



Documentation On  
**Music Recommendation System  
Using Machine Learning  
And  
Flask Based Web-Integration**

# **Data Science 2024-25**

**Submitted By:**

**AARZOO RATHAUR (Roll No: DS24070001)  
DIPALI KANASE (Roll No: DS24070004)  
BUNTY VIRWANI (Roll No: DS24070003)**

**Dr. Shantanu Pathak  
Project Guide**

**Mr. Keshav Kumar  
Course Director**

## Declaration

We, the undersigned hereby declare that the project report titled, **“Music Recommendation System Using Machine Learning And Flask-based Web-Integration”** written and submitted by us to DYPIMS, in the fulfilment of requirement for the award of degree of Data Science Course 2024-25 under the guidance of Dr Shantanu Pathak. It is our original work and we have not copied any code or content from any source without proper attribution, and we have not allowed anyone else to copy our work. The project was completed using Python and ML libraries as well as the Flask-based web-framework. The project was developed as part of our academic coursework. We also confirm that the project is original, and it has not been submitted previously for any other academic or professional purpose.

Place:

Signature:

Date:

Name:

## **ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to Dr Shantanu Pathak, Project Guide, for providing us with the guidance and support to complete this academic project. Their valuable insights, expertise and encouragement have been instrumental in the success of this project.

We would also like to thank our fellow classmates for their support and cooperation during the project. Their feedback and suggestions were helpful in improving the quality of the project.

We would like to extend our gratitude to Mr Keshav Kumar, Course Director, for providing us with the necessary resources and facilities to complete this project. Their support has been crucial in the timely completion of this project.

Finally, we would like to thank our respective families and friends for their constant encouragement and support throughout the project. It is their relentless trust and support that has driven each one of us forward in your endeavours.

We thank each and every one mentioned above for their support and guidance in completing this academic project.

## ABSTRACT

In the modern era of digital streaming, delivering personalized music suggestions is essential for enhancing user engagement. This project introduces a **music recommendation system** based on unsupervised machine learning techniques and powered by the **Flask-web-based-framework** to provide tailored song recommendations.

The system primarily employs the **k-means clustering technique** to cluster tracks based on attribute similarities and leverages the power of PCA in achieving dimensionality reduction for reduction in data complexity as well as enabling better visualizations.

Consequently, it ensures a reliable mechanism to identify relevant features of the user-input and suggest songs that align with individual tastes.

This project demonstrates the effectiveness of combining **Flask's lightweight architecture** with **ML-driven recommendation techniques** to create a robust and adaptive music recommendation platform, offering users a more immersive and personalized listening experience.

# TABLE OF CONTENTS

<b>1. Introduction</b>	<b>7</b>
1.1 Motivation/Problem Statement	9
1.2 Objective	10
<b>2. Exploratory Data Analysis</b>	<b>11</b>
2.1 Dataset: Source and Description	12
2.2 Attributes/Feature-Set in the Data	13
2.3 Data Characteristics	14
2.4 Data Visualization	15
<b>3. Interactive Dashboard using Power BI</b>	<b>17</b>
3.1 Visualizations	18
3.2 Design Choices	18
3.3 Data Insights	19
3.4 Significance of the Dashboard	19
<b>4. Model Execution</b>	<b>20</b>
4.1 Data Pre-processing	21
4.2 Determination of the number of clusters	22
4.3 K-Means Clustering Algorithm	23
4.4 Dimensionality Reduction using PCA	24
4.5 Cluster Visualization	25
4.6 Cluster-based Recommendation System	26
4.7 UI/Web-Page Integration with Flask-Framework	27
<b>5. Model Workflow and Output</b>	<b>32</b>
<b>6. Project Requirements</b>	<b>34</b>
<b>7. Conclusion and Future Scope</b>	<b>36</b>
<b>8. References</b>	<b>38</b>

## **List of Figures**

Fig 1: Dataset Description from the data-source	12
Fig 2: List of attributes of the Dataset	13
Fig 3: Information on Data-attributes	13
Fig 4 Count of Duplicate Instances	13
Fig 5: Boolean Attribute “Mode”	13
Fig 6: Integer Attribute “Key”	13
Fig 7: Count of “Track_Album_Names”	14
Fig 8: Count of “Track_Artist”	14
Fig 9: Count of “Playlist_Name”	14
Fig 10: Count of “Playlist_Genre”	14
Fig 11: Distribution of Numerical Variables	15
Fig 12: Relationship between Musical Features and Popularity	15
Fig 13: Track Popularity by Playlist Genre	16
Fig 14: Interactive Dashboard created using Power BI	18
Fig 15: Relevant Music Features Scaled	21
Fig 16: Cluster-Count Determination by Elbow Method	22
Fig 17: KMeans Object and Cluster-component counts	23
Fig 18: Distributions of Songs by Cluster	23
Fig 19: PCA object-creation with n=2	24
Fig 20: Cluster-Visualization with Cluster-Centres after PCA (n=2)	25
Fig 21: Output of 10 Recommended Songs upon entry of an Input	25
Fig 22: UI/HTML Page for the user to get recommended songs	27
Fig 23: Drop-Down List for Input Songs	28
Fig 24: UI feature to Input song-name by typing	29
Fig 25: UI output if song-name not present in the database	30
Fig 26: Output of the Song Recommendation System in the UI	31

# **CHAPTER 1: INTRODUCTION**

# 1.INTRODUCTION

In an era of digital distribution and consumption of both products and services, be it music or films, e-commerce platforms or social-media and news-portals, **personalized recommendations** are not just a usual observation but have become a minimum practical expectation.

This project works on data obtained from a music streaming platform, “**Spotify**”, where recommendations of music tracks are made based on past choices of the users.

This **music recommendation system** is an unsupervised learning problem. It has been built using a **Flask-based framework** aided by relevant explorative data analysis as well as machine learning algorithms at the back-end. It utilizes **identified attributes** of the songs/music-tracks such as energy, loudness, tempo, etc., to analyse users’ preferences and recommend them **similar** tracks through usage of **segmentation techniques** such as **k-means clustering**.

The final output has been obtained in an HTML page through integration with a Flask-framework, where recommendations can be obtained upon entering any single music track’s title.

A dashboard has also been made using **Power BI** displaying interactive features from the dataset.



## **1.1 Motivation/Problem Statement**

Recommendation Systems primarily serve to

- enhance the users' experience by offering them options
  - similar to their historical choices,
  - similar to other users' choices with similar interests, and
  - that are based on their ratings of options used by them

Significantly, they are also essential for the platforms themselves for the sake of

- enhancing the users' engagement, and consequently
- enhancing their own revenues, be it through
  - paid subscriptions,
  - profits from direct sales,
  - sales commissions, or
  - even the platform-hosted advertisement earnings.

To this end, the study and use of a recommender system is quite appropriate and useful.

This is done here for making music recommendations to users of Spotify, a music-streaming app.

## 1.2 Objective

The project covers one of the several available methods for making recommendations of items to users of any platform, be it e-commerce or entertainment.

This project covers the design, development and deployment of an unsupervised machine learning algorithm to offer music recommendations to Spotify users.

The scope of work includes:

- **Data Collection:** The data has been collected from **Kaggle**, an online database repository with specifically identified sets of features in songs like energy, danceability, tempo, etc.
- **Data Preprocessing:** Where needed, the dataset has been scaled using the **StandardScaler()** function to enable normalized operations on it thereafter.
- **Model Selection and Training:** The **k-means clustering algorithm** is applied since the problem at hand is an unsupervised machine learning problem.
- **Dimensionality Reduction:** Reduction in Complexity is obtained by reducing the dimensions using **Principal Component Analysis (PCA)**.
- **Deployment and Integration:** Integrating the model into a web-based interface using the **Flask-framework**.
- **Dashboard Creation:** For analysing the interaction of various different features through a dashboard made through **Power BI**.

## **CHAPTER 2: EXPLORATORY DATA ANALYSIS**

## 2.Exploratory Data Analysis

### 2.1 Dataset: Source and Description

The given dataset has been taken from Kaggle, a popular data repository.

The specific URL for collecting this data is:

<https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs/data>

It consists of 32833 instances and 23 features that are attributes of the songs in the collection.

The direct description of the feature-set in the data can be obtained from a readme file available at its source URL, shown below:

```
# `spotify_songs.csv`

|variable|class|description|
|---|---|---|
|track_id|character|Song unique ID|
|track_name|character|Song Name|
|track_artist|character|Song Artist|
|track_popularity|double|Song Popularity (0-100) where higher is better|
|track_album_id|character|Album unique ID|
|track_album_name|character|Song album name|
|track_album_release_date|character|Date when album released|
|playlist_name|character|Name of playlist|
|playlist_id|character|Playlist ID|
|playlist_genre|character|Playlist genre|
|playlist_subgenre|character|Playlist subgenre|
|danceability|double|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|double|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. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.|
|key|double|The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation . E.g. 0 = C, 1 = C#/Db, 2 = D, and so on. If no key was detected, the value is -1.|
|loudness|double|The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typical range between -60 and 0 db.|
|mode|double|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|double|Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. 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.|
|acousticness|double|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|double|Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.|
|liveness|double|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|double|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|double|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.|
|duration_ms|double|Duration of song in milliseconds|
```

**Fig 1: Dataset Description from the data-source**

## 2.2 Attributes/Feature-Set in the Data

Since this dataset provides an unsupervised problem to work with, there are no class-labels and hence no classes nor any attribute defined as a class.

The feature-set, i.e., the set of attributes of the songs in the dataset can be seen by the following code-output:

```
df.columns

Index(['track_id', 'track_name', 'track_artist', 'track_popularity',
      'track_album_id', 'track_album_name', 'track_album_release_date',
      'playlist_name', 'playlist_id', 'playlist_genre', 'playlist_subgenre',
      'danceability', 'energy', 'key', 'loudness', 'mode', 'speechiness',
      'acousticness', 'instrumentalness', 'liveness', 'valence', 'tempo',
      'duration_ms'],
      dtype='object')
```

Fig 2: List of attributes of the Dataset

Of these, none of them have been found to have null or duplicate values, while their datatypes can be seen from the output below:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32833 entries, 0 to 32832
Data columns (total 23 columns):
 #   Column                Non-Null Count  Dtype
---  ---
 0   track_id              32833 non-null  object
 1   track_name            32828 non-null  object
 2   track_artist          32828 non-null  object
 3   track_popularity      32833 non-null  int64
 4   track_album_id        32833 non-null  object
 5   track_album_name      32828 non-null  object
 6   track_album_release_date 32833 non-null  object
 7   playlist_name         32833 non-null  object
 8   playlist_id           32833 non-null  object
 9   playlist_genre        32833 non-null  object
10   playlist_subgenre     32833 non-null  object
11   danceability          32833 non-null  float64
12   energy                32833 non-null  float64
13   key                   32833 non-null  int64
14   loudness              32833 non-null  float64
15   mode                  32833 non-null  int64
16   speechiness           32833 non-null  float64
17   acousticness          32833 non-null  float64
18   instrumentalness       32833 non-null  float64
19   liveness              32833 non-null  float64
20   valence               32833 non-null  float64
21   tempo                 32833 non-null  float64
22   duration_ms           32833 non-null  int64
dtypes: float64(9), int64(4), object(10)
memory usage: 5.8+ MB
```

Fig 3: Information on Data-attributes

```
df.duplicated().value_counts()

False    32833
Name: count, dtype: int64
```

Fig 4 Count of Duplicate Instances

```
df['mode'].value_counts()

mode
1    18574
0    14259
Name: count, dtype: int64
```

Fig 5: Boolean Attribute “Mode”

```
df['key'].describe()

count    32833.000000
mean         5.374471
std         3.611657
min          0.000000
25%          2.000000
50%          6.000000
75%          9.000000
max         11.000000
Name: key, dtype: float64
```

Fig 6: Integer Attribute “Key”

## 2.3 Data Characteristics:

- Categorical Features: 10 features are categorical in nature.
- Numeric Features: 13 features are numeric in nature, including
  - 3 with integer-datatype, and
  - 9 with float datatype.
  - 1 with Boolean datatype (“**mode**”).

It has also been noted (fig 6) that one numeric attribute, “key” has discrete integer values between 0 and 11.

Unique values of some of the attributes with the categorical data-type were explored and are being displayed below:

df['track_album_name'].value_counts()		df['track_artist'].value_counts()	
track_album_name		track_artist	
Greatest Hits	139	Martin Garrix	161
Ultimate Freestyle Mega Mix	42	Queen	136
Gold	35	The Chainsmokers	123
Malibu	30	David Guetta	110
Rock & Rios (Remastered)	29	Don Omar	102
...		...	
Blonde comme moi	1	Underworld	1
Every Second Counts	1	The Witches	1
Birdy (Deluxe Version)	1	Tess Parks	1
Standing In The Way Of Control	1	Mick Harvey	1
Typhoon/Storm	1	Mat Zo	1
Name: count, Length: 19741, dtype: int64		Name: count, Length: 10692, dtype: int64	

Fig 7: Count of “Track Album Names”      Fig 8: Count of “Track Artist”

Accordingly, 19,741 unique track albums and 10962 unique track artists have been found in the dataset.

df['playlist_name'].value_counts()		df['playlist_genre'].value_counts()	
playlist_name		playlist_genre	
Indie Poptimism	308	edm	6043
2020 Hits & 2019 Hits â Top Global Tracks 8Y8Y8Y	247	rap	5746
Permanent Wave	244	pop	5507
Hard Rock Workout	219	r&b	5431
Ultimate Indie Presents... Best Indie Tracks of the 2010s	198	latin	5155
...		rock	4951
CSR 103:9 (GTA: SA)	7	Name: count, dtype: int64	
Big White Room-Jessie-J	7		
TOP 50 GLOBAL 2020 UPDATED WEEKLY 889 WORLDWIDE	6		
ALPAS Music Festival	3		
Post-Teen Pop	1		
Name: count, Length: 449, dtype: int64			

Fig 9: Count of “Playlist Name”

Fig 10: Count of “Playlist Genre”

Likewise, 449 playlists and 6 unique genres have been found. Most other attributes were explored in this manner.

## 2.4 Data Visualization

### Variable Distributions

The distributions of the numerical variables (except for the Boolean variable, “mode”) are being displayed by the set of graphs given below:

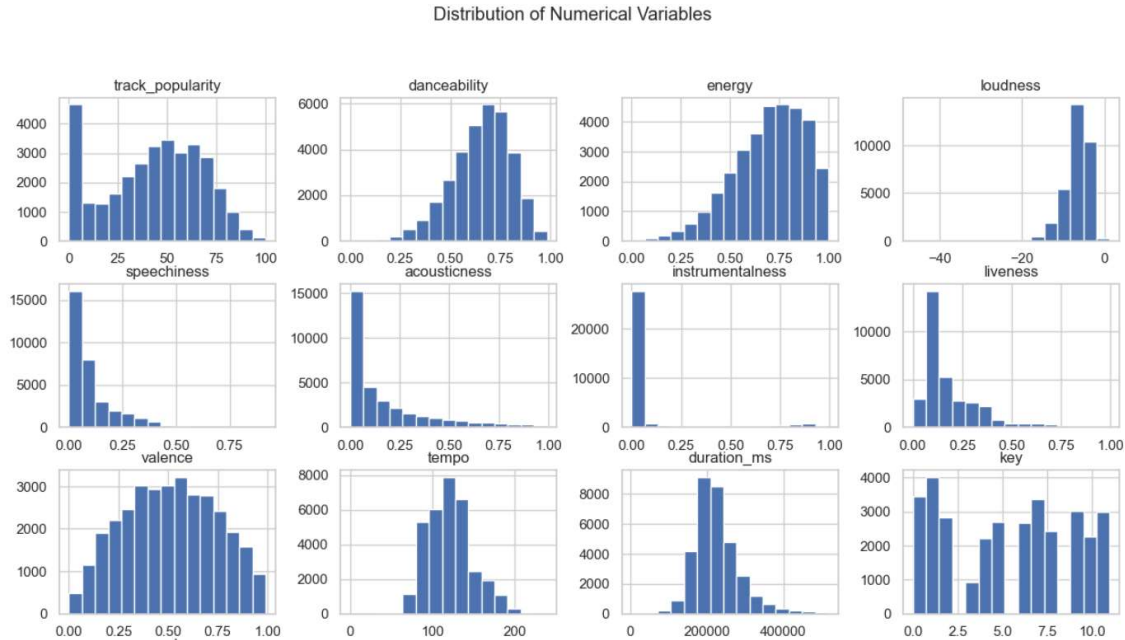


Fig 11: Distribution of Numerical Variables

### Musical Features by Popularity

The track popularity based on musical features such as “danceability”, “energy”, “valence” and “tempo” were visualized as follows:

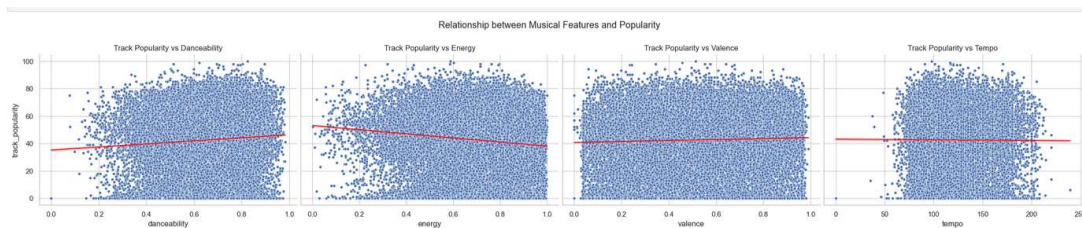


Fig 12: Relationship between Musical Features and Popularity



## Track Popularity by Playlist Genre

The average popularity of the playlist tracks grouped by their genre is visualized in the following manner:

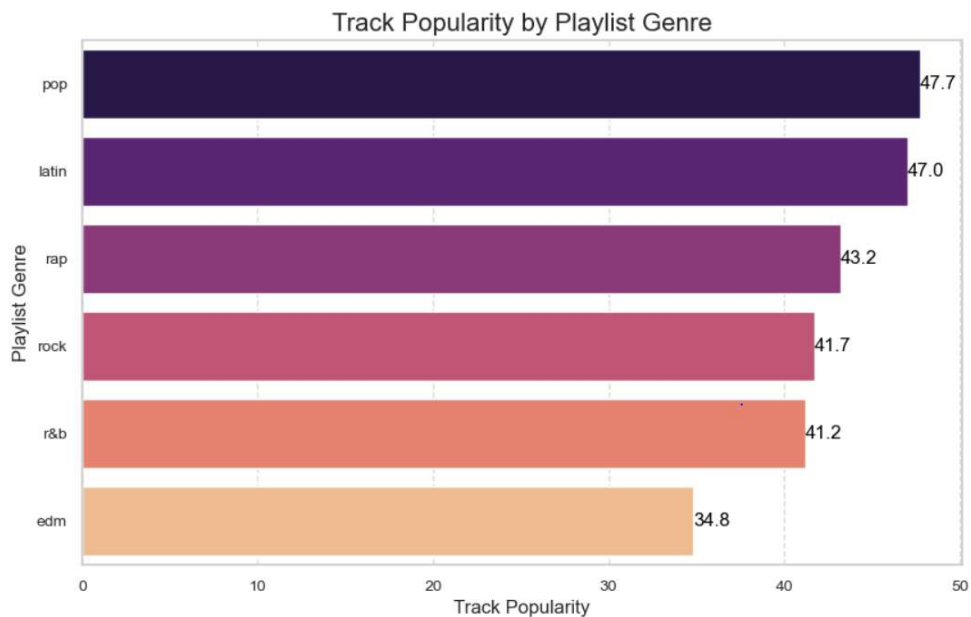


Fig 13: Track Popularity by Playlist Genre

According to this, **“pop”**, indeed as the name itself indicates, is the **most popular genre** with an average **“track popularity”** rating of 47.7.

Several other such visualizations have been obtained depicting the relations of some of the important features with each other.

However, there is no better way to check the visual variations of each of these relevant features with respect to each other than through the creation of a dashboard.

Such a dashboard has been created using Power BI.



# **CHAPTER 3: INTERACTIVE DASHBOARD USING POWER BI**

### 3. Interactive Dashboard using Power BI

An interactive dashboard offering interactions between different attributes of the dataset has been created using the **Power BI** tool.

This interactive dashboard, whose screenshot is being displayed here, displays the important attributes of the data as well as provides filters for them (in the top section) to enable interactions among various features.



Fig 14: Interactive Dashboard created using Power BI

### **3.1 Visualizations**

Various Visualizations of interactions between the attributes have been offered through this dashboard.

### **3.2 Design Choices**

Scatter plots have been created for attributes like danceability and energy because they reveal relationships between these metrics. Bar charts are effective for comparing values like popularity across artists or genres

### **3.3 Data Insights**

One interesting insight is that higher energy tracks tend to have higher popularity scores. Similarly, certain keys or tempos might be more associated with specific genres or moods.

Such insights can be useful for making music recommendations based on users' choices with much ease.

### **3.4 Significance of the Dashboard**

This dashboard offers a simple but lively tool for analysing and recommending music. It allows users to explore tracks based on their preferences and gain insights into what makes music enjoyable or popular.

## **CHAPTER 4: MODEL EXECUTION**

## 4. Model Execution

Once sufficient exploratory analysis has been done with the data, it is then subjected to the k-means clustering algorithm to make a music recommendation system.

### 4.1 Data Pre-Processing

To execute this algorithm, first the relevant features are segregated from the given data-frame and scaled using the Standard Scaler function.

The selected features exclude those having the character data-type as well as two other features, “mode” and “key” because of the kind of values they hold, as described earlier.

Accordingly, the 10 relevant features selected and the consequent scaling operation for them can be seen from the following code-snippet:

```
# Select the relevant features for clustering
features = ['danceability', 'energy', 'loudness', 'speechiness', 'acousticness',
            'instrumentalness', 'liveness', 'valence', 'tempo', 'duration_ms']

X = data[features]
```

```
# Scaling data
from sklearn.preprocessing import StandardScaler

# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Viewing
scaler
```

```
StandardScaler
```

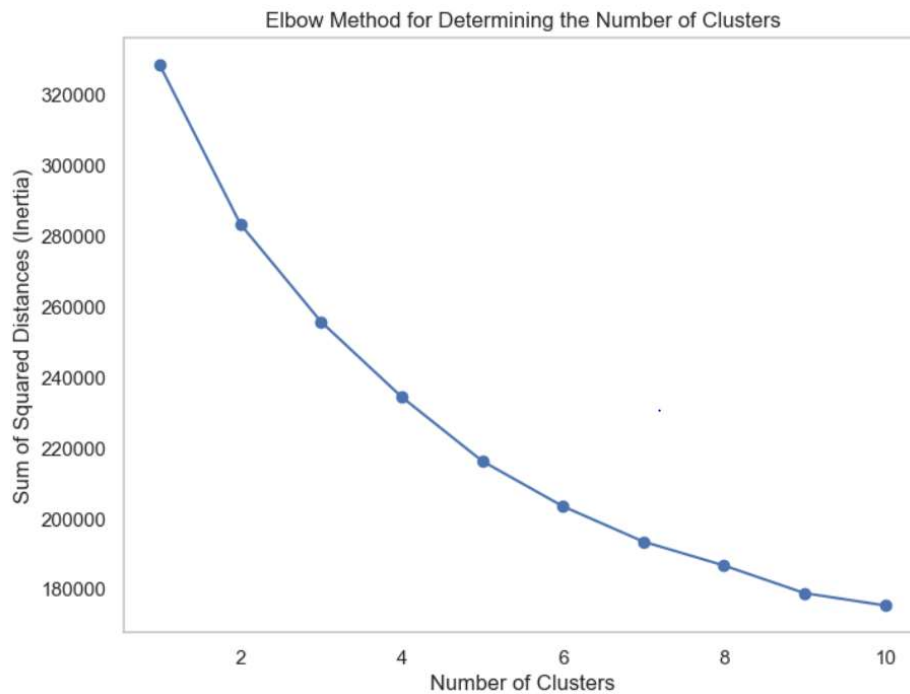
```
StandardScaler()
```

Fig 15: Relevant Music Features Scaled

## **4.2 Determination of the number of Clusters:**

For the k-means clustering algorithm to be executed, it needs to be provided with the number of clusters we seek to make from the given data.

To get the ideal number of clusters, the sum of squared distances method (known as “inertia”) is employed. The graph with the plot where this number is determined using the elbow-point of the graph is displayed below:



**Fig 16: Cluster-Count Determination by Elbow Method**

Accordingly, the ideal count of clusters for applying the k-means clustering algorithm is determined to be 4.

### 4.3 K-Means Clustering Algorithm

Once an ideal number of clusters has been identified, a Kmeans object (model) is created using  $n=4$  by running that algorithm on the given data.

The counts of music tracks (songs) in each of those 4 clusters are also determined, as displayed from the following code:

```
# The elbow point appears somewhere around 4 as the number of clusters
kmeans = KMeans(n_clusters=4, random_state=42)
data['cluster_kmeans'] = kmeans.fit_predict(X_scaled)
kmeans
```

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

```
# Viewing total cluster
data.cluster_kmeans.value_counts()
```

```
cluster_kmeans
0    13970
2    10837
1     5418
3     2608
Name: count, dtype: int64
```

Fig 17: KMeans Object and Cluster-component counts

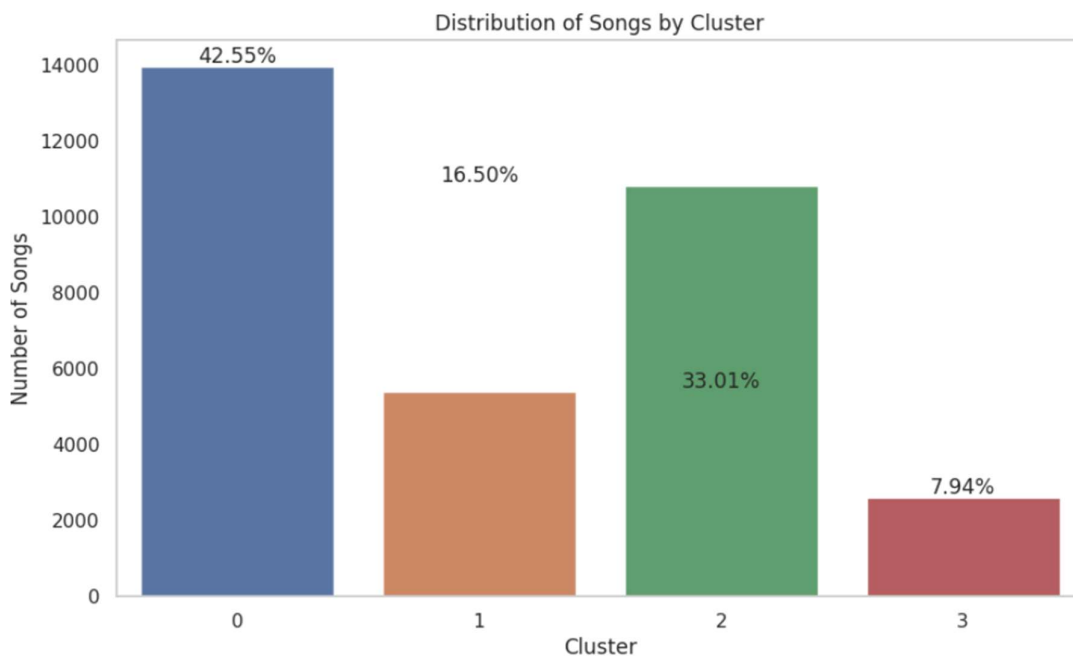


Fig 18: Distributions of Songs by Cluster

The distribution of those songs for each cluster, with both their counts as well as proportions in percentage terms is displayed in the following bar-plot.

## **4.4 Dimensionality Reduction using PCA**

As has been seen, a total of 10 features had been used for forming these clusters. However, the larger the feature-set, the more the complexity in the algorithm.

Since each feature represents a dimension in the n-dimensional feature-space, in order to reduce complexity in the adopted algorithm, the effective feature-set is decided to be reduced to a count of 2, using the method of Principal Component Analysis (PCA).

Reducing the Principal Components to a count of 2 also helps one visualize clusters on a 2-D plane.

So, not only a reduction in complexity is achieved by reducing dimensionality to a count of 2 through PCA but also a proper visualization of the clusters is enabled at the same time.

The following code has been used for applying the PCA on the scaled data attributes to reduce the dimensionality to a count of 2:

```
[58]: from sklearn.decomposition import PCA

# Perform PCA Analysis
# Initialize the PCA model specifying the number of components to reduce to (in this case, 2 components).
pca = PCA(n_components=2)

# Apply PCA on the scaled feature set X_scaled and transform the data into the new 2-dimensional space.
X_pca = pca.fit_transform(X_scaled)

# Output the PCA model object, which contains information such as the amount of variance explained by each principal component.
pca
```

[58]:

PCA

PCA(n\_components=2)

**Fig 19: PCA object-creation with n=2**



## 4.5 Cluster Visualization

The application of PCA to the feature-set in the problem, especially where the dimensions have been reduced to a count of 2, as stated before, offers the opportunity to visualize the obtained clusters in a 2D space. That is done through a pair-plot here.

Additionally, the cluster centres too have been plotted within those clusters, as depicted below.

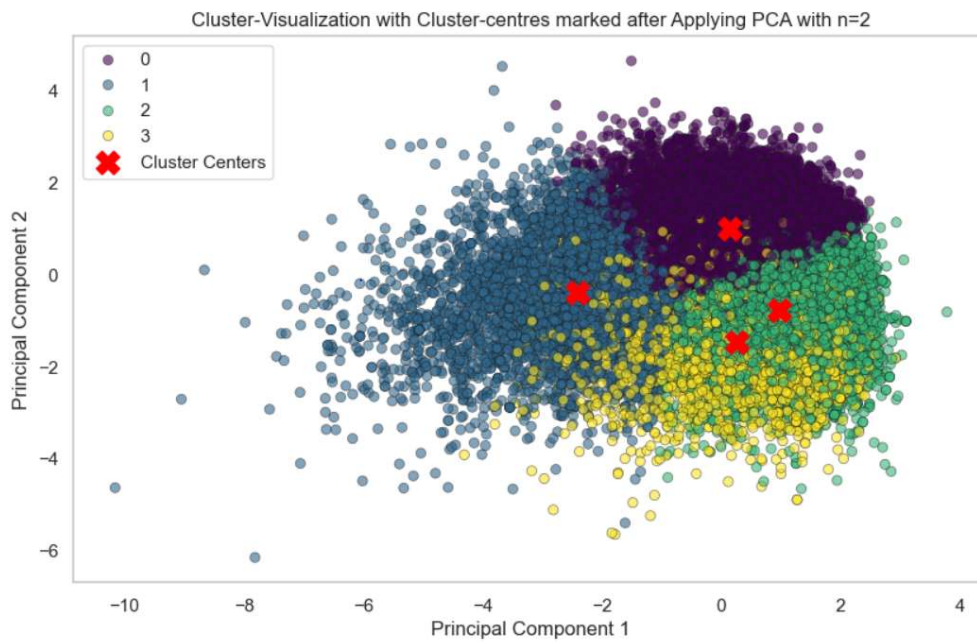


Fig 20: Cluster-Visualization with Cluster-Centres after PCA (n=2)

## 4.6 Cluster-based Recommendation System

Once the clusters have been formed and visualized, they are put into use by the algorithm to recommend songs/music-tracks after seeking an input by the user.

A total of 10 tracks have been designed to be recommended based on the user's input.

This is displayed by the code-output below:

```
1 # Example of usage
2 # Replace with the name of the song you want to use as the base for recommendations
3 song_name = str(input("Enter a Song-name: ")) #Memories - Dillon Francis Remix
4 top_recommendations_kmeans = recommend_songs_by_cluster_kmeans(song_name, data)
5
6 # If recommendations are found, print them
7 if top_recommendations_kmeans is not None:
8     print("Recommended music")
9     print(top_recommendations_kmeans)
```

Enter a Song-name: Memories - Dillon Francis Remix  
Recommended music

	track_name	track_artist
0	I Don't Care (with Justin Bieber) - Loud Luxur...	Ed Sheeran
2	All the Time - Don Diablo Remix	Zara Larsson
4	Someone You Loved - Future Humans Remix	Lewis Capaldi
5	Beautiful People (feat. Khalid) - Jack Wins Remix	Ed Sheeran
9	If I Can't Have You - Gryffin Remix	Shawn Mendes
10	Cross Me (feat. Chance the Rapper & PnB Rock) ...	Ed Sheeran
12	Body On My	Loud Luxury
15	South of the Border (feat. Camila Cabello & Ca...	Ed Sheeran
17	Say My Name (feat. Bebe Rexha & J Balvin) - Lu...	David Guetta
21	All Around The World (La La La) - Marnik Remix	R3HAB

Fig 21: Output of 10 Recommended Songs upon entry of an Input

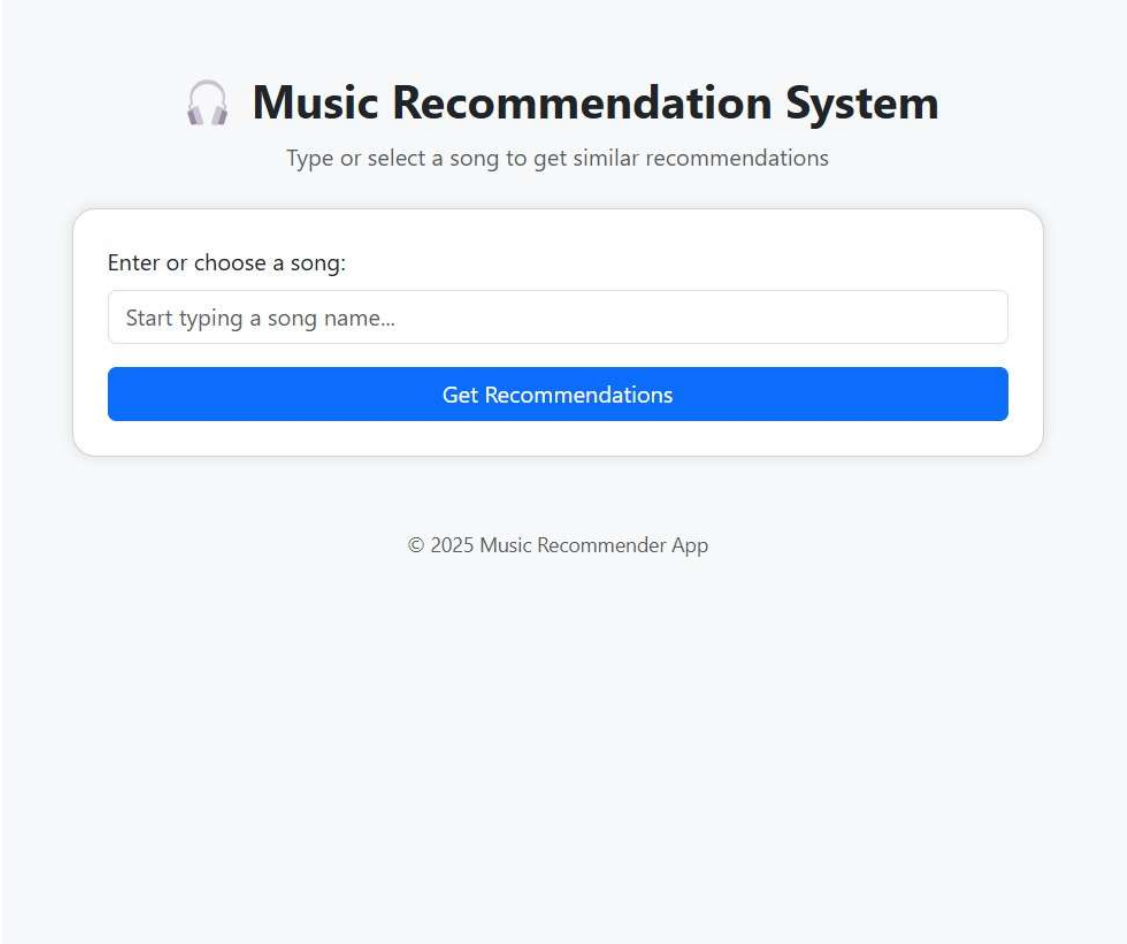
## **4.7 UI/Web-Page Integration with Flask-Framework**


The working code is then integrated with the Flask web-framework to produce an HTML page for interaction with the user.

A field for entering a song input is created. Once a user enters a song's name (music-track's title) into the given field, an output is generated with 10 recommendations based on that input.

This is displayed in the UI interface displayed below:

This is displayed in the code output given below. It may be noted that, for reference, the same input as in the previously displayed code-snippet is being entered with the same output obtained.

The image shows a web interface for a "Music Recommendation System". At the top, there is a headphones icon followed by the title "Music Recommendation System" in a bold, dark font. Below the title is a subtitle: "Type or select a song to get similar recommendations". The main content area is a white rounded rectangle with a subtle shadow. Inside this rectangle, the text "Enter or choose a song:" is positioned above a text input field. The input field has a light gray border and contains the placeholder text "Start typing a song name...". Below the input field is a solid blue button with the text "Get Recommendations" in white. At the bottom of the white container, there is a small copyright notice: "© 2025 Music Recommender App". The entire interface is set against a light blue background.

 **Music Recommendation System**

Type or select a song to get similar recommendations

Enter or choose a song:

Start typing a song name...

Get Recommendations


© 2025 Music Recommender App

Fig 22: UI/HTML Page for the user to get recommended songs

A dropdown is made for both purposes. It helps in searching by

- Typing, as well as by
- searching within the dropdown.

By selecting song, it helps the user to navigate around the given songs and select their favourite one to get more songs recommended from the system.



# Music Recommendation System

Type or select a song to get similar recommendations

Enter or choose a song:

Get Recommendations

© 2025 Music Recommender App

- I Don't Care (with Justin Biebe...
- Memories - Dillon Francis Remix
- All the Time - Don Diablo Remix
- Call You Mine - Keanu Silva Re...
- Someone You Loved - Future...
- Beautiful People (feat. Khalid)...
- Never Really Over - R3HAB Re...
- Post Malone (feat. RANI) - GA...
- Tough Love - Tiësto Remix / R...
- If I Can't Have You - Gryffin Re...
- Cross Me (feat. Chance the Ra...
- Hate Me - R3HAB Remix
- Body On My
- SOS - Laidback Luke Tribute R...
- Summer Days (feat. Macklemo...
- South of the Border (feat. Cam...
- All My Friends - Eden Prince R...
- Say My Name (feat. Bebe Rexh...
- Dancing With A Stranger (Wit...

Fig 23: Drop-Down List for Input Songs

It also provides a typing feature to enter song name by the user entirely. The availability/existence of that song in the database is then checked.

Thereafter, by clicking on the “Get Recommendations” button, the recommended songs are generated.



# Music Recommendation System

Type or select a song to get similar recommendations

Enter or choose a song:

Get Recommendations

© 2025 Music Recommender App

Fig 24: UI feature to Input song-name by typing

Should an entered song-name be incorrect or unavailable in the database, the UI gives out a message stating declaring that the given song wasn't found.

Accordingly, no recommendations are made.


The screenshot displays the 'Music Recommendation System' interface. At the top, there is a header with a headphones icon and the title 'Music Recommendation System'. Below the header, a subtitle reads 'Type or select a song to get similar recommendations'. The main input area contains a text field with the label 'Enter or choose a song:' and the text 'diwani' entered. A blue button labeled 'Get Recommendations' is positioned below the text field. Below the button, a message box with a yellow warning icon states: '⚠️ No similar songs found in the same cluster for "diwani"'. At the bottom of the interface, the copyright notice '© 2025 Music Recommender App' is visible.

Fig 25: UI output if song-name not present in the database

Once a user enters a correct song-name (music-track's title) into the given field, an output is generated with 10 recommendations based on that input.

This is displayed in the code output given below. It may be noted that, for reference, the same input as in the previously displayed code-

snippet (Pg 26) is being entered with the same output obtained.




## Music Recommendation System

Type or select a song to get similar recommendations

Enter or choose a song:

Memories - Dillon Francis Remix

Get Recommendations

 Recommended Songs:

Track Name	Artist
I Don't Care (with Justin Bieber) - Loud Luxury Remix	Ed Sheeran
All the Time - Don Diablo Remix	Zara Larsson
Someone You Loved - Future Humans Remix	Lewis Capaldi
Beautiful People (feat. Khalid) - Jack Wins Remix	Ed Sheeran
If I Can't Have You - Gryffin Remix	Shawn Mendes
Cross Me (feat. Chance the Rapper & PnB Rock) - M-22 Remix	Ed Sheeran
Body On My	Loud Luxury
South of the Border (feat. Camila Cabello & Cardi B) - Andy Jarvis Remix	Ed Sheeran
Say My Name (feat. Bebe Rexha & J Balvin) - Lucas & Steve Remix	David Guetta
All Around The World (La La La) - Marnik Remix	R3HAB

Fig 26: Output of the Song Recommendation System in the UI



## **CHAPTER 5: MODEL WORKFLOW AND OUTPUT**

## **5. Model Workflow and Output**

At the execution stage, the following flow of work is expected leading to the eventual output of song recommendations:

### **1. User Input Collection**

An input in the form of the music-track's title is entered by the user in the given field in an HTML page,

### **2. Data Processing**

The entered music track's title is matched with the database used and checked for its validity,

### **3. Prediction Generation/Output**

Once verified, the Recommendation Algorithm in the background offers 10 music-tracks as recommendations for the user based on the features of the input music-track.

The above workflow has been observed to successfully run through, and hence the model execution leads to a successful implementation and output.

## **CHAPTER 6: PROJECT REQUIREMENTS**

## 6. Project Requirements

This **Music Recommender System** project was developed using various tools that enabled in analysing the data followed by building and deploying the model. A detailed description of the tools and technologies used are given below:

### **Software Used:**

1. Python: Version 3.12.7

### **IDEs used:**

1. Jupyter Notebook
2. PyCharm (Community Version)
3. Visual Studio Code

### **Libraries Used:**

1. Sci-kit learn (sklearn): Version 1.5.1
2. Seaborn: Version 0.13.2
3. Matplotlib: 3.9.2
4. Pandas: 2.2.2
5. Numpy: 1.26.4
6. Plotly: 5.24.1

### **Web-Framework Used:**

1. Flask

### **Dashboard Used:**

1. Power BI

## **CHAPTER 7:**

# **CONCLUSION AND FUTURE SCOPE**

## **7. Conclusion and Future Scope**

The given Spotify songs' dataset has been put through the k-means clustering algorithm. The number of such clusters to be formed was determined based on the similarity (nearness) of the song features based on the sum of squares distances, also known as “inertia”, of each of those relevant features from each other.

The number of such clusters were determined to be 4. Exploration was done by keeping such clusters at a count of 3 and 5 as well.

However, 4 was found to be the most suitable cluster count which has matched the observation of the elbow point too in the relevant plotted graph. The given system, hence, provides a fairly reliable model for music recommendation based on similarity of song-features.

There is scope, however, of exploring further means of recommendation systems like those through collaborative filtering and content-based filtering techniques.

Further, deep learning methods like the Restricted Boltzmann Machines too are used to achieve robust recommender systems.

Exploration of those techniques going forward could further enhance the ease and accuracy of the recommender system algorithms.

## **CHAPTER 8: REFERENCES**

## 8. References

1. The Dataset Source on Kaggle, titled, “30000 Spotify Songs” at the URL:

<https://www.kaggle.com/datasets/joebeachcapital/30000-spotify-songs/data>

2. Python Documentation:

<https://docs.python.org/3.12/>

3. Scikitlearn Library Documentation

<https://scikit-learn.org/0.21/documentation.html>

4. Pandas Documentation

<https://pandas.pydata.org/docs/>

5. NumPy Documentation

<https://numpy.org/doc/>

6. Flask Documentation

<https://python-adv-web-apps.readthedocs.io/en/latest/flask.html>