Computer Science and Information Systems 23(1):585–601 https://doi.org/10.2298/CSIS250418083N

Audio Feature-based User Profiles for Personalized Music Recommendation: A Dataset-driven Evaluation*

Ionuţ-Dragoş Neremzoiu and Andreea Liliana Bădică

University of Craiova, Craiova, Romania
neremzoiu.ionut.k4c@student.ucv.ro
andreea.badica121@yahoo.com

Abstract. In this paper, we propose an experimental content-based track recommendation system, which relies on an aggregated features which considers audio feature values from Spotify Data Catalog, track lyrics and popularity ratings. As system evaluation protocol, we evaluate how relevant the top-5 recommended tracks are to the user profile, which is based on average audio feature (danceability, energy, valence, loudness, instrumentalness, liveness, speechiness and acousticness) values from the user's listening history. Based on the evaluation results, we come to the conclusion that the recommended tracks are similar to the user's profile.

Keywords: recommender systems, pca, k-means, aggregated feature, mean absolute error

1. Introduction

Ever since the rise of music streaming services such as Spotify or Youtube Music, recommender systems have played a huge role in promoting new songs. These systems rely on various types of data, such as geographic user data, listening history, list of liked songs, etc. Recommendation strategies are generally partitioned into three major approaches:

- *Collaborative filtering* that recommends items to users based on how other users with similar preferences and behavior have interacted with that item [14].
- *Content-based* that uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback [10] .
- *Hybrid* that combines two or more recommendation strategies in different ways to benefit from their complementary advantages [20].

One noteworthy content-based music recommendation system that focuses on the acoustic similarity of musical compositions rather than relying on metadata such as genre, artist or user ratings, is presented in [15]. This paper showcases how acoustic analysis and spectrogram-based deep learning can enhance music recommendations without relying on user data.

A noteworthy recommendation system for music tracks using collaborative filtering and two optimization methods: *dataset rescaling* and *automatic halting* is presented in

* This paper is an extended version of our previous work published in INISTA 2024 [13]. The extensions include a reduced but more useful version of a dataset that was previously used, a new evaluation protocol, new results derived from the evaluation and new discussions in terms of conclusions, which significantly enhance the depth and scope of the original study.

[9]. These methods employed various parameters to address the performance issue while retaining an acceptable accuracy. Those parameters are the following:

- A user similarity threshold (θ),
- A fraction (τ) of the full dataset randomly sampled for processing used for generating song recommendations,
- A hash-table representation to store sparse user-song interactions efficiently.

As for the hybrid approach to music recommendation, [11] combines content-based filtering with the K-means clustering algorithm. The hybrid approach aimed to improve the relevance of music recommendations and to address the cold-start problem, which occurs when a system lacks enough data about a new user or new content.

Our work proposes a new *content-based* recommendation approach based on a specially customized dataset that we created by aggregating audio features extracted from Spotify Catalog, i.e. danceability, energy, valence, loudness, instrumentalness, liveness, speechiness, acousticness, duration, tempo, as well as lyrics and popularity ratings of songs. We proposed the aggregation of these features into a new dataset in order to enhance user experience by recommending songs that better match user preferences in terms of audio features, lyrics and popularity ratings. Moreover, before recommending the tracks, we also classify the tracks from our custom-made dataset in numbered labels using K-Means algorithm as [11]. The way we choose the number of labels is highlighted in Section 3.

This paper is a significant extension and improvement of our preliminary work [13] by strengthening our experimental evaluation using a more appropriate set of test users. For every test user, we assume a user profile. This profile is based on the average values of key audio features — danceability, energy, valence, loudness, instrumentalness, liveness, speechiness, and acousticness — calculated from all tracks[1] the user played at least three times. Using this profile, we compare the averaged feature values from the user's listening history with the corresponding feature values of the top 5 recommended tracks.

The paper is structured as follows:

- Section 1 introduces the context, motivation and related works, and summarizes the content of the paper.
- Section 2 introduces the dataset description and preparation.
- Section 3 approaches the experimental system design.
- Section 4 addresses both the evaluation protocol and results.
- The last section presents our conclusions and points future work.

2. Dataset Description and Preparation

We created a novel and more suitable dataset to configure our recommendation system by integrating the dataset from the work of Ronak Doshi et al. [17] (Github dataset, 10,449 track entries) and the dataset created by Xiaomeng Zhang [19] (Jovian dataset, 174,389 track entries). We extracted only the Spotify Data Catalog features found in both datasets and we applied a preprocessing pipeline consisting of *data filtering*, *data cleaning* and

[1] The terms *track* and *song* are used interchangeably.

Table 1. Track dataset. Except lyrics and label, the field descriptions are detailed from [7, 2]

Field Name	**Field Description**
Name	Track name
Artist	The name of an artist or multiple artists separated by commas.
Duration	The duration of a track in milliseconds.
Popularity	Rating value between 0 (least popular song) and 100 (most popular song).
Key	The estimated overall key of the track. Integers map to pitches using standard "Pitch Class notation" (E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on).
Acousticness	A confidence measure from 0.0 to 1.0 of whether the track is acoustic.
Danceability	Track suitability (on a scale from 0 to 1) for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength and overall regularity.
Energy	A measure from 0.0 to 1.0 representing intensity and activity of a track.
Instrumentalness	Measure of lack of vocals (from 0 to 1). "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal".
Liveness	Detects the presence of an audience in the recording. Higher liveness values (especially above 0.8) represent an increased probability that the track was performed live.
Loudness	The overall loudness of a track in decibels (dB).
Speechiness	Measure of the presence of exclusively spoken words (speech-like). Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
Valence	Describes the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
Tempo	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece, and derives directly from the average beat duration.
Mode	Indicates the modality (major or minor) of a track.
Lyrics	Track lyrics collected through Basile Bruneau's [4] JSON API and web-scraping (*Genius*, *MusixMatch* and *AZLyrics*)
Label	The cluster label a track is assigned to.

joining, resulting a final dataset of 180,220 unique tracks. The description of the resulting dataset is provided in Table 1.

Data filtering aimed to remove duplicates. We found that only the Jovian dataset had 2159 track duplicates according to the *track id* feature. for identifying duplicates, which were ultimately removed. Thus, from this dataset we kept the remaining 172,230 tracks.

Data cleaning aimed to harmonize the format and syntax of the external representation of data. In the Jovian dataset, the artist names involved in singing a track are written between ' or " into a comma-separated list with brackets. The artist names from the Github dataset don't have any ' or " and are not included in a list with brackets. To make a simple and consistent representation, we chose the data format from the Github dataset for the artist names. For instance, the initial value *['Mamie Smith', 'The Harlem Trio']* will become *Mamie Smith, The Harlem Trio* after cleaning.

When joining the datasets by taking into account track id, we firstly checked whether the track id occurs in both datasets. Then, for each track id shared in both datasets before joining, the means of all non-categorical data fields (*acousticness*, *danceability*, etc.) were computed and saved into the final dataset. We managed to gather the lyrics of 131,651 songs out of the 180,220 of the entire initial dataset. The lyrics data was further preprocessed to compute Term Frequency – Inverse Document Frequency (TF-IDF) coefficients (see next section).

3. Experimental System Design

3.1. System overview

This section describes in further details each process from Figure 1 that contributes to the functionality of the system. For simplicity purposes, we categorize all the processes involved in three main phases: the *data loading*, the *preprocessing phase*, and the *recommendation algorithm*.

The *data loading* phase simply represents the process in which we load our custom-made dataset introduced in Section 2.

The preprocessing phase aims to cluster the songs based on their aggregated audio features (principal components) and to evaluate the importance of each word in the lyrics of each song relative to the collection of songs with lyrics (a corpus). The steps of the preprocessing phase are the following: *Feature Selection*, *Min-max Scaling*, *Dimensionality reduction*, *Clustering*, *Train-test split*, *Track label classification*, and *Lyrics TF-IDF matrix computation*.

The proposed recommendation algorithm was extended based on the proposal from [17] by aggregating the similarity of audio features as detailed in 3.3, the popularity rating and the similarity of the lyrics. The initial algorithm from [17] uses two major sources for computing the similarity measure between songs:

- the principal component analysis-based Euclidean distance between the song used for generating recommendations and a song belonging to the same label as the former song.
- the Spotify track popularity rating of a song belonging to the same label as the song used for generating recommendations.

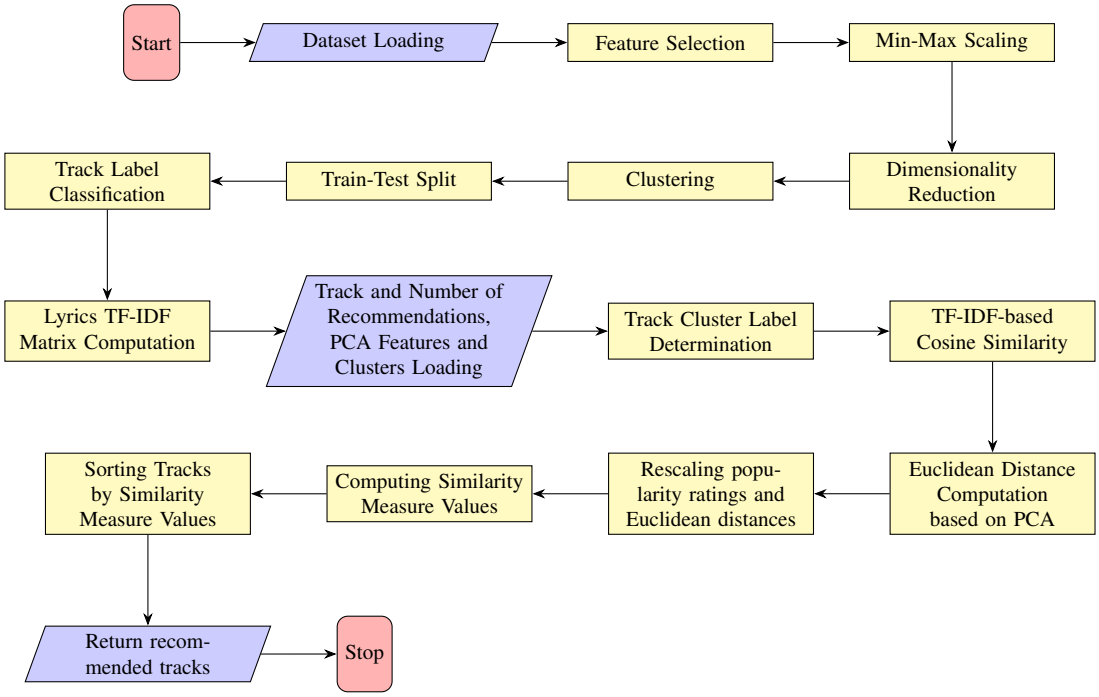

Fig. 1. *Workflow of the recommendation system.* The preprocessing phase involves all processes from *Data Loading* to *Lyrics TF-IDF Matrix Computation*, whereas the recommendation algorithm the next stage, which involves reading its necessary inputs until returning the recommendations

The similarity measure proposed by [17] is computed using Equation (1):

$$S(t, t') = \frac{P(t')}{(D(t, t'))^2} \tag{1}$$

where:

- $S(t, t')$ is the similarity measure
- $P(t')$ is the rating value of a current track t' belonging to the same label as the input track t
- $D(t, t')$ is Euclidean distance between tracks t and t'

[17] uses Principal Component Analysis (PCA) on the audio Spofity features for dimensionality reduction and *K-Means* for clustering the tracks in 4 clusters, which we also consider for the preprocessing in our dataset. However, unlike [17], we investigated in [13] a wider range of values for two parameters: i) the number of principal components that satisfy a threshold ranged between 90-95% cumulative variance and ii) the number of clusters for *K-Means Clustering*. Moreover, we adopted the parameter values obtained by *Elbow method*, *Silhouete method* and *Calinski-Harabasz Index* as evaluation measures for the optimal number of clusters.

We updated the computation of the similarity measure from (1) by using, apart from PCA features and popularity rating, the similarity measure for lyrics detailed further in Section 2. The motivation is that we observed that the initial proposal from (1) weights the popularity rating too much to the point where even if a track used for recommendations is similar to another track from the same group (based on PCA features and audio characteristics), if that track has a low popularity rating, the overall similarity measure value will be too low, if not the lowest, thus hindering its selection for recommendation.

The steps of the *recommendation algorithm* involved the determination of: *Track cluster label*, *TF-IDF-based cosine similarity*, *Euclidean distances based on the principal components*, *Rescaling popularity ratings and Euclidean distances*, and, respectively, *Track Ranking based on Similarity Measure Values*. Apart from that, it is important to highlight the fact that our recommendation algorithm has a track and the number of recommendations as set of inputs and the number of PCA features and clusters as set of hyper-parameters, which are used for evaluation in Section 4.

3.2. Preprocessing Stage

Once we load the track dataset, we perform the*feature selection* by categorizing each feature based on its data type and utility:

- **Non-categorical features**: Numerical features suitable for analysis (e.g., *danceability*, *energy*, *loudness*).
- **Categorical features**: Non-numerical or descriptive attributes (e.g., *artist*, *song title*), retained only for presentation purposes.

Categorical features are excluded from further processing. The selected **non-categorical features** used in subsequent steps are:
danceability, *energy*, *loudness*, *speechiness*, *acousticness*, *instrumentalness*, *liveness*, *valence*, *tempo*, and *duration*. Once we choose only the non-categorical features, we apply *Min-max scaling* to the selected features to map them in the range $[0, 1]$.

The main idea of PCA is to reduce the dimensionality datasets containing a large number of interrelated features, while retaining as much as possible of the variation present in the dataset. PCA accomplishes this by transforming the data into uncorrelated principal components (PCs), ordered such that the first few PCs capture the majority of the dataset's variability. [12].

To select the number of principal components, we compute the individual and *cumulative explained variance* across all possible component counts. In our case, the maximum number of principal components is 10, because the total number of selected features is 10. Table 2 displays both the individual explained variance (percentage of coverage per principal component) and the cumulative explained variance (total percentage up to a given number of components). Using these measures, we assess the variability of the original data in each component or a selected number of components and depending on how much of the initial data we are willing to discard to reduce space complexity, we select the most appropriate number to achieve the proposed threshold.

Table 2. Explained Variance of Principal Components

Component/Number of Components	% Explained Variance	% Cumulative Explained Variance
1	43.006%	19.900%
2	19.9%	62.905%
3	13.279%	76.185%
4	7.538%	83.723%
5	6.788%	90.510%
6	3.408%	93.919%
7	3.098%	97.016%
8	2.408%	99.424%
9	0.444%	99.869%
10	0.131%	100%

We used K-Means for clustering tracks given its simplicity, scalability, convergence guarantees and ability to handle diverse cluster shapes (e.g., elliptical) [8, 18]. While K-Means is widely used for clustering, it has the following limitations [8]:

1. **Choosing the Number of Clusters**: To address this, we employed the *Elbow Method*, *Silhouette Score*, and *Calinski-Harabasz Index* as evaluation measures to determine the optimal cluster count.
2. **Sensitivity to Initial Values**: We mitigated this issue using the *K-Means++* initialization algorithm [3], which improves cluster stability.
3. **Scalability with High Dimensions**: This limitation is less relevant here, as we applied *PCA* for dimensionality reduction.
4. **Handling Varying Sizes and Densities**: A key limitation of K-Means is that it works best with evenly sized, ball-shaped clusters. Since real data often violates these assumptions, we considered this factor in our methodology.

To evaluate the optimal number of clusters used for K-Means, we considered the range of [2, 10] to choose from and the following evaluation measures: *Elbow method*, *Silhouette coefficient* and *Calinski–Harabasz index*.

The idea behind the *Elbow method* is that the explained variability changes rapidly for a small number of clusters and then slows down, resulting in an elbow shaped curve. The point at which the so-called elbow is formed indicates the number of clusters that should be optimal for a given clustering algorithm [16].

The *Silhouette coefficient* [16] is slightly stricter than the *Elbow method*, but it is better at identifying relationships within and between groups. This measure indicates whether individual points are correctly assigned to their clusters. When using this coefficient, we can interpret the measure $S(i)$ of a point i as follows:

- $S(i)$ close to 0 means that the point is between two clusters.
- If $S(i)$ is close to -1 then the target point should be in a different cluster.
- If $S(i)$ is close to 1 then the point belongs to the "correct" cluster.

The *Calinski–Harabasz index* is defined based on the assumption that clusters are very compact and are sufficiently distant from each other to form a good distribution. The index is calculated by dividing the variance of the sums of squares of the distances of individual objects to their cluster centre by the sum of squares of the distance between the cluster centres. The higher that index value, the better the division [16].

In the *track label classification* stage, we implemented a k-NN model to categorize new songs according to the K-Means cluster assignments. The model takes as input the dimensionally-reduced PCA features generated in the preprocessing stage.

For the testing phase of the classification model, we performed an 80-20 train test split of the dataset, which resulted in 144,176 songs for training set and 36,044 songs for testing set. We evaluated the performance for the number of neighbours k on a $[1, 30]$ range and we decided to choose $k = 25$, because it has a good enough accuracy, since only 412 out of 36,044 tracks had inaccurate labels, when compared to their actual label values.

In the *Lyrics TF-IDF Matrix Computation* stage, we compute the $TF\text{-}IDF$ values of all terms (or words) of the lyrics of each track. Let $W = \{w_0, w_1, \ldots, w_{|W|-1}\}$ be the set of words from the lyrics across all tracks and $T = \{t_0, t_1, \ldots, t_{|T|-1}\}$ the set of tracks grouped in a label determined in *Clustering* stage, where the track t, based on which recommendations are generated, is also found. According to [1], the term frequency tf is defined as the number of occurrences of a word w in the lyrics of a track l_t and the inverse document frequency IDF, which takes into account the document frequency of a word $df(w)$ (the number of occurrences of the word w across the set of track lyrics) and the total number of tracks $|T|$, is defined as in equation (2):

$$\begin{aligned} &idf : W \to [1, 1 + \ln\left(\frac{1 + |T|}{2}\right)] \\ &df : W \to [1, |T|] \\ &idf(w) = 1 + \ln\left(\frac{1 + |T|}{1 + df(w)}\right) \end{aligned} \tag{2}$$

Once all $tf(w, t)$ values of $TF\text{-}IDF$ matrix are computed for $\forall w, t,\ 0 \leq w < |W|, 0 \leq t < |T|$, they will be normalized through Euclidean norm, as depicted in equation (3):

$$TF\text{-}IDF_{|T|\times|W|} = \left[\frac{tf\text{-}idf(w,t)}{\|TF\text{-}IDF(t)\|}\right]_{0\le w<|W|, 0\le t<|T|}$$
$$\|TF\text{-}IDF(t)\| = \sqrt{\sum_{w=0}^{|W|-1}(tf\text{-}idf(w,t))^2} \tag{3}$$

$\|TF\text{-}IDF(t)\|$ is the Euclidean norm of the $|W|$-dimensional $TF\text{-}IDF$ vector of $\forall t \in T$.

3.3. Recommendation algorithm

The first step of the recommendation algorithm is to determine the cluster label of the input track.

Next, we identify the cluster label for the input track t. Then, using the TF-IDF matrix (created during preprocessing) which includes track t and all other tracks in the same cluster, we calculate lyrics similarity. This is done by computing cosine similarity (Equation 4) between track t's TF-IDF vector and all other tracks' vectors in its cluster.

$$\cos(t,t') = \frac{TF\text{-}IDF(t)\cdot TF\text{-}IDF(t')}{\|TF\text{-}IDF(t)\|\cdot\|TF\text{-}IDF(t')\|} \tag{4}$$

Then, we determined the similarities between each pair of tracks t and t' based on their PCA features, denoted by $PCS(t,t')$, as shown in equation (7). We then used Euclidean distance $D(t,t')$ computed based on the best p PCA features of tracks. The resulting distances were "min-max" scaled into $[0,1]$ using equations (5) and (6).

Finally, the similarity $S(t,t')$ between two tracks t and t' is computed by aggregating three values (see equation (9): i) the similarity of tracks based on their acoustic features, captured as $PCS(t,t')$; ii) the cosine similarity of lyrics of tracks t and t' (see equation (4)); iii) the scaled popularity to $[0,1]$ of track t.

$$D\colon T\times T\to[0,\sqrt{p}]$$
$$D(t,t') = \sqrt{\sum_{k=0}^{p-1}(PC_k(t)-PC_k(t'))^2} \tag{5}$$

$$D_{\text{MinMax}}\colon T\times T\to[0,1]$$
$$D_{\text{MinMax}}(t,t') = \frac{D(t,t')-\min(D)}{\max(D)-\min(D)} \tag{6}$$

$$PCS\colon T\times T\to[0,1]$$
$$PCS(t,t') = 1 - D_{\text{MinMax}}(t,t') \tag{7}$$

$$P_{\text{MinMax}}\colon T\times T\to[0,1]$$
$$P_{\text{MinMax}}(t) = \frac{P(t)-\min(P)}{\max(P)-\min(P)} \tag{8}$$

$$S\colon T \times T \to [0,1]$$
$$S(t,t') = \frac{P_{\text{MinMax}}(t) + \cos(t,t') + PCS(t,t')}{3} \tag{9}$$

Equation (9) deserves an explanation. Analyzing it, we observe that function S is not symmetric in t and t'. Apparently this could be problem, but actually it is not if we realize how $S(t,t')$ is used by our recommendation algorithm. t' acts as input track, based on which recommendations t are generated. Equation (9) actually says that the provided recommendations are those with a high popularity $P_{\text{MinMax}}(t)$ and that are similar with t' in lyrics $\cos(t,t')$ and acoustics $PCS(t,t')$.

4. System Evaluation

A recommendation system should provide recommendations of items that are rather similar to items seen in the past – known as exploitation capability, as well as of items different from those seen in the past – known as exploration capability. In this section, we initially provided our evaluation of the exploitation capability of our content-based recommendation system, considering the number of principal components, the number of clusters and the number of recommendations to evaluate the precision of the recommendations generated by the system across all test users available. Next, considering the average precision of the recommendations generated from each evaluation track, we determined the optimal number of recommendations. Contrary to our expectations, the results were not so different and promising regardless of the pair comprising the number of clusters and the number of principal components. As such, we came up with a new evaluation protocol of the experimental recommendation system, which was designed as follows:

1. **User-Track Interaction Simulation**

   We simulate interactions between a test user and tracks they've listened to multiple times.

2. **Recommendation Evaluation**

   We assess how many recommended tracks align with the user's listening history, considering tracks played $\geq$ 3 times as "liked." This evaluation uses a refined test dataset that focuses exclusively on users with 5-10 frequently played tracks, which they listened to at least 3 times.

In this section, we firstly describe the evaluation dataset and how it was prepared in 4.1, then the evaluation measures in 4.2 and finally the results derived from the evaluation protocol in 4.3.

4.1. User-Track Interaction Dataset Description & Preparation

In order to evaluate the quality of the track recommendation system, we used a dataset that contains user-track interactions. This dataset was created independently from the datasets of tracks and lyrics that were used for training the recommender system.

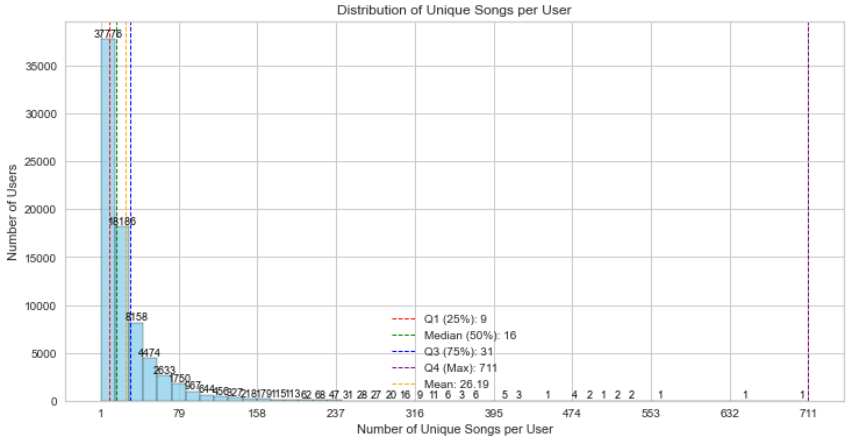

Fig. 2. Distribution of 76353 users across 9733 tracks

The dataset has 2 million entries represented by pairs (*user id*,*track id*) capturing interactions between users and tracks. The dataset contains 9733 unique tracks and 76353 unique users. Basically in our evaluation we simulate users interacting with tracks based on random samples extracted from this dataset.

In **Table 3**, we highlighted the distribution of values of each feature $f \in F$, where $F = \{$*danceability*, *energy*, *valence*, *acousticness*, *loudness*, *speechiness*, *liveness*, *instrumentalness*$\}$ across all songs which all test users listened to. Apart from **loudness**, which represents a value in dB, each feature belongs more or less to [0,1] range. As such, we can summarize the following conclusions:

- **Danceability**, **energy**, **valence** and **acousticness** are fairly evenly distributed. The mean of those features suggests that most songs exhibit a moderate tendency to be danceable, to have an overall neutral to slightly more positive emotional tone, moderate energy, and a mix of acoustic and electronic tracks.
- **Loudness** is highly variable and negative in value. The mean loudness indicates that the songs are generally quieter, since they have less than 0 dB sound levels. Furthermore, the difference between the loudest and the quietest song confirms the significant variation of sound levels.
- **Speechiness** is generally low, which means that most tracks have little ”spoken” content. The 75^{th} percentile confirms the lack of spoken content in most songs, but considering the maximum value, there are, however, some songs that can be very speech-dominant.
- **Liveness** is generally low, but in some songs it indicates that there is probably a live performance in the audio recording.
- **Instrumentalness** has a very low mean, suggesting that most tracks have vocals. However, despite the low value, the maximum value indicates that there are some tracks that are entirely instrumental.

Taking into account Figure 2, we plotted the distribution of user-track interactions by the number of users according to the number of unique songs played per user. Then we

categorize the interaction based on the minimum number of unique songs played per user (Q0), the first quartile of unique songs played per user (Q1), the median value of unique songs played per user (Q2), the third quartile of unique songs played per user (Q3) and the maximum number of unique songs played per user (Q4), as follows:

I.) $1 \leq$ number of unique songs played per user < 9 : 41706 users.
II.) $9 \leq$ number of unique songs played per user < 16 : 10911 users.
III.) $16 \leq$ number of unique songs played per user < 31 : 6848 users.
IV.) $31 \leq$ number of unique songs played per user ≤ 711 : 2691 users.

In addition, each user listened to each song a certain number of times (has a listen count). On average, a user listens to a song roughly 3 times. In both the median and the first quartile, a user listens to a song a single time. In the third quartile, a user listens to a song 3 times and the maximum number of times a user listened to a song is 2213.

As such, we decided that if a user listened to a song at least 3 times, we consider it a meaningful or relevant user-track interaction. Therefore, we filter out all rows where a user listened to a song less than 3 times. After filtering those rows out, we get the distribution of 62156 users in 9731 songs from Figure 3 and the following range values, similarly categorized, but with different values, from the previous enumeration:

I.) $1 \leq$ number of unique songs played per user < 2 : 9055 users.
II.) $2 \leq$ number of unique songs played per user < 5 : 18620 users.
III.) $5 \leq$ number of unique songs played per user < 11 : 18315 users.
IV.) $11 \leq$ number of unique songs played per user ≤ 283 : 16166 users.

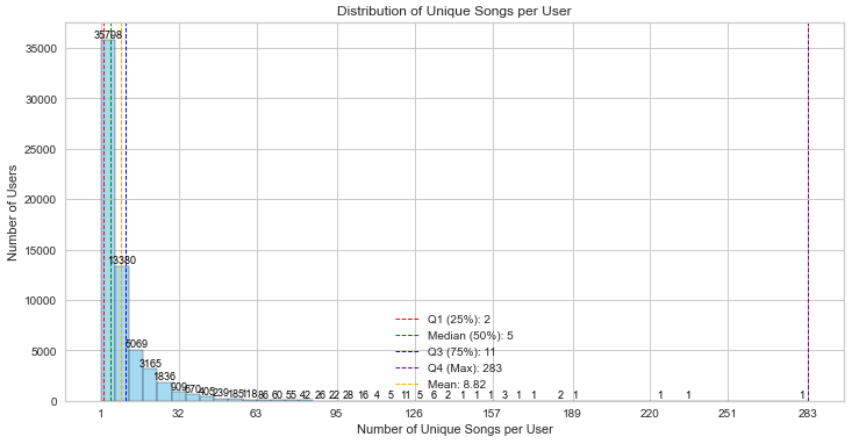

Fig. 3. Distribution of 62156 users across 9731 tracks after filtering

Table 3. Summary statistics for audio features

	Danceability	Energy	Valence	Loudness	Speechiness	Liveness	Acousticness	Instrumentalness
mean	0.533	0.48	0.517	-11.743	0.102	0.212	0.505	0.199
std	0.177	0.273	0.267	5.721	0.175	0.182	0.379	0.335
min	0	0.002	0	-42.66	0	0.017	0	0
25%	0.411	0.247	0.299	-14.993	0.035	0.1	0.098	0
50%	0.543	0.463	0.528	-10.831	0.045	0.138	0.529	0
75%	0.664	0.708	0.741	-7.452	0.076	0.271	0.901	0.262
max	0.982	1	0.984	0.781	0.968	0.996	0.996	1

4.2. Evaluation Protocol Description

For a fair evaluation protocol, we only considered the set of users U who listened to between 5 and 10 songs, and the set of unique songs T from user-track interactions, because that distribution contains a considerable number of unique users and tracks, as highlighted in Figure 3 and **Section 5.1**. For this sample of user-track interactions, we firstly created a profile for each user. The profile consists of the average values of each specific audio feature (*danceability*, *energy*, *valence*, *loudness*, *speechiness*, *liveness*, *acousticness* and *instrumentalness*) as highlighted in Definition (10), where:

- $P'(u)$ is the user profile of user $u \in U$
- $P'(u, f)$ is the user profile feature value of user u relative to an audio feature $f \in F$
- T_u is the set of tracks T played by a user $u \in U$,
- $f_{t,u}$ is the feature value of a track $t \in T$ which a user $u \in U$ listened to
- $|T_u|$ is the total number of tracks a user $u \in U$ listened to
- $|F|$ is the total number of features

$$\begin{aligned} P'(u) &= [P'(u, f)]_{|F|} \\ P'(u, f) &= \frac{\sum_{t_u \in T_u} f_{t,u}}{|T_u|} \end{aligned} \tag{10}$$

Once the user profile is determined, we randomly choose a track from the list of tracks the user listened to at least 3 times. Each track t_u has a probability $P(T_u = t_u) = \frac{listen\_count(u,t_u)}{\sum_{t \in T_u} listen\_count(u,t_u)}$ such that $P(T_u) = \sum_{t_u \in T_u} P(T_u = t_u) = 1$, where:

- $listen\_count(u, t_u)$ is the number of times a user u listened to a track t_u
- $\sum_{t \in T_u} listen\_count(u, t_u)$ is the total number of times a user u listened to all the tracks in T_u

After a track is randomly chosen, we use it as input in the recommender system to generate the top 5 most similar tracks. The recommendations are assessed as in Equations (11) and (12). Equation (11) measures on average how close specific feature values f_r of recommended tracks $R(t_u)$ are to the same features represented as the average values $P'(u, f)$ of the tracks T_u found in the user profile $P'(u)$, while Equation (12) computes the aggregated feature error $AFE(u)$ for each user $u \in U$ based on each individual feature from

Equation (11), which offers an overall assessment of similarity across all features $f \in F$ of all tracks $t_u \in T_u$ for each user $u \in U$.

$$MAE(t_u, f) = \frac{\sum_{r \in R(t_u)} |P(u, f) - f_r|}{|R(t_u)|} \tag{11}$$

$$AFE(u) = \frac{\sum_{f \in F} MAE(t_u, f)}{|F|} \tag{12}$$

4.3. Evaluation Results

In **Table 4**, we computed the mean absolute error $MAE(t_u, f)$ for each audio feature $f \in F$ between a randomly chosen track $t_u \in T_u$ played by each user $u \in U$ and all recommended tracks $R(t_u)$, and, the aggregated feature error $AFE(u)$ across all features $f \in F$ for each user $u \in U$, which are defined in Equations (11) and (12). Then, based on values from **Table 5**, we can notice that overall the recommendations provided to all test users are relevant based on their user profile, which consists of certain threshold values for each audio feature $f \in F$, because the difference in terms of audio features between the recommendations and the user preference pattern found in all audio features is barely noticeable, which is a good indicator of relevancy.

Table 4. Mean Absolute Error $MAE(t_u, f)$ of each feature $f \in F$ and Aggregated Feature Error $AFE(u)$ for each test user $u \in U$

User ID	Danceability	Energy	Valence	Loudness	Speechiness	Liveness	Acousticness	Instrumentalness	Aggregated Feature
0	0.066	0.119	0.319	1.559	0.01	0.055	0.569	0.001	0.337
1	0.082	0.424	0.159	6.765	0.023	0.01	0.794	0.03	0.941
2	0.094	0.318	0.125	7.048	0.022	0.123	0.166	0.266	0.885
3	0.28	0.035	0.156	0.817	0.611	0.138	0.457	0.001	0.312
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
18314	0.212	0.239	0.642	3.549	0.014	0.05	0.653	0	0.670

Table 5. Summary statistics for across all $MAE(t_u, f)$ and $AFE(u)$ values

	Danceability	Energy	Valence	Loudness	Speechiness	Liveness	Acousticness	Instrumentalness	Aggregated Feature
mean	0.151	0.206	0.202	4.886	0.065	0.095	0.287	0.179	0.759
std	0.082	0.133	0.126	3.469	0.133	0.105	0.217	0.299	0.469
min	0.005	0	0.015	0.229	0	0	0.001	0	0.11
25%	0.092	0.102	0.108	2.568	0.013	0.036	0.111	0.001	0.451
50%	0.132	0.169	0.165	3.939	0.023	0.063	0.215	0.012	0.627
75%	0.191	0.282	0.27	6.047	0.050	0.113	0.424	0.2	0.906
max	0.605	0.830	0.848	30.478	0.921	0.879	0.983	0.971	4.154

5. Conclusions and Future Works

In this extended version of the paper, we achieved the following contributions:

- We focused on evaluating a smaller but more meaningful set of user-track interactions, unlike the previous one, which was more vast, but exhibited a significant imbalance in user listening behavior. Specifically, a large proportion of users have interacted with few songs, while a small subset of users accounts for the majority of listening activity. The smaller subset used for the evaluation was achieved by filtering out users who listened to less than 3 songs and then considering only users who listened to between 5 and 10 songs. In other words, the number of songs per user had a range of [Q1, Q2), where Q1 is the first quartile of unique songs a user listened to and Q2 is the median value of unique played songs. As such, we ended up with 128,834 unique user interactions consisting of 18,315 unique users and 9,733 unique songs.
- We created a user profile for each user in the set of test users, where we considered the aggregated value of each specific audio feature in all the songs found in the set of users, who played between 5 and 10 songs at least 3 times.
- We compared how suitable the recommendations are for each feature with each user profile using the mean absolute error measure.
- We compared how suitable the recommendations are across all features with each user profile by averaging the mean absolute error measure of each feature involved.

As future work, we can enhance the recommender system by incorporating a weighting mechanism to capture more user preferences (track genre, release year, album, etc.) for all tracks which have features that allow this. In terms of system evaluation, we can improve the evaluation protocol by using more appropriate information retrieval measures mentioned in [6] and [5].

- **Precision**: evaluates the proportion of relevant tracks in top-k recommendations. Although it was done in the previous version of this paper, we realized that the setup of the recommender system was not designed appropriately so that the precision of recommendations to each user can be properly and successfully evaluated by taking into account the available human ground truth. Hence, this is one reason why the recommendation system should be revised.
- **Average Precision (AP)**: evaluates the quality of classification within the top-k recommendations.
- **Normalized Distributed Cumulative Gain (NDCG)**: evaluates a given ordered classification with an "ideal" classifciation (can be thought of human ground truth) to a condensed list of recommended tracks.
- **Hit-Rate**: evaluates if there is a least one relevant in top-k recommended items for each user in the set of test users. However, one significant drawback of hit-rate is that a hit that occurs in the first position is treated equally with a hit that occurs in the last position of the list of recommendations.
- **Average Reciprocal Hit-Rank (ARHR)**: addresses the limitation of the hit-rate that rewards each hit equally regardless of its position by rewarding hits that occur earlier in top-k list with higher weights. This way, the highest value of ARHR is equal to the hit-rate and occurs when all the hits occur in the first position, whereas the lowest value of the ARHR is equal to the ratio between the hit-rate and the number of test

users when all the hits occur in the last position in the list of the top-k recommendations.

References

1. 6.2 feature extraction - sci-kit learn documentation. https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction
2. Alex Hsieh, Owen Luo, T.P.: Music attributes and its effect on popularity, https://ahsieh53632.github.io/music-attributes-and-popularity/
3. Arthur, D., Vassilvitskii, S., et al.: k-means++: The advantages of careful seeding. In: Soda. vol. 7, pp. 1027–1035 (2007)
4. Burneau, B.: New website: lyrics.ovh (January 2017), https://ntag.fr/lyrics-ovh-only-the-lyrics/
5. Castells, P., Moffat, A.: Offline recommender system evaluation: Challenges and new directions. AI Magazine 43 (06 2022)
6. Deshpande, M., Karypis, G.: Item-based top-n recommendation algorithms. ACM Trans. Inf. Syst. 22(1), 143–177 (Jan 2004), https://doi.org/10.1145/963770.963776
7. for Developers, S.: Get several tracks' audio features. https://developer.spotify.com/documentation/web-api/reference/get-several-audio-features
8. Google Developers: K-Means Advantages and Disadvantages. https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages, last updated 2022-07-18 UTC
9. Khuat, V.: Music Recommendation Using Collaborative Filtering. https://portfolios.cs.earlham.edu/wp-content/uploads/2019/08/final_paper_draft_Vuong.pdf (2018), senior Capstone Experience, Earlham College
10. Lops, P., de Gemmis, M., Semeraro, G.: Content-based Recommender Systems: State of the Art and Trends, pp. 73–105 (01 2011)
11. Mandloi, K., Mittal, A.: Hybrid music recommendation system using content-based filtering and k-mean clustering algorithm. International Journal of Computer Sciences and Engineering 6, 1498–1501 (07 2018)
12. Mishra, S.P., Sarkar, U., Sanjay, S.T., Devi, D., Swain, P., Saikhom, R., Panda, S., Laishram, M.: Multivariate statistical data analysis- principal component analysis (pca). International Journal of Livestock Research 7(5), 60–78 (2017)
13. Neremzoiu, I.D., Bădică, A., Ganea, E.: Track recommender system based on lyrics, audio and popularity features. In: 2024 International Conference on INnovations in Intelligent SysTems and Applications (INISTA). pp. 1–6 (2024)
14. Nilashi, M., Bagherifard, K., Ibrahim, A.P.D.O., Alizadeh, H., Lasisi, A., Roozegar, N.: Collaborative filtering recommender systems. Research Journal of Applied Sciences, Engineering and Technology 5, 4168–4182 (04 2013)
15. Niyazov, A., Mikhailova, E., Egorova, O.: Content-based music recommendation system. vol. 29, pp. 274–279 (05 2021)
16. Rachwał, A., Popławska, E., Gorgol, I., Cieplak, T., Pliszczuk, D., Skowron, Ł., Rymarczyk, T.: Determining the quality of a dataset in clustering terms. Applied Sciences 13(5), 2942 (2023)
17. Ronak Doshi, Nithin Raj, A.R.S.: Song recommendation system (November 2017), https://github.com/ronak66/Song-Recommendation-System
18. Yuan, C., Yang, H.: Research on k-value selection method of k-means clustering algorithm. J 2(2), 226–235 (2019), https://www.mdpi.com/2571-8800/2/2/16
19. Zhang, X.: spotify-dataframe (2021), https://jovian.ai/zhangxm963/spotify-dataframe/v/39/files?filename=spotify-dataset-19212020-160k-tracks/data.csv
20. Çano, E.: Hybrid recommender systems: A systematic literature review. Intelligent Data Analysis 21, 1487–1524 (11 2017)

Ionuț-Dragoș Neremzoiu is a first-year PhD student in Computer Science at the University of Craiova, Faculty of Automation, Computers and Electronics. He also works as a Systems Engineer at the University's Faculty of Law. His current research focuses on Explainable LLM-based Recommender Systems, which is also the topic of his doctoral thesis. His broader research interests include artificial intelligence, machine learning, recommender systems, and large language models.

Andreea Liliana Bădică holds a PhD in Management from the University of Craiova, where she conducted research between 2019 and 2022. She is currently employed as a Sales Operations Specialist at BMW Group in Salzburg, Austria, where she contributes to operational processes related to ordering across the European region. Her academic background and professional experience reflect a strong interest in the intersection of business operations, strategic management, and international markets.

Received: April 18, 2025; Accepted: September 01, 2025.