mean Average Precision (mAP) was used, in this case. mAP is a commonly used metric to estimate the accuracy of the recommendation system.[10] This metric was used in Million Song Dataset Challenge held by Spotify.

The average precision is calculated for each cluster in the set of recommended songs. From the following table, we can infer that the cluster with the best performance in terms of precision for the given set of user recommendations is 5 with 0.07191290052984407. Our highest value is less than 0,17144, the value with the highest mAP by the winner of the MSD Challenge [1]. However, we can still infer that our model is good enough since it used content-based filtering instead of collaborative filtering.

7 Music Recommendation System

This part brings together all the results of earlier sections. We concluded that KMeans was the best clustering algorithm to be used for our Music Recommendation System based on the Davies scores of the tested models. KMeans is generally computed as:

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

Where:

$$\begin{aligned} C &: \text{Set of clusters (partitions of the data)} \\ k &: \text{Number of clusters} \\ \mathbf{x} &: \text{Data point} \\ \boldsymbol{\mu}_i &: \text{Mean of data points in cluster } C_i \end{aligned}$$

The structure of the obtained Music Recommendation System is as follows:

1. Train KMeans on the training dataset using the optimal number of clusters k based on the lowest value of the Davies-Bouldin Score.
2. Calculate the 'mean history' by taking the mean of the features of the last 'n' songs in the user's music history.
3. Assign 'mean history' to a cluster, which will serve as a hard boundary for song recommendations.
4. Calculate the similarity between 'mean history' and songs in the cluster using metrics such as Cosine similarity (two vectors A and B are more similar as the cosine value approaches 1) and Euclidean distance.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Where:

$\cos(\theta)$ is the cosine of the angle between vectors $\mathbf{A}$ and $\mathbf{B}$.

$\mathbf{A}$ is one of the vectors being compared.

$\mathbf{B}$ is the other vector being compared.

$\|\mathbf{A}\|$ is the Euclidean norm (magnitude) of vector $\mathbf{A}$.

$\|\mathbf{B}\|$ is the Euclidean norm (magnitude) of vector $\mathbf{B}$.

5. Select 's' songs with high similarity scores to recommend songs that closely match the user's music history.
6. Select 's' songs with low similarity scores to provide recommendations that are less similar but still within the same cluster as 'mean history.'
7. Recommend the 's' selected songs to the user for an improved music listening experience.

8 Conclusion

Music recommendation systems play a pivotal role in our daily lives, enriching our music-listening experiences. Continuous research in this domain aims to refine the recommendation approach. In our paper, we employed K-Means to build a content-based music recommendation system using metadata and music audio features. For future work, we want to focus on overcoming the issues raised with clustering algorithms that do not take k as an input hyperparameter.

References

[1] Fabio Aiolli. A preliminary study on a recommender system for the million songs dataset challenge. volume 964, 01 2013.

[2] Jiakun Fang, David Grunberg, Simon Luit, and Ye Wang. Development of a music recommendation system for motivating exercise. *2017 International Conference on Orange Technologies (ICOT)*, pages 83–86, 2017.

[3] Ferdos Fessahaye, Luis Perez, Tiffany Zhan, Raymond Zhang, Calais Fossier, Robyn Markarian, Carter Chiu, Justin Zhan, Laxmi Gewali, and Paul Oh. T-recsys: A novel music recommendation system using deep learning. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6, 2019.

[4] Takahiro Hayashi, Nobuaki Ishii, Masato Ishimori, and Koji Abe. Stability improvement of indirect matching for music information retrieval. *2015 IEEE International Symposium on Multimedia (ISM)*, pages 229–232, 2015.

[5] Miao Jiang, Ziyi Yang, and Chen Zhao. What to play next? a rnn-based music recommendation system. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 356–358, 2017.

[6] Paolo Massa and Paolo Avesani. Trust-aware collaborative filtering for recommender systems. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 492–508, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[7] Martijn Millecamp, Nyi Nyi Htun, Yucheng Jin, and Katrien Verbert. Controlling spotify recommendations: Effects of personal characteristics on music recommender user interfaces. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, UMAP '18, page 101–109, New York, NY, USA, 2018. Association for Computing Machinery.

[8] P. J. Rentfrow and S. D. Gosling. The do re mi's of everyday life: the structure and personality correlates of music preferences. *Journal of Personality and Social Psychology*, 84:1236–1256, 2003.

[9] Markus Schedl. Deep learning in music recommendation systems. *Frontiers in Applied Mathematics and Statistics*, 5, 2019.

[10] Gurpreet Singh, Vishal Kashyap, and Jaskirat Singh. Hybrid music recommendation using k-means clustering. 2017.

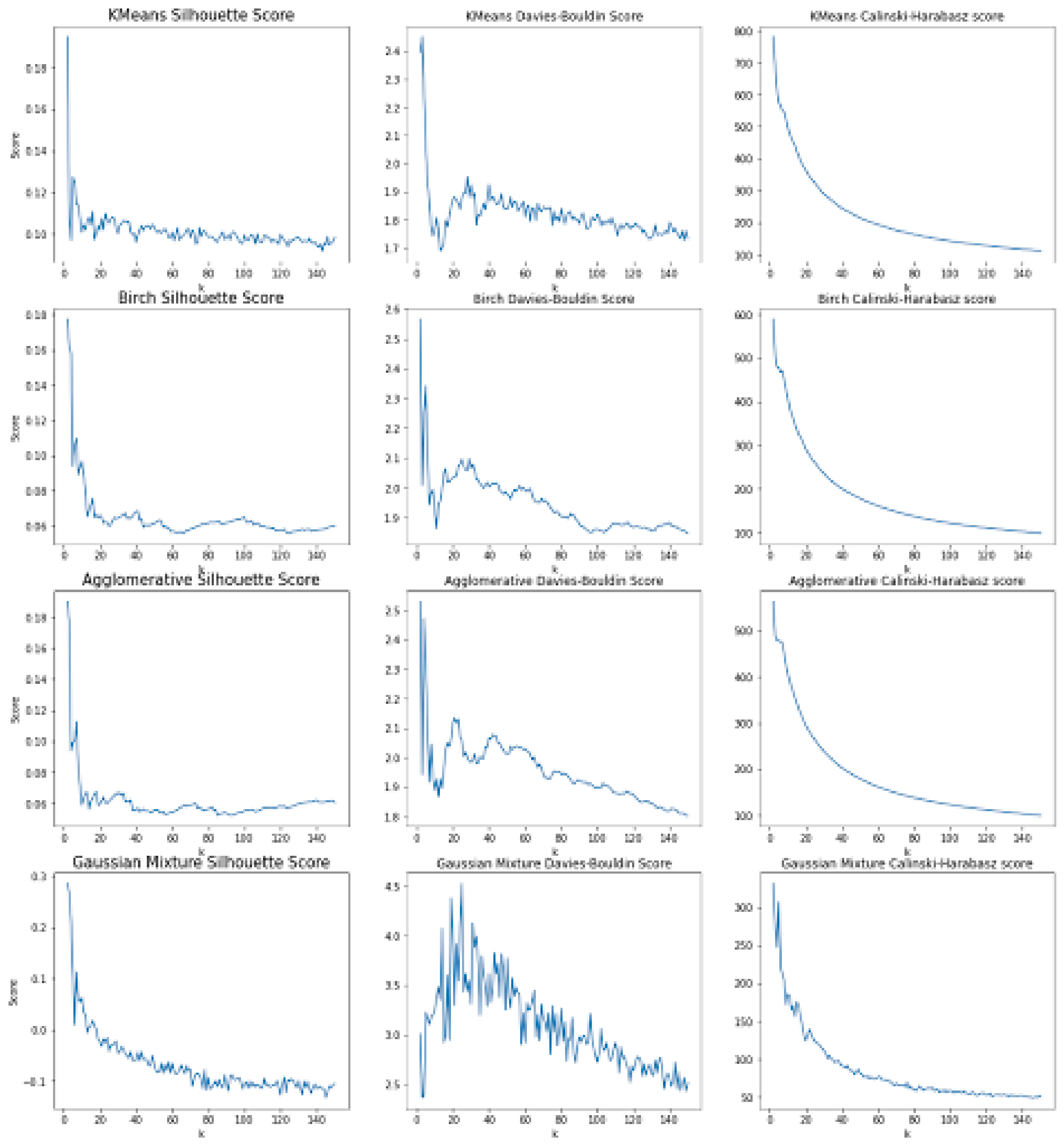


Figure 5: Results for clustering algorithms with k

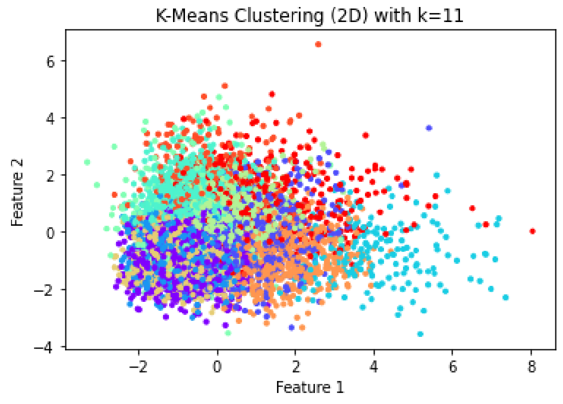


Figure 6: K-Means Clustering Graph in 2D

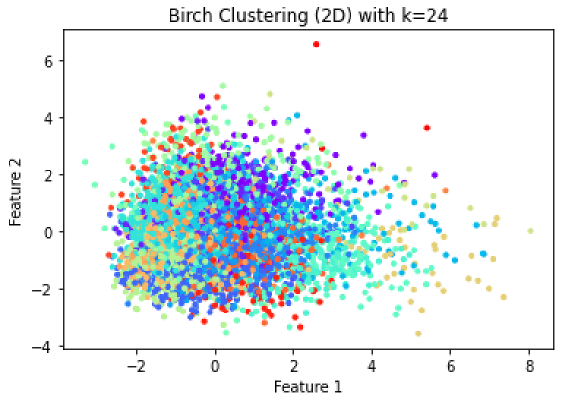


Figure 7: Birch Clustering in 2D

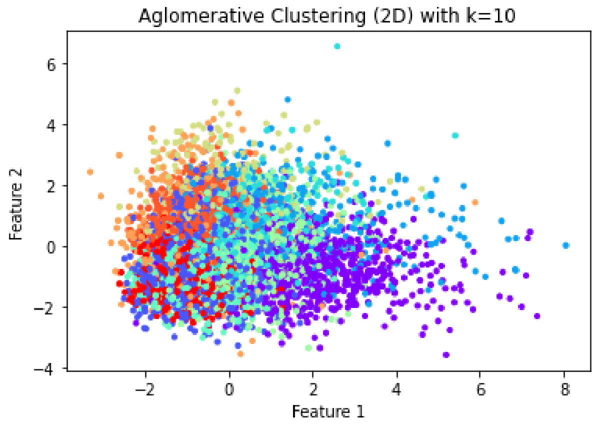


Figure 8: Agglomerative clustering in 2D