INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Developing and Comparing Machine Learning Algorithms for Music Recommendation

Yoon-Teck Bau [a,*], Puteri Ainna Ezzurin Mohd Reza [a], Kian-Chin Lee [a]

Faculty of Computing and Informatics, Multimedia University, Cyberjaya, Selangor, 63100, Malaysia
Corresponding author: ytbau@mmu.edu.my

***Abstract*—The increasing prevalence of song skipping in music streaming applications negatively impacts user satisfaction and subscription retention. Dissatisfaction often arises when users encounter songs they actively dislike, highlighting a gap between user expectations and the value offered by these services. To address this, music recommendation algorithms were researched and developed. Initially, data collection is started. Data collection is through the Spotify application programming interface. This initiation step sets the stage for subsequent exploratory data analysis. Exploratory data analysis examined the collected data to plot a bar chart for total songs released over the years, plot a bar chart for the popularity of songs based on the year it is released, visualize word cloud on frequently mentioned music genres, chart count plot for explicit songs, and chart count plot for song modalities. Data preprocessing involved cleaning the data, handling missing values, and checking for null values to prepare the application of machine learning algorithms. Four machine learning algorithms were applied, k-means, mini-batch k-means, Gaussian mixture, and density-based spatial clustering of applications with noise (DBSCAN), to analyze musical features like rhythm, tempo, and other relevant music attributes. The results showed that the k-means outperforms all other algorithms evaluated regarding recommendation quality, as measured by the Calinski-Harabasz score. Based on the evaluation, the best machine learning will then be applied to identify similarities between songs and be used to generate music recommendation results.**

***Keywords*—Machine learning algorithms; music recommendation; Spotify API; K-means; mini batch k-means; gaussian mixture.**

Manuscript received 10 Apr. 2024; revised 28 Jun. 2024; accepted 11 Oct. 2024. Date of publication 30 Nov. 2024.

I. INTRODUCTION

The rapid expansion of music streaming has changed how we enjoy music, but keeping users happy and maintaining subscriptions are still significant challenges. A big factor is a song skipping, where users frequently encounter tracks they dislike, interrupting the smooth, personalized experience they expect. This frustration often leads to canceled subscriptions, as users feel the service needs to meet their needs. At the heart of the problem is the limitation of current music recommendation systems. While they provide suggestions, these systems often need to fully capture each listener's unique taste, resulting in a playlist filled with songs that do not hit the mark. This research proposes a more personalized approach to music recommendation. By aligning recommendations more closely with individual preferences and reducing exposure to disliked tracks, we aim to improve the listening experience, minimize frustration, and, ultimately, encourage long-term subscription loyalty in the music streaming world.

The research question is: How will this research develop a music recommendation that leverages the best algorithm analyzing song similarities based on musical features such as energy, tempo, and listening history to generate personalized recommendations, minimize song skipping, and enhance user satisfaction within music streaming services? By effectively addressing this question, this research will contribute to a more enjoyable and valuable music streaming experience for users, thereby boosting the success of music streaming service providers.

A. Music Recommendation and Existing Algorithms

Recommender systems have undergone extensive development over the years and have been implemented across a range of problem domains: health [1], [2], [3], [4], [5] , e-commerce [6], [7], [8], [9], tourism [10], [11], [12], [13], movie [14], [15], [16], [17], and more [18], [19], [20].

Music recommendations have played a significant role in the streaming industry, aiming to bring listeners a genuinely personalized experience based on their unique tastes and

listening habits. These systems analyze the music and each user's preferences, using sophisticated algorithms to suggest songs that feel like a natural fit. Thanks to these advancements, listeners now receive recommendations beyond the top charts, matching their style and mood and making music discovery more accessible and enjoyable.

Existing literature has extensively explored the use of recommendation algorithms in music recommendation systems. Referring to Fig. 1, there are four categories of recommendation algorithms which are collaborative filtering, content-based, hybrid, and machine-learning [21], [22]. They have been widely studied and implemented in various music recommendations.

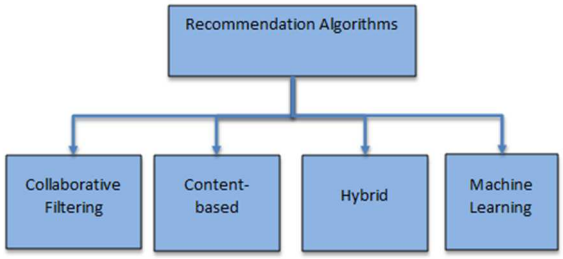

Fig. 1 Categories of Recommendation Algorithm

A collaborative filtering algorithm analyses user behavior and preferences to recommend music that similar users have enjoyed. Content-based algorithms, on the other hand, focus on the characteristics of the music itself to make recommendations. For example, genre, tempo, and rhythm. To exploit the strengths of both techniques, the hybrid method combines collaborative filtering and a content-based algorithm. Machine learning algorithms like K-means are utilized within collaborative filtering, content-based, and hybrid recommendation algorithms to learn patterns and relationships. This section explores the existing recommendation algorithms and their effectiveness.

1) Collaborative Recommendation Filtering: Collaborative filtering is one of the most widely used approaches to music recommendation algorithms. Collaborative filtering is an approach to filtering items by considering the preferences of similar users, aiming to recommend items that align with a particular user's tastes [23]. There are two types of collaborative filtering: user-based and item-based.

An active user for whom this recommendation is directed in a user-based collaborative filtering algorithm. The collaborative filtering algorithm will first search for similar users. This user shares active users' rating patterns with other users. Collaborative filtering is based on this similarity in factors such as history, preferences, and user decisions when purchasing, viewing, or enjoying content. For example, in Fig. 2 below, if user A's preferences for fruits match the highest number of fruits to user C's, they have similar interests. Therefore, user C will also enjoy mango and grapes that the user has yet to taste, as user A prefers.

In the item-based approach, it is also called item-item collaborative filtering. This is a type of recommendation algorithm that considers the similarity between items to recommend products. Illustrating the mechanics of item-based collaborative filtering through an example. Fig. 3 below consists of three individuals: A, B, and C. User A has expressed preferences for mango, pineapple, and grape, while user B enjoys pineapple and grape. On the other hand, user C's liking extends to pineapple and banana. In item-based collaborative filtering, the algorithm identifies item similarities based on shared preferences between two users.

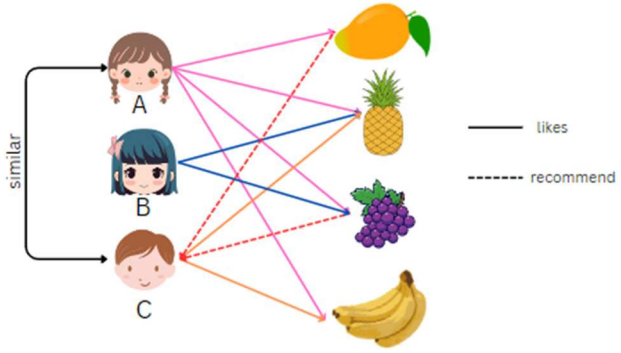

Fig. 2 User-based collaborative filtering approach

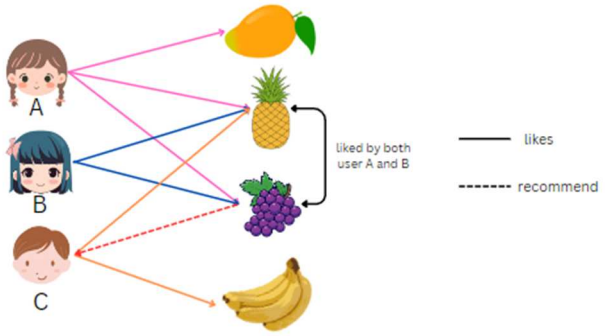

Fig. 3 Item-based collaborative filtering approach

Now, when considering recommendations for user C, the algorithm leverages the similarity between pineapple and grape, which are liked by both user A and user B. Since user C likes pineapple but has not been exposed to grapes, the item-based collaborative filtering algorithm recommends grapes to user C. This recommendation is grounded in the assumption that if two items are favored by other users, introducing an item liked by other users but not yet experienced by one user can enhance the recommendation experience.

One of the primary benefits of these recommendation algorithms is their simplicity in delivering tailored material and their flexibility in accommodating evolving user preferences. However, conventional collaborative filtering approaches have drawbacks, especially when it comes to resolving the cold-start issue and dataset sparsity.

Fareed et al. [24] involves integrating social connections among users into the collaborative filtering algorithm. The performance metric used for evaluating their recommendation algorithm is precision. Their proposed system is in its application, which is in the movie recommendation domain.

Tian et al. [22] employ k-means for their recommendation algorithm. Data sparsity arises when the user-item matrix, representing interactions between users and items, is extensive and contains numerous missing data points. In this context, the user-item matrix can become vast and sparse in university library books, where the available book collection

is enormous. The sparsity of the user-book matrix implies that most users have only read a small subset of the overall database, resulting in a high percentage of missing data. K-means is a technique used to group similar objects based on their characteristics. By applying k-means to the user-item matrix, the aim is to group users and books so that objects within the same cluster are more similar to each other than those in other clusters.

The study by Movafegh and Rezapour [25] used hybrid based on a collaborative filtering recommendation algorithm. Two hybrid approaches introduced were k-means with particle swarm optimization and singular value decomposition with a genetic algorithm. Their two proposed methods exhibited performance using three performance metrics: precision, recall, and mean absolute error. Their study did not specify the domain of the recommender system but focused solely on hybrid based on the collaborative filtering approach.

2) Content-based Recommendation Algorithm: In the realm of information retrieval and recommendation systems, content-based algorithms serve as a methodology for constructing recommender systems by considering both user preferences and the attributes of items. Typically, a collection of features distinguishes each item, and subsequent recommendations are generated by evaluating the cumulative value of these features within the user's historical interactions [26]. Unlike collaborative filtering, which relies on user-item interactions and similarities among users, content-based algorithms focus on the attributes and features of items themselves.

Content-based algorithms are widely used in various areas, such as online shopping, music streaming, news, etc. It has a good track record of providing reliable and proper recommendations, particularly for specific or specialized items. The essence of content-based algorithms is creating user profiles and item representations based on relevant features. These features could range from textual content, metadata, or other measurable characteristics describing the items. By leveraging natural language processing techniques, content-based algorithms can discern patterns and preferences, delivering recommendations that align with users' tastes and preferences.

To illustrate this concept, consider an example of how a content-based recommendation system could operate in the context of suggesting fruits. Based on Fig. 4 below, imagine four fruits are available, and a user has already tasted and enjoyed the first two fruits, strawberry and blueberry. The recommendation model automatically suggests the third fruit, raspberry, instead of the fourth, banana, as it is more akin to the first two fruits. This resemblance can be computed based on various features such as the type of fruit, its color, taste, size, and other relevant characteristics.

Content-based filtering offers several advantages, the primary one being avoiding the "cold start" problem. Collaborative filtering can encounter challenges in scenarios where a new website or community needs more users and more user connections. While content-based algorithms do require initial inputs from users to initiate recommendations, they typically provide higher-quality early recommendations compared to collaborative systems. This is because content-based algorithms do not necessitate the accumulation and correlation of many data points before reaching an optimized state.

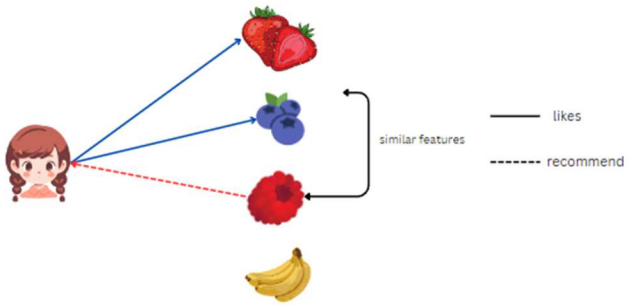

Fig. 4 Recommendation using content-based algorithm

Content-based algorithms ensure transparent recommendations for users, enhancing trust by providing insights into the reasoning behind suggestions. In contrast, collaborative filtering may present instances where users find it challenging to comprehend the basis for specific recommendations. For instance, consider a scenario where a group of users who frequently purchase hiking gear also happens to buy camping tents. A collaborative system might suggest camping tents to users who bought hiking gear, even if they have never expressed interest in or purchased such products.

Another example could be in the context of book recommendations. Suppose a group of users who enjoy mystery novels also happen to buy cookbooks. In that case, a collaborative system might recommend cookbooks to other users who like mystery novels but may have different culinary interests. These examples highlight how content-based algorithms offer a more transparent and understandable recommendation process than collaborative filtering.

Akshay et al. [26] introduced a recommendation algorithm that clusters songs according to implicit properties perceived by the listener, including timbre and acoustics. With song and listener properties, their algorithm could deliver recommendations tailored to the song history and user preferences. Before integrating the k-means algorithm into their recommendation algorithm, the researchers employed the elbow method to determine the number of clusters.

Ahuja et al. [27] advocated the integration of two algorithms, k-means, and k-nearest neighbor, to content-based algorithms in movie recommender systems, aiming to add personalization by considering similar users' preferences. Their two recommendation algorithms leveraged the preferences of users with similar tastes. The Movielens dataset, which comprises a diverse collection of movie ratings and user feedback, is utilized. Root mean squared error was the performance metric to compare their two recommendation algorithms.

3) Hybrid Recommendation Algorithm: A hybrid recommendation system is a unique approach that seamlessly merges content-based and collaborative filtering methods. Combining these two techniques addresses the limitations encountered when utilizing them individually. Implementing hybrid recommender systems spoils choices, allowing for various approaches.

One implementation method involves generating predictions separately using content and collaborative methods and then combining these predictions. Alternatively, the hybrid system can integrate the capabilities of collaborative methods into a content-based approach and vice versa. This hybrid recommendation algorithm increases complexity in adapting to the specific challenges posed by different recommendation scenarios.

Afoudi et al. [23] introduced a recommendation algorithm that employed a hybrid approach, seamlessly integrating collaborative filtering and content-based algorithms. It is then supervised by the ranking list generated by the self-organizing map neural network technique. The approach involved merging scores from all models, capitalizing on the individual strengths of each method.

According to [21], a hybrid recommendation algorithm that integrates collaborative filtering and content-based algorithms based on deep learning is introduced. Their hybrid recommendation algorithm incorporated auxiliary information about users and items into a deep neural network architecture with insufficient historical data. The paper evaluated their only proposed hybrid recommendation algorithm in terms of mean absolute error, mean squared error, and root mean squared error.

II. Materials and Method

A. Machine Learning Algorithms

To implement machine learning to recommend music effectively, it's crucial to explore different algorithms to see which performs best for the dataset and the specific goals of the recommendation. This means testing various approaches, such as k-means, mini-batch k-means, Gaussian mixture, and density-based clustering. By carefully evaluating each algorithm with the right metrics and visualization tools, we can find the one that most accurately picks up on the patterns and similarities in the data. This step-by-step approach ensures that the chosen algorithm will add real value to the music recommendation system, giving users a truly personalized experience.

1) K-Means: K-means are an unsupervised learning method to organize unlabeled datasets into distinct clusters. In this technique, the parameter "k" denotes the predetermined number of clusters to be formed. For instance, setting k=2 will result in two clusters, while k=3 will yield three clusters, and so forth. This approach enables the algorithm to identify autonomously, and group data points into categories without requiring prior training. Functioning as a centroid-based algorithm, k-means assigns each cluster a centroid, with the primary objective being to minimize the distance between data points and their respective centroids. The algorithm iteratively processes the unlabeled dataset, dividing it into k clusters and refining them until optimal clustering is achieved. Notably, the value of k must be specified beforehand for this method.

The k-means algorithm essentially carries out two primary tasks. Firstly, it iteratively determines the optimal locations for k centroids, typically by minimizing the total within-cluster variance through an iterative approach. Secondly, it assigns each data point to the nearest centroid, thereby creating clusters characterized by shared similarities among their constituent data points while maintaining distinctiveness from other clusters.

Before proceeding with implementing k-means on the dataset, the elbow method will be employed to ascertain the optimal number of clusters. The elbow method is generally applied to identify the optimum number of clusters in a dataset. The method derives its name from the shape of the plotted curve, which resembles an elbow or a bend. The fundamental idea behind the elbow method is to evaluate the algorithm performance for different numbers of clusters and observe how the inertia changes with respect to the number of clusters.

The elbow method relies on the within-cluster sum of squares (WCSS), quantifying the total variability within clusters. WCSS, representing the sum of squared distances from each data point to its assigned cluster centroid, encapsulates how tightly grouped the data points are within clusters. For instance, when considering "k" clusters, the formula to compute WCSS involves summing the squared distances of each point to its respective cluster center.

$$WCSS = \Sigma_j^k \Sigma_{P_i \, in \, C_j} \; distance\left(P_i C_j\right)^2 \qquad (1)$$

The elbow method plots the number of clusters against their corresponding inertia values. The inertia typically decreases as the number of clusters increases because more clusters allow the algorithm to fit the data better. However, the decrease in inertia tends to slow down as the number of clusters becomes excessively large. Inertia decreases because adding more clusters may lead to overfitting, where each data point has its cluster, resulting in minimal improvement in algorithm quality. The main objective of the elbow method is to determine at a given point in the plot where the rate of inertia reduction significantly slows and produces an apparent elbow or curve bending. This point indicates a balance between minimizing inertia and avoiding excessive complexity in the clustering model. The optimal number of clusters is often chosen as the value corresponding to this elbow point.

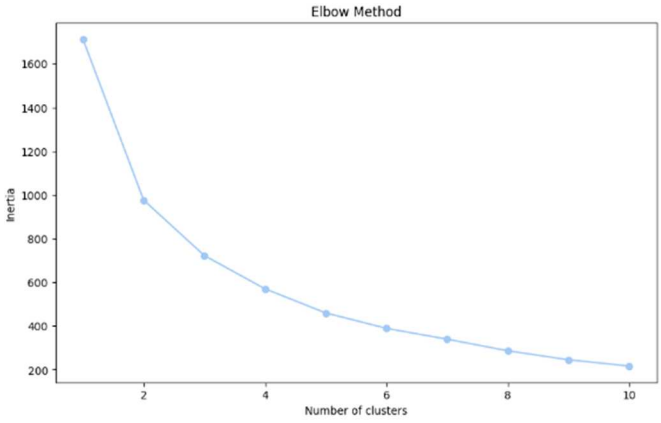

Fig. 5 Example of Elbow Method

As seen in Fig. 5, the optimal number of clusters was determined to be two after using the elbow method to analyze the data. This decision stemmed from identifying a prominent inflection point, or elbow in the plotted curve, indicating a notable deceleration in the rate of decline in WCSS.

2) Gaussian Mixture: Gaussian mixture is a probabilistic model used for clustering data points into multiple groups based on their similarity. Gaussian mixture clustering allows for soft assignment, where data points can belong to multiple clusters with varying membership probability, unlike traditional clustering algorithms that assign each point to only one cluster. This method assumes that the data distribution comprises multiple Gaussian distributions, each representing a distinct cluster. During the clustering process, the algorithm iteratively adjusts the parameters of these Gaussian distributions to maximize the likelihood of the observed data. This optimization process involves estimating the mean and covariance matrix of each Gaussian distribution and the mixing coefficients that determine the relative importance of each distribution. Gaussian mixture clustering is particularly useful when dealing with complex data distributions that a single Gaussian distribution cannot effectively model.

3) Density-based Spatial Clustering of Applications with Noise: Density-based spatial clustering of applications with noise (DBSCAN) is a non-parametric machine learning algorithm that groups closely packed data points while distinguishing outliers in a dataset. It is considered a non-parametric machine learning algorithm because it does not make explicit assumptions about the underlying data distribution. It operates based on the density of data points rather than assuming a predefined number of clusters like k-means. The fundamental concept behind DBSCAN is identifying core, border, and noise points within the dataset.

At the outset, DBSCAN randomly selects a point from the dataset and examines its neighborhood based on a specified distance threshold epsilon (ε). The selected point is labeled as a core point if the number of points within this neighborhood exceeds a predefined threshold. Core points are then used to expand clusters by adding neighboring points within the ε radius. Any point that falls within the ε radius of a core point is considered part of the same cluster. Border points are those within the ε radius of a core point but do not have enough neighbors to be considered core points. These points are assigned to the cluster of their respective core points. Noise points neither belong to any cluster nor meet the criteria to be considered border points. They are typically outliers or points that do not conform to the density characteristics of any cluster. By iteratively expanding clusters from core points and merging neighboring clusters, DBSCAN effectively delineates clusters of varying shapes and sizes without requiring prior knowledge of the number of clusters in the dataset.

4) Mini-batch k-means: Mini-batch k-means represent a modification of the k-means algorithm tailored for enhanced efficiency with a large dataset. Unlike the original k-means algorithm, which handles the entire dataset simultaneously, mini-batch k-means operates on small batches of data iteratively. This approach mitigates the computational expenses associated with processing large datasets. This makes it particularly suitable for datasets that are too large to fit into memory or that have high-dimensional feature spaces. The fundamental concept behind mini-batch k-means revolves around updating cluster centroids by utilizing subsets, or mini-batches, of the input data rather than the entire dataset in each iteration. This strategy aims to expedite convergence and alleviate computational burdens, particularly in scenarios where the dataset exceeds memory constraints. With mini-batch k-means, a new random sample of the dataset is selected for cluster updates in each iteration, repeating until convergence is achieved. Each mini-batch contributes to cluster updates through an approximate combination of prototypes and data results governed by a decreasing learning rate over iterations. This learning rate diminishes inversely with the number of iterations, signifying a reduced impact of new data on cluster adjustments as the algorithm progresses. Convergence occurs when consecutive iterations yield no further changes in cluster assignments. Despite the computational time savings offered by the algorithm, there are potential trade-offs that may result in a decline in cluster quality.

B. Performance Metric

In evaluating machine learning algorithms, one commonly used performance metric is the Calinski-Harabasz score [28]. This score, alternatively referred to as the variance ratio criterion, computes the ratio between the sum of dispersion among clusters and the dispersion within clusters for a specific clustering solution. The Calinski-Harabasz score measures clustering quality and evaluates how well-separated the clusters are. It's calculated by dividing the dispersion among clusters by the dispersion within clusters. The higher the Calinski-Harabasz score, the better the solution. Mathematically, the Calinski-Harabasz (CH) score can be expressed as:

$$CH(k) = \frac{tr(B_k)}{tr(W_k)} \times \frac{n-k}{k-1} \quad (2)$$

where k is the number of clusters, n is the total of data points, $tr(B_k)$ is the trace of the between-cluster dispersion matrix, and $tr(W_k)$ is the trace of the within-cluster dispersion matrix.

The between-cluster dispersion matrix, B_k, as shown by the third equation below, measures the distance between cluster centroids, while the within-cluster dispersion matrix, W_k as demonstrated by the fourth equation below, measures the distance within clusters, n_q is the total number of data points in cluster q, c_q is the set of all data in cluster q, c_E is the center point of all data, and T is the transpose operation for the matrix [28].

$$B_k = \sum_{q=1}^{k} n_q \, (c_q - c_E)(c_q - c_E)^T \quad (3)$$

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_E)(x - c_E)^T \quad (4)$$

III. Result and Discussion

The data collection process began by outlining the dataset's origins and key features, laying a solid foundation for analysis. Through exploratory data analysis (EDA), we discovered patterns, trends, and relationships in the data, offering insights to guide our next steps. The data then underwent careful preparation, including cleaning, handling any missing or null values, and readying it for use in machine learning. We chose four machine learning algorithms to build the recommendation system based on their strengths. K-means and mini-batch k-means are ideal for large datasets, clustering data into similar groups quickly and efficiently. Gaussian mixture models add flexibility, capturing complex,

overlapping clusters. DBSCAN, on the other hand, excels in finding clusters of various shapes and densities and can handle noise well. After testing each algorithm, we evaluated their performance on the dataset to determine the best fit.
With these results, the recommendation system can generate music suggestions tailored to users' unique tastes, completing the journey from data collection to personalized music recommendations.

A. Data Collection

Fig. 6 below shows data collection through the web application programming interface (API), which is Spotify API, encompassing a broad spectrum of music attributes and associated metadata. This process involves extracting multifaceted information pertinent to songs, albums, and artists, capturing details such as track duration, tempo, key, mode, and acoustics. Notably, the dataset comprises the top songs of 12 years spanning from 2013 to 2024, which have been combined to create this dataset. The dataset has 20 features. Duplicated songs were dropped, and the dataset was left with 1003 songs. This structure enables detailed analysis of various musical attributes and metadata, supporting insights into music trends and characteristics within the dataset.

	name	album	artist	release_date	length	popularity	danceability	explicit	acousticness	energy	instrumentalness
0	Mirrors	The 20/20 Experience (Deluxe Version)	Justin Timberlake	2013	484146	82	0.574	False	0.23400	0.512	0.000000
1	Blurred Lines	Blurred Lines	Robin Thicke	2013	283053	56	0.861	False	0.00412	0.504	0.000018
2	Royals	Pure Heroine	Lorde	2013	190185	81	0.674	False	0.12100	0.428	0.000000
3	Another Love	Long Way Down (Deluxe)	Tom Odell	2013	244360	90	0.445	True	0.69500	0.537	0.000017
4	Locked out of Heaven	Unorthodox Jukebox	Bruno Mars	2012	233478	90	0.726	False	0.04900	0.698	0.000000

Fig. 6 Dataset of music collected each year from 2013 to 2024

In the dataset, 20 features as shown in Fig. 7, have been extracted and displayed for analysis. These include information such as the name of the song ('name'), the album it belongs to ('album'), and the artist who created it ('artist'). Additionally, details like the release date ('release_date') and the length of the song in seconds ('length') provide temporal context. Other attributes like 'popularity', 'danceability', 'explicit', 'acousticness', 'energy', 'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo', 'time_signature', 'valence', 'key', 'mode', and 'genres' offer insights into various musical characteristics and metadata. Each of these columns serves as vital information for further exploration and analysis, enabling this research to delve deeper into the dynamics of music and its attributes.

```
Index(['name', 'album', 'artist', 'release_date', 'length', 'popularity',
       'danceability', 'explicit', 'acousticness', 'energy',
       'instrumentalness', 'liveness', 'loudness', 'speechiness', 'tempo',
       'time_signature', 'valence', 'key', 'mode', 'genres'],
      dtype='object')
```

Fig. 7 Column names for the dataset

The dimensionality of the dataset in Fig. 8 indicates that it contains 1003 songs and 20 features. In other words, there are 1003 songs in the dataset, each with 20 different attributes or features. This information provides an overview of the size and structure of the dataset, which is essential for understanding the scope of the analysis and the amount of data available for exploration.

```
Dimensionality of the dataframe: (1003, 20)
```

Fig. 8 Dimensionality of dataset

B. Exploratory Data Analysis

Fig. 9 provides information for the song years, sorted by the number of released songs. As illustrated in Fig. 10 below, the bar chart indicates variations in the number of songs released across different years, spanning from 1985 to 2024. In the exact figure, Fig. 10 provides information about top songs on the number of released songs sorted by year. In 2016, the highest number of songs were released, with a count of 112, followed closely by 2017, with a count of 108, and in 2020, with a count of 101 songs. The year 2021 saw a slightly lower count of 82 released songs. On the other hand, years like 2019 had a lower count of 29 songs. As the data collection progresses into more recent years, there is a noticeable decrease in the number of songs released, with only eight songs recorded up to March 2024. There are sporadic releases in earlier years, such as 1985, 2002, 2004, and 2011, each with only one song. In summary, the visualization sheds light on the distribution of song releases from 1985 to 2024, reflecting the trends and preferences within the music industry's top selections based on listeners.

```
Sorted number of released songs for the years:
 Year  Counts
2016     112
2017     108
2020     101
2014      97
2015      91
2022      88
2023      88
2018      84
2021      82
2013      75
2012      36
2019      29
2024       8
1985       1
2011       1
2002       1
2004       1
```

Fig. 9 Sorted number of released songs for the years

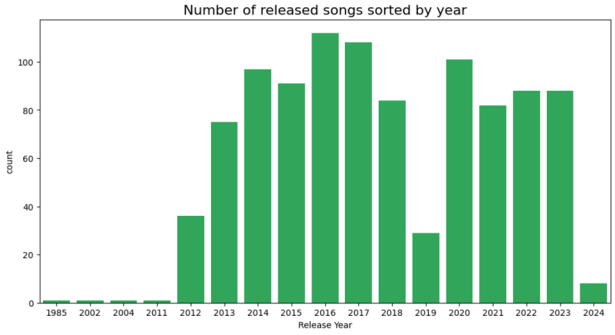

Fig. 10 Bar chart for the number of released songs sorted by year

The word cloud visualization in Fig. 11 offers a compelling snapshot of the most mentioned genres within the dataset, showcasing the prominence of pop, hip-hop, and dance-pop. These genres emerge as dominant themes, characterized by more extensive and bolder representations within the word cloud, indicating their frequent occurrence and prevalence in the dataset. This visualization highlights the popularity of these genres and provides insight into the music industry's trends and preferences over the years.

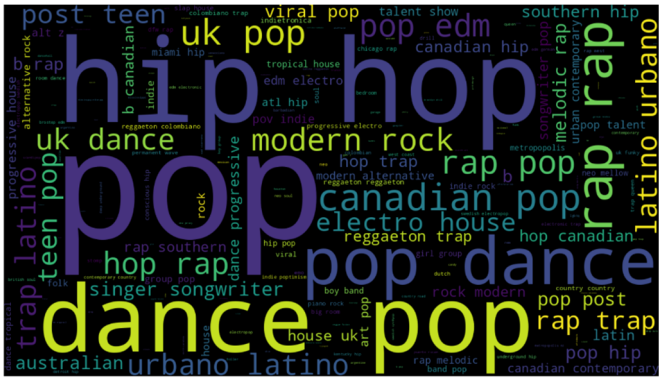

Fig. 11 Word cloud visualization on most mentioned genres

Based on the observations made, the Fig. 12 visualization illustrates that songs released in 2002 that made it into the top songs exhibit the highest popularity scores. This suggests a notable resonance with audiences and widespread acclaim for the music released in 2002. Additionally, songs from 2004 and 2024 also demonstrate considerable popularity, though to a slightly lesser extent compared to those from 2002.

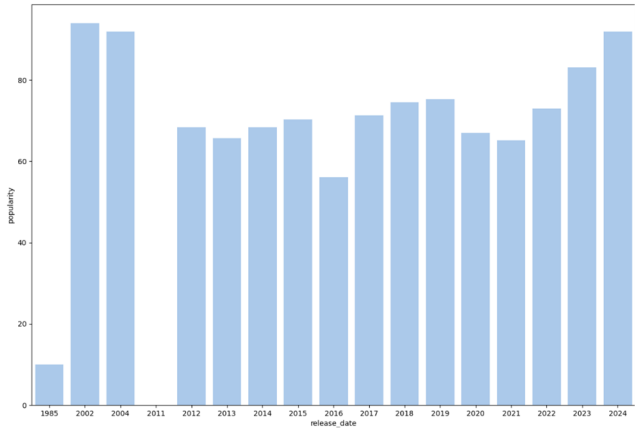

Fig. 12 Popularity of songs based on the year it is released

The accompanying visualizations in Fig. 13 visually represent this dataset distribution, making it easier to compare the counts of explicit song lyrics and minor-major song modes. The bar plot for the 'explicit' attribute showcases the frequency of explicit and non-explicit song lyrics, while the bar plot for the 'mode' attribute illustrates the occurrence of music key modes, which are minor and have zero value, and major, which has the one value in the dataset.

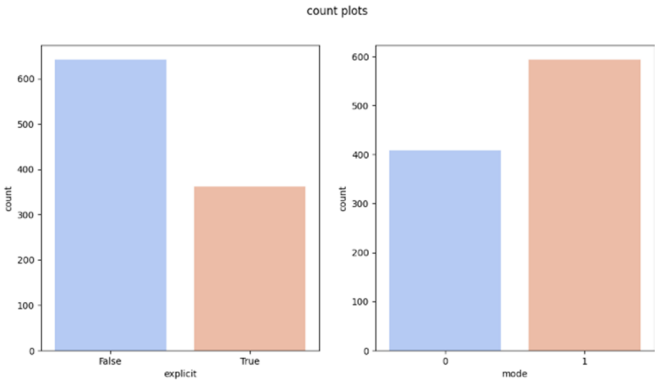

Fig. 13 Count plot for explicit and mode column

The collected data presents the counts of explicit and mode values. For the 'explicit' attribute in Fig. 14, there are 642 occurrences where the value is False, indicating that most songs in the dataset do not contain explicit content. Conversely, 361 occurrences where the value is True, suggesting that a smaller portion of the songs feature explicit content.

```
Explicit value counts:
explicit
False    642
True     361
```

Fig. 14 Value counts for explicit song lyrics column

The value counts for the 'mode' attribute in Fig. 15 reveals that mode 1 appears 594 times, while mode 0 appears 409 times. The 'Mode' attribute typically represents the modality of the song, with Mode 1 corresponding to primary keys and Mode 0 to minor keys in music theory. This distribution indicates that songs with a major essential mode (mode 1) are more prevalent in the dataset than those with a minor key mode (mode 0).

```
Mode value counts:
mode
1    594
0    409
```

Fig. 15 Value counts for mode column

C. Data Preprocessing

The provided dataset information in Fig. 16 reveals a comprehensive dataset overview comprising 1003 entries across 20 columns. Through an examination of the data, it's evident that no missing values are present in any of the columns, ensuring the completeness and integrity of the dataset. Each column contains non-null entries, indicating that all necessary information is available for analysis. The absence of null values simplifies the analysis process, eliminating the need for data imputation or handling missing values and allowing for a more accurate and efficient dataset exploration.

#	Column	Non-Null Count	Dtype
0	name	1003 non-null	object
1	album	1003 non-null	object
2	artist	1003 non-null	object
3	release_date	1003 non-null	int64
4	length	1003 non-null	int64
5	popularity	1003 non-null	int64
6	danceability	1003 non-null	float64
7	explicit	1003 non-null	bool
8	acousticness	1003 non-null	float64
9	energy	1003 non-null	float64
10	instrumentalness	1003 non-null	float64
11	liveness	1003 non-null	float64
12	loudness	1003 non-null	float64
13	speechiness	1003 non-null	float64
14	tempo	1003 non-null	float64
15	time_signature	1003 non-null	int64
16	valence	1003 non-null	float64
17	key	1003 non-null	int64
18	mode	1003 non-null	int64
19	genres	1003 non-null	object

Fig. 16 Overview of non -null value count per column

Upon inspection, no duplicate samples have been identified within the dataset as shown in Fig. 17. This suggests that each entry within the dataset is unique, devoid of any repetitions or redundancies. The absence of duplicate samples ensures the integrity and accuracy of the dataset, as each observation represents distinct information without any unnecessary replication.

```
No duplicate samples found
```

Fig. 17 Findings of any duplicate value in dataset

D. K-means Algorithm

In conducting the elbow method to determine the optimal number of clusters, k, for the dataset, it was observed that the inertia, which represents the sum of squared distances of samples to their closest cluster center, decreased as the number of clusters increased. However, as seen in Fig. 18 there was a noticeable inflection point at 3-rd cluster where the rate of decrease in inertia significantly slowed down. This inflection point is often referred to as the "elbow" of the plot. This finding suggests that 3-rd cluster is the most suitable choice for partitioning the data, as it strikes a balance between capturing meaningful patterns within the data while minimizing redundancy. By selecting 3-rd cluster or *k* is 3, a more effective solution for the given dataset can be achieved.

By utilizing the k-means algorithm with number of clusters equals to 3 and fitting it to the preprocessed data, this model successfully partitioned the data into three distinct clusters based on the similarity of data points. Each cluster is represented by a centroid, and data points are assigned to the cluster with the nearest centroid. This approach allows for the identification of underlying patterns and structures within the data, enabling the grouping of similar data points together. By setting the random seed for reproducibility, the results obtained from the clustering process can be replicated consistently.

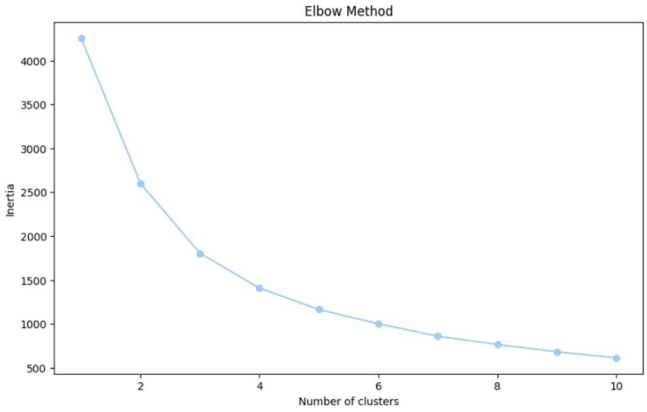

Fig. 18 Elbow method for K-means

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

Fig. 19 K-means parameters

E. Density-based Spatial Clustering of Applications with Noise Algorithm

Through a systematic evaluation of different parameter combinations for the density-based spatial clustering of applications with noise (DBSCAN) algorithm focusing on two parameters which are epsilon (eps) and minimum samples (min_samples), the analysis aimed to identify the optimal configuration that maximizes clustering quality. By utilizing the Calinski-Harabasz score as a metric to assess clustering performance, the study sought to pinpoint the parameter values that lead to well-defined and distinct clusters within the dataset. The iterative process involved calculating the Calinski-Harabasz score for varying eps and min_samples parameters, with the ultimate objective of determining the parameters that yield the highest Calinski-Harabasz score, indicative of superior clustering outcomes.

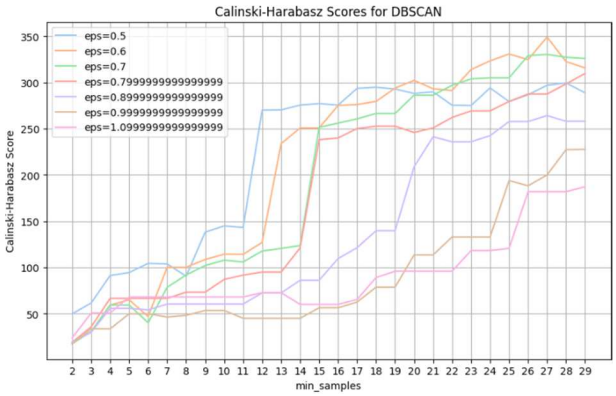

Fig. 20 Calinski-Harabasz score for DBSCAN parameter tuning

Following this comprehensive analysis in Fig. 20, it was determined that an eps value of 0.6 paired with a minimum number of samples value of 27 emerged as the most suitable combination of parameters for the DBSCAN algorithm, as it produced the highest Calinski-Harabasz score. These optimized parameters selection ensures that the DBSCAN algorithm effectively captures the underlying structures and

patterns present in the dataset, leading to more accurate and meaningful clustering results. The lines graph visualization of the Calinski-Harabasz scores across parameter variations for DBSCAN below provides valuable insights into the impact of hyperparameters tuning on clustering performance.

By initializing the DBSCAN algorithm with an eps value of 0.6 and a minimum number of samples value of 27 as seen in Fig. 21. The chosen parameter values of eps and min_samples determine the distance threshold for defining neighborhood points and the minimum number of points required to form a dense region.

```
DBSCAN
DBSCAN(eps=0.6, min_samples=27)
```

Fig. 21 DBSCAN parameters

F. Mini Batch K-means Algorithm

In the findings of the mini-batch k-means algorithm, it was determined through the elbow method that the optimal number of clusters is 4 which can be seen in Fig. 22. This suggests that the data points can be effectively grouped into four distinct clusters based on their similarities. By identifying this optimal cluster number, the algorithm can efficiently partition the data into meaningful groups, aiding in data analysis and pattern recognition.

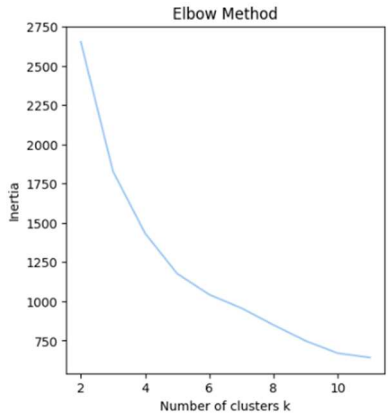

Fig. 22 Elbow method for mini batch k-means

Mini batch k-means algorithm is initialized with a specified number of clusters to 4 and a random state for reproducibility as seen in Fig. 23. This process involves iteratively updating the cluster centroids based on mini-batches of the input data, making it computationally efficient for large dataset. By setting the number of clusters to 4, the algorithm will aim to partition the data into four distinct groups based on their similarities.

```
MiniBatchKMeans
MiniBatchKMeans(n_clusters=4, random_state=42)
```

Fig. 23 Mini batch k-means parameters

G. Gaussian Mixture Algorithm

In Fig. 24, the Gaussian mixture model is initialized with a specified number of components set to 20 to account for the number of features in the dataset, and a random state is defined for reproducibility. By utilizing a Gaussian mixture model, the algorithm aims to model the underlying distribution of the data using a combination of Gaussian distributions, allowing for more flexible and complex cluster shapes compared to traditional k-means clustering. The training process involves estimating mixtures of parameters using the Gaussian mixture algorithm based on the input data.

```
GaussianMixture
GaussianMixture(n_components=20, random_state=42)
```

Fig. 24 Gaussian mixture parameters

H. Performance Scores for Four Machine Learning Algorithms

The Calinski-Harabasz score serves as a crucial metric for assessing the efficacy of algorithms in identifying meaningful patterns within data. A higher score indicates better-defined and more separate clusters, signifying the algorithm's ability to capture underlying structures effectively.

K-means achieved a Calinski score of 680.081410 which suggests it performed the best among the four machine learning algorithms. Similarly, the mini batch k-means algorithm achieved a score of 668.276546, showcasing that the model ranked second among these algorithms. Comparatively, the Gaussian mixture algorithm yielded a Calinski score of 595.570594, indicating that the model ranked third among these algorithms. However, DBSCAN scored the lowest at 327.996767, suggesting challenges, likely due to its sensitivity to density variations and noise.

Algorithm	score
K-means	680.081410
DBSCAN	327.996767
Mini Batch K-means	668.276546
Gausian Mixture	595.570594

Fig. 25 Calinski-Harabasz score for four algorithms

The bar plot visualizing the Calinski-Harabasz scores for the four machine learning algorithms. It provides a clear comparison of their performances. The plot illustrates the relative strengths of each algorithm, with the length of each bar representing the corresponding Calinski-Harabasz score. This visualization allows for an immediate and intuitive understanding of the machine learning performances, aiding in the identification of the most effective algorithm.

The K-means excelled due to its ability to create compact and well-separated clusters, aligning well with the Calinski-Harabasz score criteria, which rewards such cluster structures. The features of the music dataset, including audio characteristics, formed spherical clusters that K-means effectively exploited. The simplicity and efficiency of K-means allowed for quick convergence to stable clusters, offering a clear structure evaluated highly by the Calinski-Harabasz score. Additionally, K-means required fewer parameters to tune, ensuring more consistent performance

across different settings. This combination of factors led to the best performance of K-means among three other algorithms.

Fig. 26 Bar plot visualization for Calinski-Harabasz scores for all four machine learning algorithms

I. Music Recommendation using K-means Algorithm

The decision to utilize the k-means algorithm in the music recommendation was made based on the evaluation of various machine learning algorithms using the Calinski-Harabasz score. Among the four machine learning algorithms considered, k-means emerged as the top performer, boasting the highest Calinski-Harabasz score.

Song: [{'name': 'Easy On Me', 'release_date': 2021}]

	release_date	name	artist
443	2017	Hold On	Chord Overstreet
624	2021	traitor	Olivia Rodrigo
613	2022	Glimpse of Us	Joji
366	2016	This Town	Niall Horan
937	2019	Before You Go	Lewis Capaldi
695	2021	Until I Found You	Stephen Sanchez
456	2017	Happier	Ed Sheeran
753	2019	All Of The Girls You Loved Before	Taylor Swift
420	2017	Too Good At Goodbyes	Sam Smith
670	2022	The Joker And The Queen (feat. Taylor Swift)	Ed Sheeran

Fig. 27 Music recommendation results for the user who listens to the song named "Easy On Me"

By opting for k-means for the music recommendation, it is capitalizing on its capacity to discern meaningful patterns and similarities in the dataset, thereby enhancing the precision and relevance of the recommendation results generated.

The music recommendation using k-means is designed to accept unique inputs based on the song name and its released year, and subsequently provide music recommendation results of similar songs based on features within the dataset. By leveraging the best machine learning algorithm, which is k-means, the k-means algorithm identifies songs with comparable attributes, including musical characteristics and genre. K-means algorithm ensures that music recommendation results are relevant. By surfacing songs with similar attributes based on what users listen to, the algorithm fosters engagement and satisfaction, ultimately enhancing the overall listening experience. Fig. 27 and Fig. 28 below are the two different examples of music recommendation results based on what users listening to.

Song: [{'name': 'a thousand years', 'release_date': 2019}]

	release_date	name	artist
92	2013	human	Christina Perri
416	2017	Perfect	Ed Sheeran
276	2015	Let It All Go	Birdy
657	2021	Easy On Me	Adele
128	2013	Let It Go - From "Frozen"/Soundtrack Version	Various Artists
366	2016	This Town	Niall Horan
202	2015	Love Me Like You Do - From "Fifty Shades Of Grey"	Ellie Goulding
718	2022	Here With Me	d4vd
557	2017	Rewrite The Stars	Zac Efron
13	2013	Wrecking Ball	Miley Cyrus

Fig. 28 Music recommendation results for the user who listens to the song named "a thousand years".

IV. Conclusion

In summary, analyzing the performance of different machine learning algorithms—k-means, mini batch k-means, Gaussian mixture, and DBSCAN—gave us valuable insights into their strengths. The evaluation using the Calinski-Harabasz score revealed k-means as the top performer, with its high score indicating an effective grouping of similar data points. Based on this, k-means was chosen for our music recommendation system because of its strong ability to uncover patterns and similarities that enhance the relevance of song suggestions to user preferences. On the other hand, DBSCAN faced limitations, particularly with varying cluster densities and high-dimensional data, where density-based clustering is less effective.. Moving forward, further exploration of applying other algorithm type such as integrating deep learning models [29], [30], [31] for the music recommendation and expending literature review scope to cover more algorithms.

References

[1] P. Chinnasamy, W.-K. Wong, A. A. Raja, O. I. Khalaf, A. Kiran, and J. C. Babu, "Health Recommendation System using Deep Learning-based Collaborative Filtering," Heliyon, vol. 9, no. 12, p. e22844, Dec. 2023, doi: 10.1016/j.heliyon.2023.e22844.

[2] S. Forouzandeh, M. Rostami, K. Berahmand, and R. Sheikhpour, "Health-aware food recommendation system with dual attention in heterogeneous graphs," Computers in Biology and Medicine, vol. 169, p. 107882, Feb. 2024, doi: 10.1016/j.compbiomed.2023.107882.

[3] R. F. T. Ceskoutsé, A. B. Bomgni, D. R. Gnimpieba Zanfack, D. D. M. Agany, T. Bouetou Bouetou, and E. Gnimpieba Zohim, "Sub-clustering based recommendation system for stroke patient: Identification of a specific drug class for a given patient," Computers in Biology and Medicine, vol. 171, p. 108117, Mar. 2024, doi:10.1016/j.compbiomed.2024.108117.

[4] S. Siji Rani, P. Shilpa, and A. G. Menon, "Enhancing Drug Recommendations: A Modified LSTM Approach in Intelligent Deep Learning Systems," Procedia Computer Science, vol. 233, pp. 872–881, 2024, doi: 10.1016/j.procs.2024.03.276.

[5] I. Ahmed, M. Ahmad, A. Chehri, and G. Jeon, "A heterogeneous network embedded medicine recommendation system based on LSTM," Future Generation Computer Systems, vol. 149, pp. 1–11, Dec. 2023, doi: 10.1016/j.future.2023.07.004.

[6] G. Liu, "An ecommerce recommendation algorithm based on link prediction," Alexandria Engineering Journal, vol. 61, no. 1, pp. 905–910, Jan. 2022, doi: 10.1016/j.aej.2021.04.081.

[7] M. Mao, S. Chen, F. Zhang, J. Han, and Q. Xiao, "Hybrid ecommerce recommendation model incorporating product taxonomy and

folksonomy," Knowledge-Based Systems, vol. 214, p. 106720, Feb. 2021, doi: 10.1016/j.knosys.2020.106720.
[8] R. V. Karthik and S. Ganapathy, "A fuzzy recommendation system for predicting the customers interests using sentiment analysis and ontology in e-commerce," Applied Soft Computing, vol. 108, p. 107396, Sep. 2021, doi: 10.1016/j.asoc.2021.107396.
[9] B. Walek and P. Fajmon, "A hybrid recommender system for an online store using a fuzzy expert system," Expert Systems with Applications, vol. 212, p. 118565, Feb. 2023, doi: 10.1016/j.eswa.2022.118565.
[10] Y. Yuan and W. Zheng, "Your trip, your way: An adaptive tourism recommendation system," Applied Soft Computing, vol. 154, p. 111330, Mar. 2024, doi: 10.1016/j.asoc.2024.111330.
[11] W. Zheng, Z. Liao, and Z. Lin, "Navigating through the complex transport system: A heuristic approach for city tourism recommendation," Tourism Management, vol. 81, p. 104162, Dec. 2020, doi: 10.1016/j.tourman.2020.104162.
[12] Z. Abbasi-Moud, H. Vahdat-Nejad, and J. Sadri, "Tourism recommendation system based on semantic clustering and sentiment analysis," Expert Systems with Applications, vol. 167, p. 114324, Apr. 2021, doi: 10.1016/j.eswa.2020.114324.
[13] Z. Abbasi-Moud, S. Hosseinabadi, M. Kelarestaghi, and F. Eshghi, "CAFOB: Context-aware fuzzy-ontology-based tourism recommendation system," Expert Systems with Applications, vol. 199, p. 116877, Aug. 2022, doi: 10.1016/j.eswa.2022.116877.
[14] G. Behera and N. Nain, "Collaborative Filtering with Temporal Features for Movie Recommendation System," Procedia Computer Science, vol. 218, pp. 1366–1373, 2023, doi:10.1016/j.procs.2023.01.115.
[15] N. G. K, K. T. V. Durga, N. Hrishita, R. R, and M. Panda, "A Cross-Platform Movie Filtering and Recommendation System Using Big Data Analytics," Procedia Computer Science, vol. 235, pp. 81–90, 2024, doi: 10.1016/j.procs.2024.04.008.
[16] S. Airen and J. Agrawal, "Movie Recommender System Using Parameter Tuning of User and Movie Neighbourhood via Co-Clustering," Procedia Computer Science, vol. 218, pp. 1176–1183, 2023, doi: 10.1016/j.procs.2023.01.096.
[17] N. Pavitha et al., "Movie recommendation and sentiment analysis using machine learning," Global Transitions Proceedings, vol. 3, no. 1, pp. 279–284, Jun. 2022, doi: 10.1016/j.gltp.2022.03.012.
[18] H. Xian and S. Wu, "Design of an English vocabulary e-learning recommendation system based on word bag model and recurrent neural network algorithm," Entertainment Computing, vol. 50, p. 100692, May 2024, doi: 10.1016/j.entcom.2024.100692.
[19] W. Xi, "Research on E-learning interactive English vocabulary recommendation education system based on naive Bayes algorithm," Entertainment Computing, vol. 51, p. 100732, Sep. 2024, doi:10.1016/j.entcom.2024.100732.
[20] Y. Zhai, "Design of Oral English Training System Based on Big Data Content Recommendation Algorithm," Procedia Computer Science, vol. 208, pp. 420–426, 2022, doi: 10.1016/j.procs.2022.10.059.
[21] K. R, P. Kumar, and B. Bhasker, "DNNRec: A novel deep learning based hybrid recommender system," Expert Systems with Applications, vol. 144, p. 113054, Apr. 2020, doi:10.1016/j.eswa.2019.113054.
[22] Y. Tian, B. Zheng, Y. Wang, Y. Zhang, and Q. Wu, "College Library Personalized Recommendation System Based on Hybrid Recommendation Algorithm," Procedia CIRP, vol. 83, pp. 490–494, 2019, doi: 10.1016/j.procir.2019.04.126.
[23] Y. Afoudi, M. Lazaar, and M. Al Achhab, "Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network," Simulation Modelling Practice and Theory, vol. 113, p. 102375, Dec. 2021, doi:10.1016/j.simpat.2021.102375.
[24] A. Fareed, S. Hassan, S. B. Belhaouari, and Z. Halim, "A collaborative filtering recommendation framework utilizing social networks," Machine Learning with Applications, vol. 14, p. 100495, Dec. 2023, doi: 10.1016/j.mlwa.2023.100495.
[25] Z. Movafegh and A. Rezapour, "Improving collaborative recommender system using hybrid clustering and optimized singular value decomposition," Engineering Applications of Artificial Intelligence, vol. 126, p. 107109, Nov. 2023, doi:10.1016/j.engappai.2023.107109.
[26] A. S. Akshay, M. Madhura, B. Mounika, & H.S. Rohit, B. M. Vikranth, "Music Recommendation System," 2019.
[27] Ahuja R, Solanki A, and Nayyar A, Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor.
[28] H. Yu, C. Zhou, J. Bao, H. Yao, P. Yan, and Q. Wang, "Analysis and Effect Evaluation of Offshore Wind Power Output Characteristics Based on Gaussian Mixed Clustering," Procedia Computer Science, vol. 224, pp. 389–394, 2023, doi: 10.1016/j.procs.2023.09.053.
[29] F. O. Isinkaye, M. O. Olusanya, and P. K. Singh, "Deep learning and content-based filtering techniques for improving plant disease identification and treatment recommendations: A comprehensive review," Heliyon, vol. 10, no. 9, p. e29583, May 2024, doi:10.1016/j.heliyon.2024.e29583.
[30] N. Li and Y. Xia, "Movie recommendation based on ALS collaborative filtering recommendation algorithm with deep learning model," Entertainment Computing, vol. 51, p. 100715, Sep. 2024, doi:10.1016/j.entcom.2024.100715.
[31] J. Guo, J. He, and X. Wu, "Shopping trip recommendations: A novel deep learning-enhanced global planning approach," Decision Support Systems, vol. 182, p. 114238, Jul. 2024, doi:10.1016/j.dss.2024.114238.