**MUSIC RECOMMENDER SYSTEM**

**Group 8**

1. **Participant information**

* Do Nghiem Duc 20214892
* Tran Van Toan 20214932
* Nguyen Huu Nam 20210630

1. **Abstract**

Music is an integral part of our lives, transcending boundaries and connecting people through its universal language. In today's digital era, we are presented with an overwhelming abundance of music options, making it increasingly challenging to discover new tracks that align with our tastes. This has prompted the development of music recommender systems powered by machine learning algorithms, which aim to personalize the music listening experience by suggesting relevant and exciting songs to individual users.

In this project, we delved into the realm of machine learning in the context of music recommendation, exploring the algorithms, data, and techniques employed to create a robust and accurate music recommender system. Our music recommender system utilizes a content-based filtering approach, incorporating K-means clustering and neural networks.

1. **Data Analysis**
   1. Data source

* Our original dataset, sourced from Kaggle, comprises of 114,000 songs with their general attributes including audio features.
* We obtained an additional dataset by crawling Spotify's API, comprising Spotify's recommendations for the songs in our original dataset. Upon merging it with our original dataset, the resulting dataset contains approximately 680,000 songs, with around 350,000 being unique.
  1. Statistical properties

The dataset contains the following columns:

* **track\_id:** The Spotify ID for the track
* **arstists:** The artists' names who performed the track.
* **album\_name:** The album name in which the track appears
* **track\_name:** Name of the track
* **popularity:** The popularity of a track is a value between 0 and 100, with 100 being the most popular.
* **duration\_ms:** The track length in milliseconds
* **explicit:** Whether or not the track has explicit lyrics (true = yes it does; false = no it does not OR unknown)
* **danceability:** Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable
* **energy:** Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
* **key:** The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
* **loudness:** The overall loudness of a track in decibels (dB).
* **mode:** Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0
* **speechiness:** Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording, the closer to 1.0 the attribute value.
* **accousticness:** A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
* **instrumentalness:** Predicts whether a track contains no vocals. The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.
* **liveness:** Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
* **valence:** A measure from 0.0 to 1.0 describing 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.
* **time\_signature:** An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of 3/4, to 7/4.
* A screenshot of a computer

  Description automatically generated**track\_genre:** The genre in which the track belongs.

Correlations between features

A group of blue and white graphs

Description automatically generated

A graph with blue bars

Description automatically generatedA graph with a bar graph

Description automatically generated

Distribution of the features

We have made the decision to build our model exclusively on the 9 audio features, namely acousticness, danceability, energy, instrumentalness, liveness, valence, tempo, speechiness and loudness.

* 1. Preprocessing

After removing duplicates and null value, the dataset shrinked from 114,000 songs to about 90,000 songs. Using the selected features, we apply the Standard Normal Distribution to normalize the data and scale the features accordingly.

A screenshot of a computer screen

Description automatically generated

Correlations between scaled features

1. **Approaches**

We have two different approaches to the problem:

* 1. K-means

4.1.1 Choosing k

First, we apply the “elbow method”, which is to plot the sum of squared distances (SSE), the total distance between each data point and its assigned cluster centroid, against different values of k (from 2 to 20). The intuition behind this method is that as k increases, the SSE typically decreases since each data point will be closer to its centroid. However, beyond a certain point, the decrease in SSE becomes less significant, resulting in a smoother curve resembling an "elbow." The optimal value of k is often chosen at the point where the decrease in SSE significantly slows down, indicating diminishing returns in clustering improvement.

A graph with a line

Description automatically generated

“Elbow method” plot

Although it is pretty ambiguous as the plot does not contain a sharp elbow, there is still the sign of an elbow forming at the point of k = 7.

However, we acknowledge that the interpretation of the elbow curve can be somewhat subjective and unreliable. To obtain a more objective assessment of the number of clusters, we will employ the use of the silhouette score. The silhouette score measures how well each data point fits within its assigned cluster compared to other clusters. It ranges from -1 to 1, where a score closer to 1 indicates well-separated clusters.

The silhouette score for each datapoint is calculated using the following formula:

where is the average distance (a) between the data point and all other data points within the same cluster, and is the average distance between the data point and all data points in the nearest neighboring cluster (i.e., the cluster with the next closest centroid). The silhouette score for the entire clustering situation will be calculated as the average silhouette score of the dataset.

After computing the silhouette scores for k values ranging from 2 to 20, we plotted the results to gain insights into the clustering performance and here is the result:

A graph with red lines

Description automatically generated

Silhouette score plot

Hence, we decided that k = 7 is a reasonable amount of cluster.

4.1.2 Clustering

After applying K-means clustering with k = 7, we analyzed the distribution and attributes of the data points within each cluster to gain insights into the composition and characteristics of the clusters.

A graph of a bar graph

Description automatically generated

A diagram of a cluster of colored dots

Description automatically generated

A colorful dots on a white background

Description automatically generated

A graph of a cluster of colored dots

Description automatically generated

4.1.3. Recommendations

After we are done with clustering, the input will be a list of track\_id and the output for each of the song will be the Spotify link of the 5 nearest songs from the same cluster with the input song.



* 1. Neural networks #1

This second model using neural networks is designed to learn exclusively from the Spotify API. We worked under the assumption that Spotify's recommender system is mostly based on its users' preferences. Given the vast number of users on the music platform, we believed that utilizing the recommendations provided by the API would accurately reflect people's music choices.

* + 1. Model
* The model consists of 2 parts:
* A neural network that is utilized to learn an embedding of each song in the dataset, ensuring that the embeddings of songs in the same recommended list, along with the input, are closely grouped together.
* A second neural network is employed to learn a mapping from the tracks' attributes to the embeddings. This is done to facilitate generalization to songs that are not yet included in our dataset.
* We utilized the Keras module from the Tensorflow library for this model.
  + - 1. The first neural network

This neural net utilizes the Keras’ Embedding layer, which learns a matrix where each row represents the embedding of its respective input.

* Input and output:
* Input: This neural network has two inputs: the song and its "group." The "group" refers to songs that share the same recommended list, including the input song, from Spotify, forming a cohesive group. These inputs are fed into two separate layers: one for the song embedding and another for the group embedding.
* Output: The output is the dot product between the song embedding and the group embedding. If a song belongs to a group, their embeddings' dot product will be 1. If they are unrelated, the dot product will be -1. (…)
* Architecture:

The net consists of 5 layers:

* 2 Input layers, one for songs and one for groups
* 2 Embedding layer, also one is for songs, and one is for groups, that takes the respective Input layer’s output as input.
* 1 Dot layer, which serves as the output of the neural net, calculates the dot product between the embeddings of the input song and the input group. The loss function we use here is the mean absolute error (MAE) between the output and the label (1 or -1).

Our optimizer for the network is the Adam optimizer, which is stochastic gradient descent but with an adaptive learning rate schedule.

* + - 1. The second neural network:

This network utilizes the Keras’ Dense layer, which is a layer in the usual fully connected neural networks.

* Input and Output:
* Input: The input consists of a song's attributes, divided into two parts: the audio attributes of the song, including danceability, etc., and the genres of the song.
* Output: The output is the embedding of the input song.
* Architecture:

The second neural network consists of 7 layers:

* 2 Input layers, one for the audio attributes and one for the genres, where the genres are one-hot encoded.
* 2 consecutive Dense layers, taking the genres as input to reduce the dimensionality of the encoding.
* 1 Concatenate layer, which combines the audio attributes with the reduced encoding of the genres.
* 2 additional consecutive Dense layers, the first one takes the Concatenate layer output as input, and the second one produces the final output.

All Dense layers utilize the hyperbolic tangent (tanh) activation function. The loss function employed here is cosine similarity.

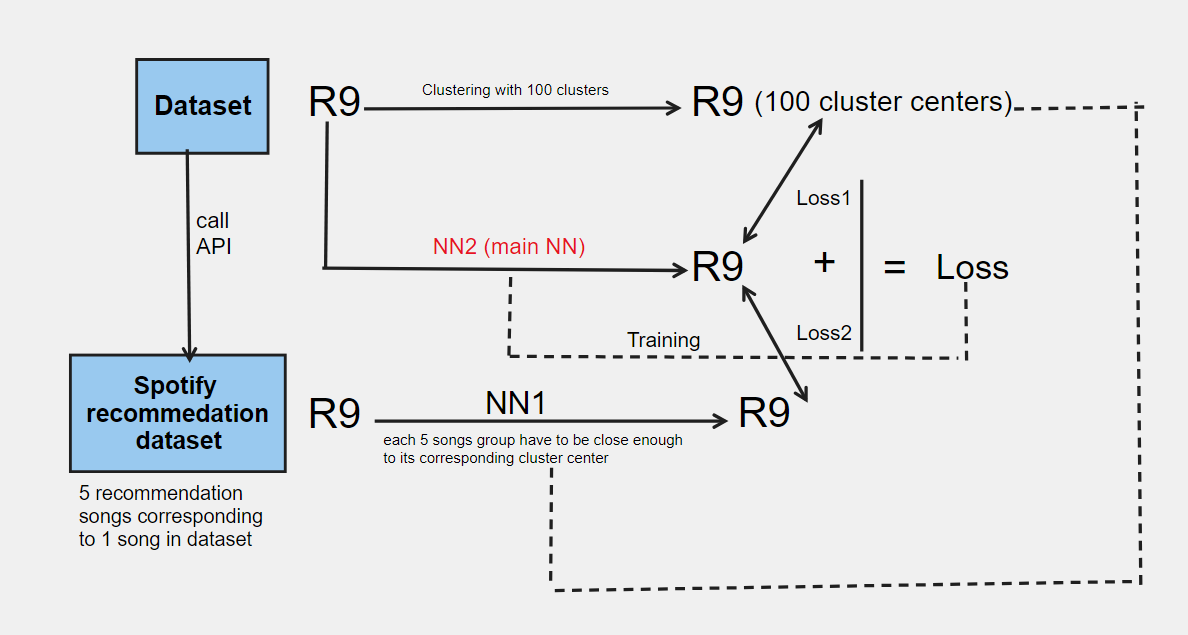
* + 1. Training
       1. First neural network
* In the first neural network, we began by indexing the track\_id and group for easier control and organization. For each track\_id, we determined its closeness to a group. If a song belonged to a group, we referred to them to be of "direct relation." If a song belonged to another group that shared at least one song with the directly associated group, we considered them to be of “of degree 2 related." This notion of closeness extends to higher orders, such as "of degree k+1 related" for songs in a group that are not within the kth degree but share at least one song with a kth degree close group.
* To capture the idea that two groups sharing songs are essentially the same, we included all pairs of (track\_id, group) that were " of degree 2 related" in the "track\_group\_pairs" set. Additionally, due to the large number of parameters in the first neural network (approximately **13 million**), we also added all pairs of (track\_id, group) that were " of degree 3 relation " to the set, resulting in a total of around **111 million** pairs of related tracks and groups.
* We chose an embedding size of 25 for both the songs and the groups.
* During training, we employed a batch size of **1024** and observed the convergence of the loss function after **4** epochs, with a final value of **0.7329.** As our main objective was to have the embedding align with our notion of closeness between pairs, the loss function served as our primary metric.
  + - 1. Second neural network
* After removing all duplicates from our dataset, our goal is to train the neural network to map the audio attributes and genres to the embedding space. To achieve this, we normalize the audio attributes using the MinMaxScaler() from the scikit-learn library. Additionally, we apply one-hot encoding to the genres. In the case of songs with multiple genres, the encoding is the sum of the encodings for each individual genre.
* We then feed these processed inputs to our second neural network and focus on parameter tuning to minimize overfitting while maintaining an acceptable loss margin.
* Since the purpose of this network is to give a crude approximation of the underlying function that we assume to be exists, we only train this network for 1 epoch, with all of our data in the training set.
  + 1. Testing recommendations:
* In our code implementation, we incorporated functionality to provide recommendations for both tracks that are already present in our dataset and tracks that are outside of our dataset. The code takes into account the weight of the track, obtained either directly from the output of the first network or estimated from the output of the second network. Based on these weights, the code returns the closest tracks after mapping.
* By leveraging the mapping obtained from the neural networks, the code is able to generate recommendations that are closely related to the input track, regardless of whether the track is already in our dataset or not. This allows for a broader range of recommendations, expanding beyond the tracks present in our initial dataset.
* A screenshot of a computer

  Description automatically generatedWe test it with some songs from the Spotify website, like: [Cruel Summer – Taylor Swift](https://open.spotify.com/track/1BxfuPKGuaTgP7aM0Bbdwr) <https://open.spotify.com/track/1BxfuPKGuaTgP7aM0Bbdwr?si=e34231578d5f4120>
* The results obtained (as saved in our .ipynb file) provide recommendations that, in our subjective opinion, exhibit relevance to the input songs. These recommendations may be relevant in terms of factors such as tempo, tones, genres, and other musical characteristics.
* However, it is important to note that in certain cases, the recommendations may not align perfectly with user preferences or other non-musical factors. For example, recommendations might not take into account factors like user languages or other contextual considerations that could influence the appropriateness of the recommendations.
* Ultimately, while the recommendations align with our defined notion of closeness and the desired objective of the system, it's essential to consider user-specific preferences and other non-musical factors when evaluating the suitability of the recommendations in practice.
  1. Neural networks (2) + K-means

This model is the combination of 4.1 and 4.2, resulting in semi-supervised learning.

4.3.1. Model architecture

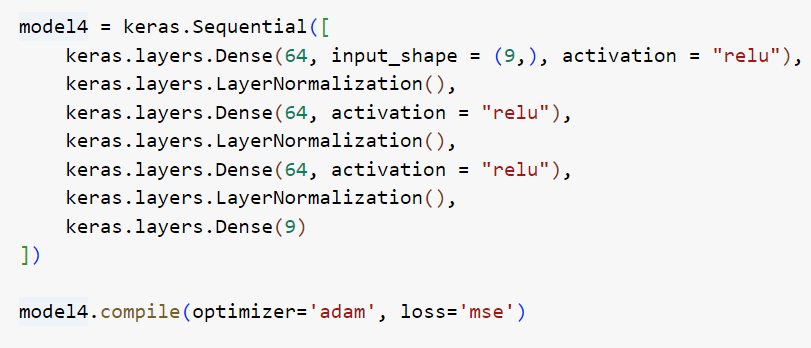
We came up with this architecture, where the main model will be learned based on clustering process (unsupervised learning) and NN1 (supervised learning). By doing this, our recommended songs will benefit from both Spotify recommendation system and our clustering work.



4.3.1.1. NN1

This neural net is to map Spotify recommended songs to other space where each 5 songs will be close to their original song’s cluster center.

NN1 contains 3 simple blocks of multilayer perceptron with layer normalization.



4.3.1.2. NN2 (main NN)

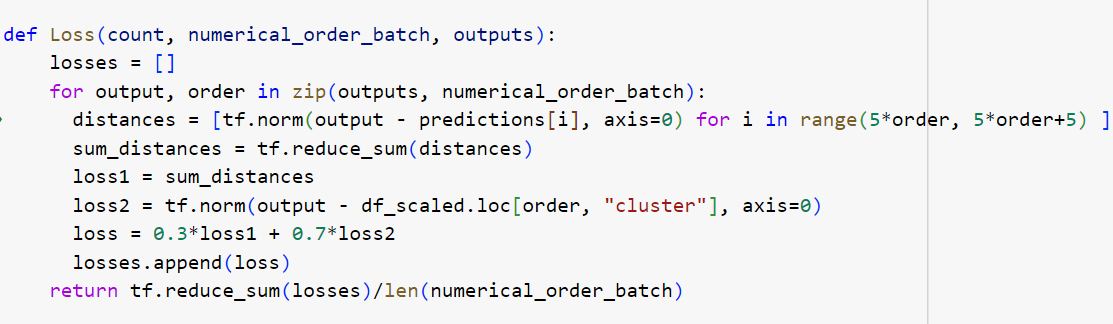
As the loss of NN2 is complicated, we have to code the neural nets from scratch.

- NN2 shares similar architecture with NN1

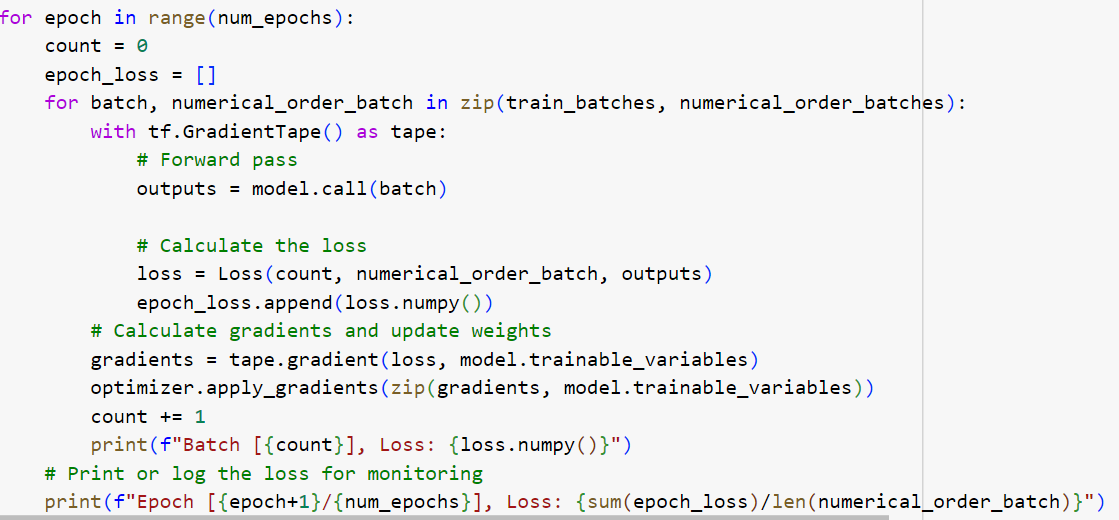


-

Where loss1 = sum (distance of current song to 5 Spotify labeled songs) and loss 2 = distance of current song to its cluster center



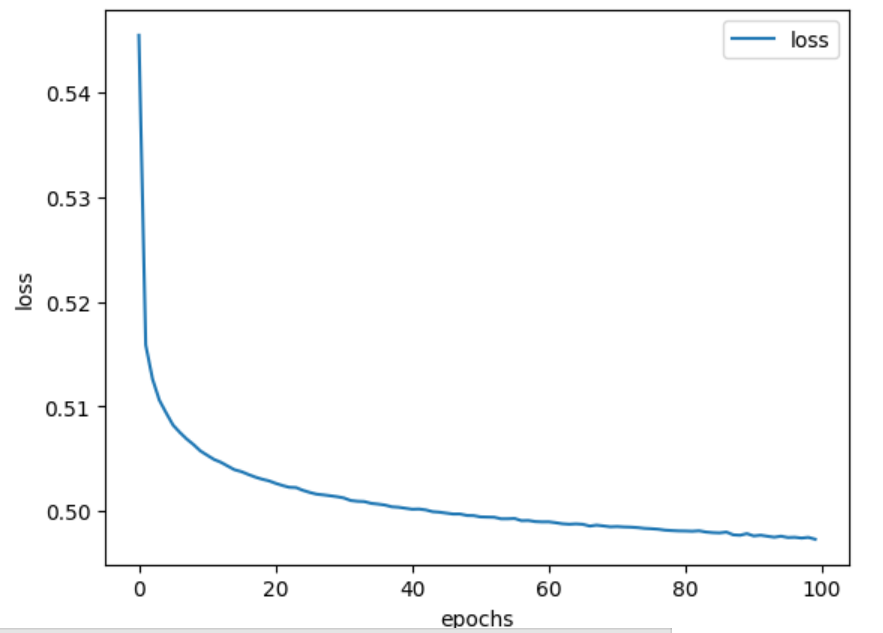
Our training process from scratch.



4.3.2. Training

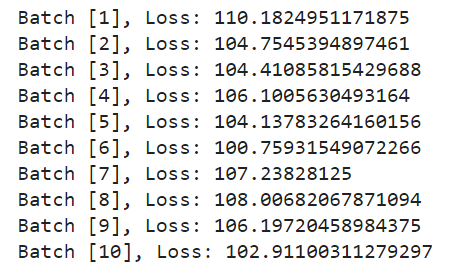
4.3.2.1. NN1

- After many experiments with different architectures, batch sizes, optimizers and learning rates, we decided to choose the above NN1 with the following result.



4.3.2.2. NN2

Unfortunately, NN2 does not converge well. We decided to save model parameters when the loss is around 96, which we notice as the lowest loss during the training process. Here are the first 10 batches during training.



1. **Findings, discussions, and conclusions**

* K-means clustering: While the clusters generated by the algorithm may not exhibit highly distinct boundaries or clear separations, this level of differentiation is deemed adequate for our purposes. This observation can be attributed to the characteristics of the dataset, which likely exhibits a wide distribution within the dataset. However, this is an expected outcome given the inherent diversity and variability in the data.

A screenshot of a computer screen

Description automatically generated

* Second model using neural networks: while the recommendations align with our defined notion of closeness and the desired objective of the system, it's essential to consider user-specific preferences and other non-musical factors when evaluating the suitability of the recommendations in practice.

Also, the amount of data needed for the second neural network of the model to be well-learned is much larger than expected, and this is something that we should improve on in the future.

1. **Problems and difficulties**

* Rating the performance of the models:
* The second neural network of model 2 keeps overfitting because of the lack of data. Since the dataset is limited, we decided to undertrain it to be as a crude approximation.
* Code from scratch in model 3 is challenging. The model takes lots more time training also.

1. **Proposal on future work**

* Differently weighted attributes: In our current approach, we treat all 9 audio features equally in terms of their significance. However, recognizing that certain attributes may have greater relevance than others, we intend to delve deeper into the domain knowledge of music to ascertain their relative importance and assign appropriate weights.
* Additional features to consider: While our current models solely rely on the 9 audio features, we recognize the potential influence of text-based attributes (such as artists) and categorical features (such as genre) on users' music preferences. It is important to conduct a thorough exploration and determine the extent of impact these features may have in order to enhance the personalization and overall performance of our models.
* Collaborative-based filtering: We propose to gather more data from Spotify in order to learn the assumed underlying function of people’s music preference to enhance our recommendation system.

1. **References**

* [Neural Network Embedding Recommendation System | Kaggle](https://www.kaggle.com/code/willkoehrsen/neural-network-embedding-recommendation-system)

Libraries that we used:

*Tensorflow, Numpy, Joblib, Pickle, …*