A Project report on

MUSIC RECOMMENDER SYSTEM USING K-MEANS CLUSTERING

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

in

Computer Science and Engineering (AI&ML)

Submitted by

BUDDIGA INDIRA SAI
21H55A6602

MODUGU CHARITHA
20H51A6638

VARDHINENI AKSHITH RAO
20H51A6666

Under the esteemed guidance of

G. SREENIVAS

ASST.PROFESSOR-CSE(AI&ML)

Department of Computer Science and Engineering (AI&ML)

CMR COLLEGE OF ENGINEERING&TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2020- 2024

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (AI&ML)

CERTIFICATE

This is to certify that the mini Project 1 report entitled **"MUSIC RECOMMENDER SYSTEM USING K-MEANS CLUSTERING"** being submitted by B.INDIRA SAI (21H55A6602), M. CHARITHA (20H51A6638), V. AKSHITH RAO (20H51A6666) in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering(AI&ML)** is a record of bonafide work carried out his/her under my guidance and supervision. The results embodies in this project report have not been submitted to any other University or Institute for the award of any Degree.

G.Sreenivas
Associate Professor
Dept. of CSE(AI&ML)

Dr. P. Sruthi
Associate Professor and HOD
Dept. of CSE(AI&ML)

ACKNOWLEDGEMENT

With great pleasure we want to take this opportunity to express my heartfelt gratitude to all the people who helped in making this project work a grand success.

We are grateful to **G. Sreenivas,** Asst.Professor-CSE(AI&ML), Department of Computer Science and Engineering (AI&ML) for his valuable technical suggestions and guidance during the execution of this project work.

We would like to thank **Dr. P. Sruthi,** Head of the Department of Computer Science and Engineering (AI&ML), CMR College of Engineering and Technology, who is the major driving forces to complete my project work successfully.

We are very grateful to **Dr. Vijaya Kumar Koppula**, Dean-Academic, CMR College of Engineering and Technology, for his constant support and motivation in carrying out the project work successfully.

We are highly indebted to **Dr. V A Narayana,** Principal, CMR College of Engineering and Technology, for giving permission to carry out this project in a successful and fruitful way.

We would like to thank the Teaching & Non- teaching staff of Department of Computer Science and Engineering (AI&ML) for their co-operation

We express our sincere thanks to **Mr. Ch. Gopal Reddy**, Secretary, CMR Group of Institutions, for his continuous care.

Finally, We extend thanks to our parents who stood behind us at different stages of this Project. We sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

B. Indira Sai 21H55A6602
M. Charitha 20H51A6638
V. Akshit Rao 20H51A6666

TABLE OF CONTENTS

List of Figures

ABSTRACT

In this paper, We presented a personalized MUSIC RECOMMENDER SYSTEM based on K-MEANS CLUSTERING Algorithm. In personalized Music Recommender System, We proposed a Content Filtering Recommender Algorithm to combine the output of the network with the log files to Recommend Music to the user. The proposed system contains the log files which stores the previous history of playlist of music heard by the user. The proposed Music Recommender System extracts the users history from the log file and recommends music to the user under each recommendation . Content-based methods gives recommendations based on the similarity of songs played, contents or attributes while collaborative methods make a prediction on possible preferences using a matrix with ratings on different songs. The system extracts the music from input and finds the music that are close to the query music which the query has played previously. We use millions of song datasets to evaluate the personalized Music Recommender System. The data cleaning is done by the data science algorithms. The detection of songs is done by finding the similar music genre. In our project, we will be using a sample data set of songs to find correlations between users and songs so that a new song will be recommended to them based on their previous history. We will implement this project using libraries like NumPy, Pandas. We will also be using Cosine similarity

Keywords : Numpy, Pandas, Cosine Similarity , PCA, Clustering

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

With the explosion of network in the past decades, Internet has became the major source of retrieving multimedia information such as video, books, and music etc. People have considered that music is an important aspect of their lives and t hey listen to music. However, The problem now is to organize and manage the millions of music titles produced by society. A good music recommender system should be able to automatically detect preferences and generate playlists according to the user. The proposed system is to detect music based on music similarity. The system extracts the music from input and finds the music that are close to the query. Meanwhile, The development of recommender systems provides a great opportunity for industry to aggregate the users who are interested in music. We need to generate the best music recommender system which is needed to predict which are based on customization, by using K-means clustering, Machine Learning. Everyone has different taste in music is unique which means that no matter what music you make, someone is bound to enjoy listening to it. While the Music industry may favour certain types of music more than others. Music is of great benefit to the users, regardless of whether we are renowned recording artists, merely fans of music. The number of songs available exceeds the listening capacity of single individual. There are at nearly 97 millions of songs. These are only the songs officially released. Starting there, let's say that there are currently around 1 million song writers alive that we know about. In total, there must be billions just there and Spotify itself is by no means. The limit of music. What about all the CDs and records made over the past century which have not been digitized. There are trillions and trillions of songs in the world, So many that an estimate is impossible, and the potential more an infinitely greater number which have not yet been made, A world of music is for us to enjoy. Currently, based on users listening behaviour and historical ratings, Content based filtering algorithm has been found to perform well. In content-based method similarity between the songs are checked and songs are recommended based on the similarity score. The content -based filtering method can also be considered as per detection. They keep the user engaged by finding interesting music in the form of recommendations. They give the scope for exploration and discovery of music that the user may not know exists. Because it is a music recommender there is never less entertainment.

1.1 LITERATURE REVIEW

An ideal music recommender system should be able to automatically recommend personalised music to human listeners. So far, many music discovery websites such as Last.fm, All music, Pandora, Audio baba Mog, Spotify, Apple Genius, have aggregated millions of users, and the development is more explosive. Here, we present the most popular approaches, metadata information retrieval, content-based information retrieval, emotion-based model , context-based information retrieval and hybrid models . The proposed system contains the recommendation system which acts as a great advantage to resolve. The module deals the check of similar music genre and detects the songs that are similar to musical notes that means which are played previously. The content-based filtering technique is used to recommend songs to the users of similar groups. Involving content based filtering technique is an advantage to recommend songs by achieving the customization. The three types of collaborative filtering involved are memory based collaborative filtering, model based filtering, hybrid based filtering. The existing recommender systems using collaborative filtering algorithms have gained a great success, which greatly contributes to the success of the business. Recently a newer algorithm using neural network, neural collaborative filtering was proposed. The content-based filtering technique is used to predict the song by analyzing the song track. It is rooted in information retrieval and information filtering that recommends a song which is similar to those the user has listened to in the past rather than what the user have rated ‘like’ Lots of research have been paid attention one extracting and comparing the features in finding perceptual similar tracks. Based on the extracted features, The distance between songs are measured. Three typical similarity measurements are K-means clustering, Expectation-Maximization Sampling, Average Feature Vectors with Euclidean Distance. For content based algorithm, a lot of researchers have proposed different methods using Machine Learning technique, such as Decision Tree based , Support Vector Machine based and even logistic regression . We can fully utilize the knowledge we learnt from the class to implement these algorithms. Digitization of music has led to easier access to different forms music across the globe. Increasing work pressure denies the necessary time to listen and evaluate music for a creation of a personal music library. One solution might be developing music search engine or recommendation system based on different moods.

1.2 PROBLEM STATEMENT

The main aim of music recommendation system using k-means clustering is to generate the best music recommendation system by predicting based on previous listening history of user by using Content based filtering, Machine Learning, Data Analysis. The basic task in music recommendation system is to generate the best music recommendation system by predicting based on customization and detecting the similar music genre.

1.3 RESEARCH OBJECTIVE

Different recommendations primarily need to work for the satisfication of the users. Identifying user grievances thereby resolving them leads to customer satisfaction as well as trustworthiness. The today's world many people are busy and suffering a lot in their life so to overcome that problem for atleast sometime and the best solution is listening to the music. So we decided the project on a music recommendation system so that users can listen a song based on their interests, can get the recommendation based on their previous listening history.

1.4 PROJECT SCOPE AND LIMITATIONS

In the future, we would like to try the following things:

1.Using audio signal (e.g. audio frequency) to recommend songs

2.Trying Convolutional Neural Network

3.Making the recommender system a real-time

Designing a personalized music recommender is complicated. The future research direction will be mainly focused on user-centric music recommender systems Therefore, future music recommenders should be able to lead the users to reasonably choose music. In the end, we are hoping that through this study, we can build the bridge among isolated research in all the other disciplines. It's really hard for people to come up with a good heuristic which actually reflects what we want the algorithm to do. It might not find the most optimal solution to the defined problem in all cases.

LIMITATIONS

1. It can play songs in real life
2. It has only English song datasets

CHAPTER 2

BACKGROUND WORK

2.1 EXISTING SYSTEM METHOD

RESEARCH ON MUSIC CONTENT AND RECOMMENDATION TECHNOLOGY BASED ON DEEP LEARNING

Collaboration Filtering (CF) Uses the numerical reviews given by the user and is mainly based upon the historical data of the user available to the system. The historical data available helps to build the user profile and the data available about the item is used to make the item profile. Both the user profile and the item profile are used to make a recommendation system. Collaborative filtering is considered the most basic and the easiest method to find recommendations and make predictions regarding the sales of a product. It does have some disadvantages which has led to the development of new methods and techniques.

Memory-Based Collaborative Filtering (Neighbourhood based) People with similar interests are combined to form a group and every user is a part of that . User –based Collaborative Filtering and Item implement and scales well with correlated items. There is no need items being recommended. There are many limitations of memory problem.

Model-based Collaborative Filtering Complex patters which are based on training data, are the models (such as data mining algorithms, machine learning) and then intelligent predictions are made for Collaborative Filtering tasks for the real world data which are based on learnt models. It intuitive rationale for recommendations. Model disadvantage of model-based collaborative Filtering is that it loses useful information for dimensionality reduction techniques.

Content Based Recommender System This focuses on the features of the products and creating a user profile depending on the previous reviews and also a profile of the item in accordance with the features it provides and the reviews it has received. It is observed that reviews usually contain product feature and user opinion in pairs . It is observed that users reviews contain a feature of the product followed by his/her opinion about the product.

Context Based Recommender System Extending the user/item convention to the circumstances of the user to incorporate the contextual information is what is achieved in context-based recommender systems. This helps to abandon the process of making the user fill a huge number of personal details. It aids in extracting information about a particular community of individual sand this information proves to be of high importance to improve the suggestions provided to a user and makes the system more efficient. Recommender systems require situational information of the user and context based recommender system accesses this information directly using various techniques (such as GPS) and does not bother the user with this. The user's location data, social data, current time, weather data is taken into consideration as the contextual data and is given as input to the system. An approximate address of the user is determined and the location is saved. Social data of a user can be accessed by requesting permission to a social account of the user. Contextual factors are of two types: Dynamic and Static, depending on whether they change with time or not. 1) Dynamic: When the contextual factors change over time and hence unstable. They may change by explicit user feedback. User feedback is generally used for refining the profile of user to get better results of recommendations. The biggest challenge is that if a system is considered to be dynamic then the system should be able to find out when to switch to a different underlying context model. 2) Static: The contextual factors don't change over time and hence stable. For example., To buy a cell phone the contextual factors can be Time, purpose of purchasing and only them while entire purchasing purpose recommendation application runs. Contextual factors are of three types: Fully observable, Partially observable and Unobservable, depending on what is being observed. 1) Fully observable: Complete structure and values of contextual factors are known explicitly, at the time when recommendations are made. 2)Partially observable: Some of the information is known explicitly about the contextual factors. 3) Unobservable: There is no information of contextual factors explicitly available in it.

Recommender System Based on Genetic Algorithm The proposed system aims to effectively adapt and respond to immediate changes in users' preferences. The experiments conducted in an objective manner exhibit that our system is able to recommend items suitable with the subjective favourite of each individual user. The content-based filtering technique is applied to generate the initial population of GA. The recommender system is divided into three phases:

1)Feature extraction 2)Feature evaluation 3)Interactive phase

2.1.1 INTRODUCTION

This research aims to create a better music algorithm that incorporates user data for deep learning, A candidate matrix compression technique for suggestion improvement, accuracy, recall rate, and other metrics as evaluation criteria. In terms of recommendation methods. The music recommendation method based on predicting user behaviour data and the recommendation method based on automatic tag generation are proposed. The music features obtained by audio processing are fully utilized, and the depth content information in music audio data is combined with other data for recommendation, which improves the tag quality and avoids the problem of low coverage. The results show that this model can extract the effective feature representation of songs in different classification criteria and achieve a good classification effect simultaneously. The resultant recommendation engine has evolved into a link between users' wants and material, allowing users to not only locate possible content they are interested in, but also better present unpopular content and discover new people. The use of recommendation system technology to music is a significant one. The goal of this study is to develop and construct a music feature extraction strategy that may be used in music recommendation contexts. With the training set of music bottom feature set and deep confidence network, a music information prediction model is created. The article-side automated encoder learns audio features using the convolution layer and lyrics features using the full connection layer. The user-side automatic encoder learns the user item score vector using the full connection layer. After pretraining, combine matrix decomposition to train the tight coupling model. Combining the characteristics of melody and sound quality, and according to the scenes with large amount of data, the music features can be obtained efficiently, which complements the resource data.

2.1.2 MERITS, DEMERITS AND CHALLENGES

MERITS

1. Investigates the recommendation technique
2. Audio processing technology
3. Combining the address of some of the problems of the present music platform's recommendation system.
4. The recommendation system make diverse recommendations for users so that users can have different experiences

DEMERITS:

1. It does not support retrieval and playback.
2. It is very complicated
3. The algorithm treats all items as equal in the calculation process, without considering the importance of items.

CHALLENGES:

1. Overcoming the cold start problem
2. Automatic playlist continuation
3. Evaluating music recommender systems
4. Review the state of the art of the respective tasks

2.1.3 IMPLEMENTATION OF MUSIC RECOMMENDER SYSTEM

At present, the recommendation based on deep learning overcomes the obstacles of traditional linear model, thus significantly improving the recommendation quality. Deep learning can effectively capture the nonlinear relationship between users and items and obtain the vector representation of users or items by vectorization or coding. In the general training process, the process of deep learning is usually divided into supervised learning and unsupervised learning. The recommendation engine contains a recommendation algorithm and a recommendation rationale that will construct a link between the user characteristics and the things to be suggested to propose the target items of interest to users based on the established link. The recommendation engine comprehensively calculates the information of the user's education, age, label, and gene description of the item to be recommended and then combines the user's preference for the item: depending on the item itself, it may include the user's rating of the item, the user's click record, etc. and finally forms the recommendation result.

Self-encoder is an important basic model of deep learning. Its basic realization process encodes the input information, extracts the main information, and reproduces the input signal as much as possible. Then, in the concrete realization process, the system needs to be able to capture the main factors that can represent the input information. The general structure includes the visible, hidden, and encoder and decoder between them [23]. The basic idea of content-based recommendation algorithm is to analyze the characteristics of users' preference behaviors according to the historical information of users, get user preference sets, and match these sets with the recommended content to achieve recommendation. Commonly used music recommendation algorithms include the recommendation algorithm based on labeled content and music based on music features.

The traditional content-based recommendation algorithm first gets the items that have interacted with users and then gets the user's preference model through similar user's active behaviors such as users' likes or ratings of items. User feedback on the project includes implicit and explicit ones. The former is the log of users' marks when using the system, which reflects users' interest in the project, such as browsing the lyrics of songs and the number of times the songs are played repeatedly. The latter is the user's display of embodied content in addition to ordinary browsing on the system. The flowchart of music content identification and recommendation based on deep learning is shown in Figure.

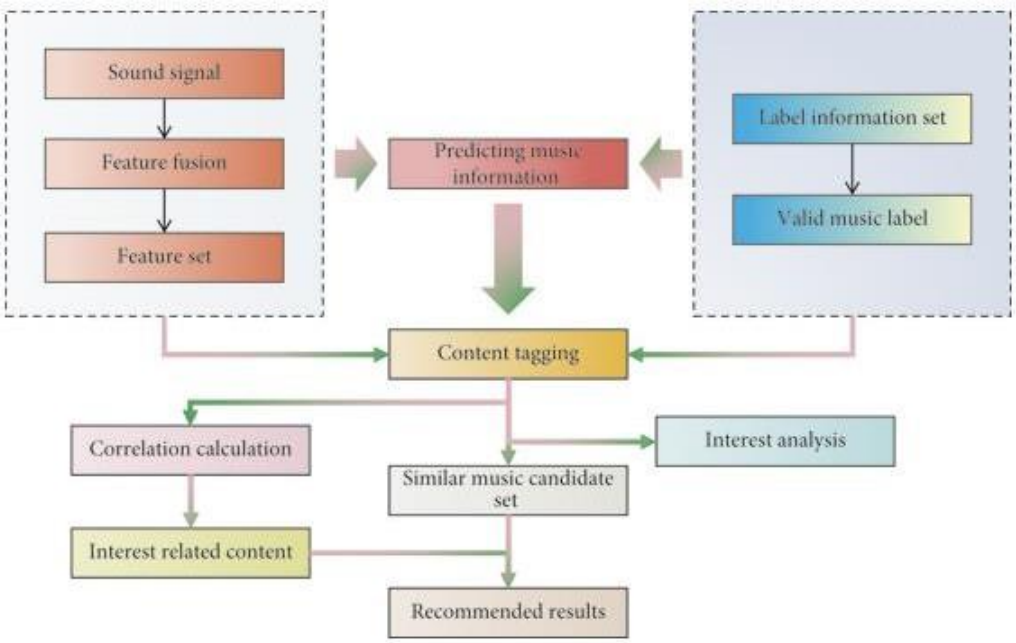


Fig-1 Personalized music recommendation process based on deep learning

To make the recommendation of songs more in line with the user's taste, it is necessary to extract the spectrogram of songs and the audio sequences contained in them and then classify them. The traditional recommendation algorithm classifies songs after processing pictures and audio sequence information, and there is a technical bottleneck in integrating the classified data into the recommendation model. How to make good use of these picture data and audio time series plays an important role in the generation of recommendation results. The picture data and audio time series can be processed by neural network recommendation algorithm, to classify songs, and at the same time, the classified data can be integrated into the recommendation model to achieve more accurate recommendation results.

In the field of content-based music recommendation, this means that our classifier has learned better music feature representation. To recommend suitable songs to users, we also need to know users' preferences, how to rate users or call them user portraits, and how to match users and songs. Similarity or other association rules can be used. From these data, users' labels can be obtained by direct analysis or mathematical modeling, and machine learning methods can also obtain quantitative indicators or abstract features of users.

The data set provides the playing information of songs. Obviously, we can know the popularity of songs through the playing information. The more times a song is clicked, the higher its popularity. This is an important feature in the recommendation scenario of non-cold start problem. In the cold start scene, the popularity of new music is naturally zero. The number of iterations should be kept to a minimum in practice to prevent wasting time and producing inefficiencies.

The highly relevant music is recommended to users from the music library through music information prediction and user interest degree calculation, based on the corresponding relationship between music information such as music tags and music tags based on users' interests, to meet the personalized needs of users. The suggestion of new users primarily assumes that new users like a certain genre and then picks 100 pieces of music from a variety of genres at random as the user's favorite music. Recommend an additional 500 pieces of music by using the recommendation model and analyze the distribution of the recommended music styles to see if they are consistent with their favorite styles.

Therefore, we adopt the method of supervised learning and use the idea of classification to deal with the problem of music label generation. Take a group of music with clear labels as a training set, their musical characteristics as input, and the probability of containing a label as output and then predict the labels of other music after training. In the content-based recommendation system, another intuitive recommendation method can be adopted. If the characteristics of users and items are obtained at the same time, we can match similar users and items to achieve the purpose of recommendation. In this way, the effectiveness of user characteristics is particularly important, and the behavior of mining user characteristics has a special name, called user portrait. Users always prefer listening to music, either for the genre or for a certain singer. Therefore, counting the recommendation list can directly observe whether the recommended results have practical significance.

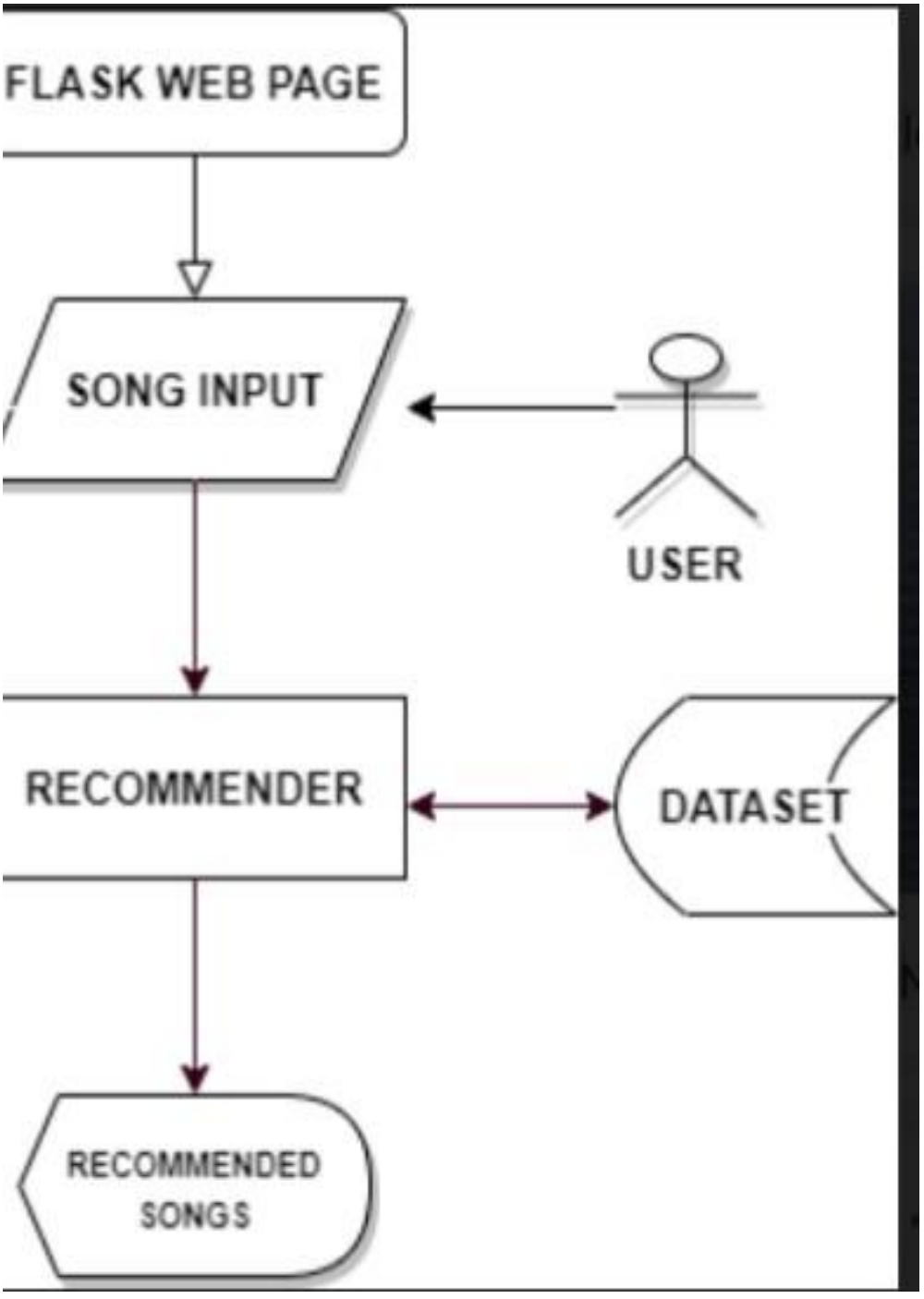
CHAPTER 3
SYSTEM DESIGN

CHAPTER 3
SYSTEM DESIGNS

3.1 PROPOSED SYSTEM INTRODUCTION

A good music recommender system should be able to automatically detect preferences and generate playlists accordingly. The development of recommender systems provides a great opportunity for industry to aggregate the users who are interested in music. We need to generate the best music recommendation system which is need to predict based on similarity score, by using content based filtering and Machine Learning. Most of the recommender systems use collaborative filtering. The existing recommender systems using collaborative filtering algorithms have gained a great success. In contrast, we are using a content based filtering. Content based systems focus on the features of the products and aim at creating a user profile depending on the previous reviews and also a profile of the item in accordance with the features it provides and the reviews it has received .It is observed that reviews usually contain product feature and user opinion in pairs . It is observed that users' reviews contain a feature of the product followed by his/her opinion about the product. Content based recommendation systems help overcome sparsity problem that is faced in collaborative filtering-based recommendation system. Content based collaborative filtering is more widely used to compare pure CF and pure Content-base. In CF the problem of sparsity is overcome (converting sparse user filled matrix into full user rating matrix) by using content-based prediction. Relevant entities of an item and relations are kept together as input. The content-based filtering technique is used to predict the song by analyzing the song track. It is rooted in information retrieval and information filtering that recommends a song which is like those the user has listened to in the past rather than what the user have rated 'like' Lots of research have been paid attention on extracting and comparing the acoustic features in finding perceptual similar tracks. The most representative ones so far are timbre, rhythm. Based on the extracted features, the distance between songs is measured. Three typical similarity measurements are K-means clustering with Earth-Mover's Distance, Expectation-Maximization with Monte Carlo Sampling, Average Feature Vectors with Euclidean Distance.

3.2 PROPOSED SYSTEM ARCHITECTURE



Graph between artist name and listen

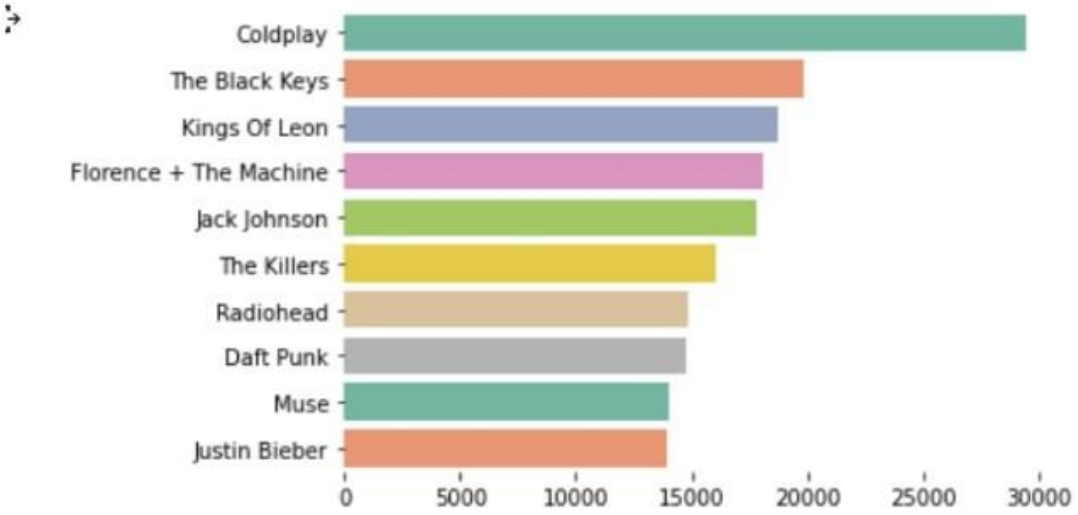


Fig-2

Graph between title and listen count

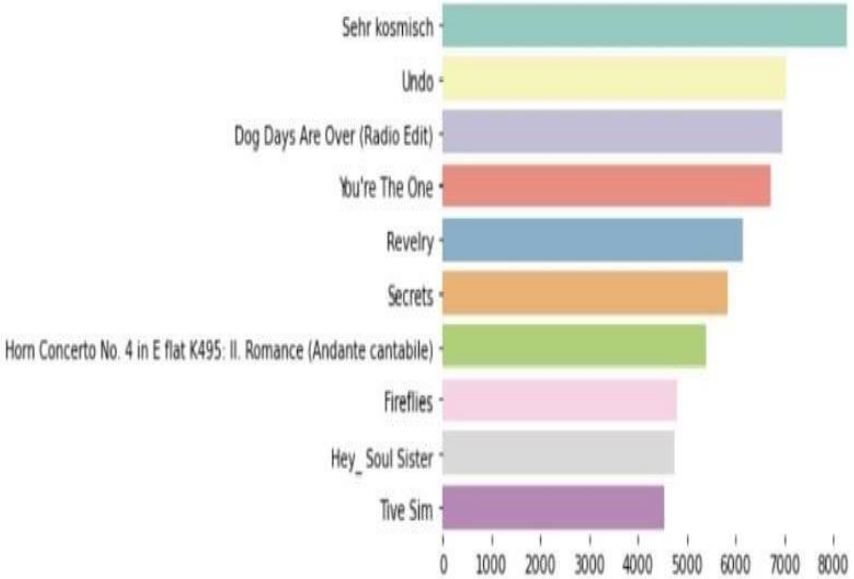


Fig-3

3.2 ADVANTAGES OF PROPOSED METHODS

ADVANTAGES:

1. It is optimal
2. It determines the songs precisely
3. Compared to the other recommender systems, It gives more accurate recommendations
4. Without any derivations it is completely dependent on the content and music of the songs

3.3 IMPLEMENTATION OF PROPOSED METHODS

The approach for the proposed system is in following steps:

Data analyzing: Data analysis is defined as a process of cleaning, transforming, and modeling data to discover useful information for business decision-making. The purpose of Data Analysis is to extract useful information from data and taking the decision based upon the data analysis. So here, from the datasets collected we are trying to extract the useful information.

Data visualization and EDA: Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

Exploratory Data Analysis - EDA is applied to investigate the data and summarize the key insights. It will give you the basic understanding of your data, it's distribution, null values and much more. You can either explore data using graphs or through some python functions. Here, we are applying these for better understanding of the data.

Data pre-processing: Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. We are pre-processing the data using StandardScaler().

StandardScaler() is used for standardization of data. Standardization means converting data into standard format.

$$z = \frac{x - \mu}{\sigma}$$

Principal component analysis (PCA): The Principal Component Analysis is a popular unsupervised learning technique for reducing the dimensionality of data. It increases interpretability yet, at the same time, it minimizes information loss. It helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D. PCA helps in finding a sequence of linear combinations of variables. It is a feature extraction technique, so it contains the important variables and drops the least important variable.

Clustering with K-means: K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabelled dataset into different clusters. It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabelled dataset on its own without the need for any training. Here, The simple K-means clustering algorithm is used to divide the genres in this dataset into ten clusters based on the numerical audio features of each genres.

Cosine similarity: For our song recommendation system, we are going to use cosine similarity and particularly, its implementation from Scikit-learn. We want to calculate the cosine similarity of each item with every other item in the dataset. So we just pass the lyrics_matrix as argument. Once we get the similarities, we'll store in a dictionary called similarities, the names of the 50 most similar songs for each song in our dataset. Cosine similarity is a metric used to measure how similar two items are. Mathematically, it measures the cosine of the angle between the two vectors projected into a multi-dimensional space. The output value ranges from 0-1.0 means no similarity where as 1 means that both the items are 100% similar.

The python cosine similarity as the dot product of the input samples. Given two vectors of attributes, A and B, the cosine similarity, cos(product and magnitude). Basically, the cosine similarity is the dot product of two vectors divided by the product of the magnitude of each vector. We divide the dot product by the magnitude because we are measuring only angle difference. On the other hand, dot product is taking the angle difference and magnitude into account. If we divide the dot product by the product of each vectors magnitude we normalize our data and only measure the angle difference. Dot product is a better measure of similarity if we can ignore magnitude. The cosine of a 0 degree angle is 1, therefore the closer to 1 the cosine similarity is the more similar the similar items are.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Build Recommender System: Based on the analysis and visualizations, it's clear that similar genres tend to have data points that are located close to each other while similar types of songs are also clustered together. This observation makes perfect sense. Similar genres will sound similar and will come from similar time periods while the same can be said for songs within those genres. We can use this idea to build a recommendation system by taking the data points of the songs a user has listened to and recommending songs corresponding to nearby data points.

Spotify is a Python client for the Spotify Web API that makes it easy for developers to fetch data and query Spotify's catalog for songs. You have to install using pip install spotify

After installing Spotify, you will need to create an app on the Spotify Developer's page and save your Client ID and secret key.

CHAPTER 4
RESULTS AND DISCUSSION

CHAPTER 4

RESULTS AND DISCUSSION

4.1 DATA COLLECTION AND PERFORMANCE METRICS

TERMINOLOGIES USED:

1. Spotify datasets:
2. data.csv
3. data_by_artist.csv
4. data_by_year.csv
5. data_by_genres.csv

PERFORMANCE METRICS:

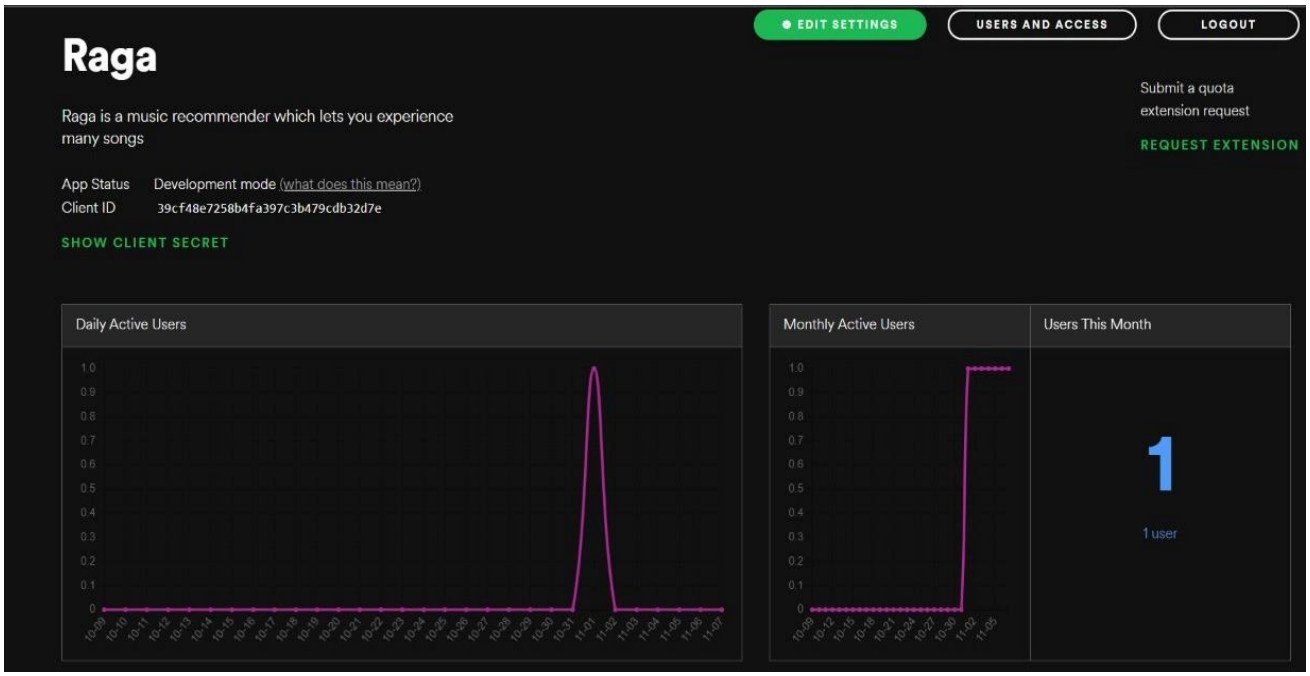
1. It recommends the songs in less time
2. It does the cosine similarity
3. Recommends accurate songs
4. It can recommended ‘n’ number of songs which are similar
5. It given similar songs to the input given

4.2 COMPARISION OF EXISTING SOLUTION

COMPARISION:

1. We are using content-based filtering technique.
2. Most of the existing models are with collaborative filtering but here in our recommendation system we are using content-based filtering.
3. Existing models are more complex often, but we created a simple recommendation system.
4. Compared to the existing, Our system is more accurate.
5. Existing models have some deviations in the predictions but our model is completely dependent on the content and music of the songs.

4.3 RESULTS AND SCREENSHOTS



Import Libraries

```
import os
import numpy as np
import pandas as pd

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist

import warnings
warnings.filterwarnings("ignore")
```

Read Data

```
data = pd.read_csv("../input/spotify-dataset/data/data.csv")
genre_data = pd.read_csv('../input/spotify-dataset/data/data_by_genres.csv')
year_data = pd.read_csv('../input/spotify-dataset/data/data_by_year.csv')
```

In [3]:

```
print(data.info())
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 170653 entries, 0 to 170652
Data columns (total 19 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   valence           170653 non-null  float64
 1   year              170653 non-null  int64
 2   acousticness      170653 non-null  float64
 3   artists           170653 non-null  object
 4   danceability      170653 non-null  float64
 5   duration_ms       170653 non-null  int64
 6   energy            170653 non-null  float64
 7   explicit          170653 non-null  int64
 8   id                170653 non-null  object
 9   instrumentalness  170653 non-null  float64
 10  key               170653 non-null  int64
 11  liveness          170653 non-null  float64
 12  loudness          170653 non-null  float64
 13  mode              170653 non-null  int64
 14  name              170653 non-null  object
 15  popularity        170653 non-null  int64
 16  release_date      170653 non-null  object
 17  speechiness       170653 non-null  float64
 18  tempo             170653 non-null float64
dtypes: float64(9), int64(6), object(4)
memory usage: 24.7+ MB
None
```

```
print(genre_data.info())
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2973 entries, 0 to 2972
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
------------------                    ---------
 0   mode              2973 non-null   int64
 1   genres            2973 non-null   object
 2   acousticness      2973 non-null   float64
 3   danceability      2973 non-null   float64
 4   duration_ms       2973 non-null   float64
 5   energy            2973 non-null   float64
 6   instrumentalness  2973 non-null   float64
 7   liveness          2973 non-null   float64
 8   loudness          2973 non-null   float64
 9   speechiness       2973 non-null   float64
 10  tempo             2973 non-null   float64
 11  valence           2973 non-null   float64
 12  popularity        2973 non-null   float64
 13  key               2973 non-null   int64
dtypes: float64(11), int64(2), object(1)
memory usage: 325.3+ KB
None
```

```
print(year_data.info())
```

OUTPUT:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  ---------
  0   mode              100 non-null    int64
  1   year              100 non-null    int64
  2   acousticness      100 non-null    float64
  3   danceability      100 non-null    float64
  4   duration_ms       100 non-null    float64
  5   energy            100 non-null    float64
  6   instrumentalness  100 non-null    float64
  7   liveness          100 non-null    float64
  8   loudness          100 non-null    float64
  9   speechiness       100 non-null    float64
  10  tempo             100 non-null    float64
  11  valence           100 non-null    float64
  12  popularity        100 non-null    float64
  13  key               100 non-null    int64
dtypes: float64(11), int64(3)
memory usage: 11.1 KB
None
```

We are going to check for all the analysis with the target as popularity. Before going to do that let's check for the Feature Correlation by considering a few features, we are going to use yellowbrick package.

CODE:

```
from yellowbrick.target import FeatureCorrelation

feature_names = ['acousticness', 'danceability', 'energy', 'instrumentalness',
       'liveness', 'loudness', 'speechiness', 'tempo', 'valence','duration_ms'
,'explicit','key','mode','year']

X, y = data[feature_names], data['popularity']

# Create a list of the feature names
features = np.array(feature_names)

# Instantiate the visualizer
visualizer = FeatureCorrelation(labels=features)

plt.rcParams['figure.figsize']=(20,20)
visualizer.fit(X, y)     # Fit the data to the visualizer
visualizer.show(
```

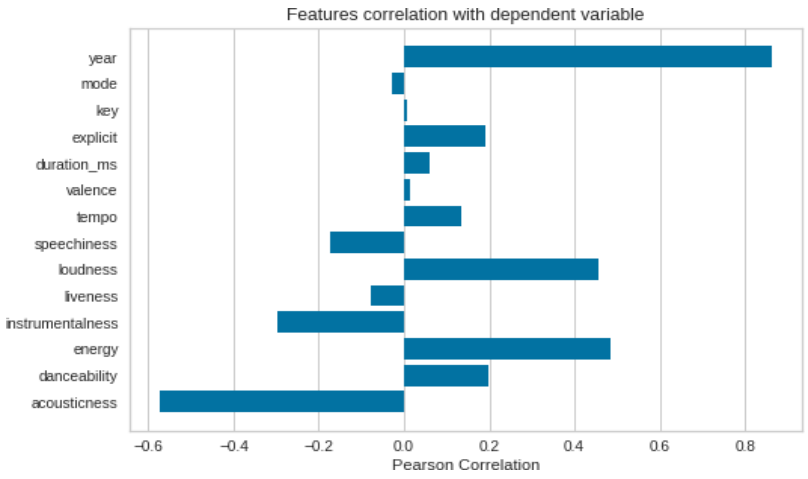


Fig-4

OUTPUT: <matplotlib.axes._subplots.AxesSubplot at 0x7fa91797c210>

Data Understanding by Visualization and EDA

Music Over Time

Using the data grouped by year, we can understand how the overall sound of music has changed from 1921 to 2020.

CODE:

```
def get_decade(year):
    period_start = int(year/10) * 10
    decade = '{}s'.format(period_start)
    return decade

data['decade'] = data['year'].apply(get_decade)

sns.set(rc={'figure.figsize':(11 ,6)})
sns.countplot(data['decade'])
```

```
OUTPUT: <matplotlib.axes._subplots.AxesSubplot at 0x7fa917986a10>
```

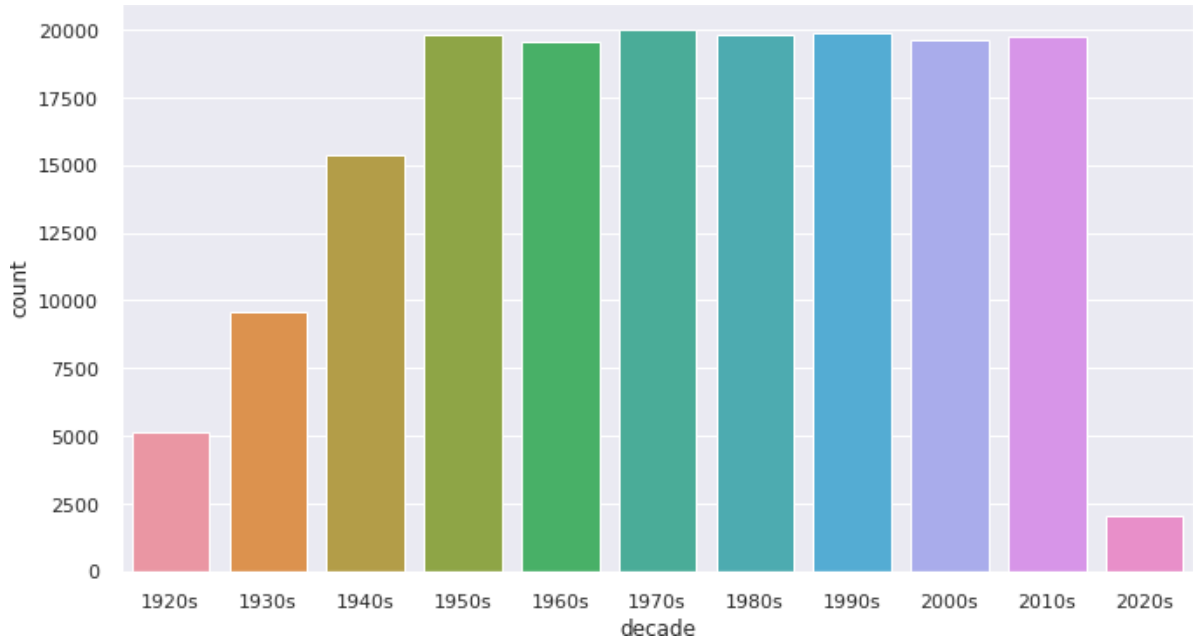


Fig-5

```
sound_features = ['acousticness', 'danceability', 'energy', 'instrumentalness'
, 'liveness', 'valence']
fig = px.line(year_data, x='year', y=sound_features)
fig.show()
```

Fig-6

Characteristics of Different Genres

This dataset contains the audio features for different songs along with the audio features for different genres. We can use this information to compare different genres and understand their unique differences in sound.

```
top10_genres = genre_data.nlargest(10, 'popularity')

fig = px.bar(top10_genres, x='genres', y=['valence', 'energy', 'danceability',
'acousticness'], barmode='group')
fig.show()
```

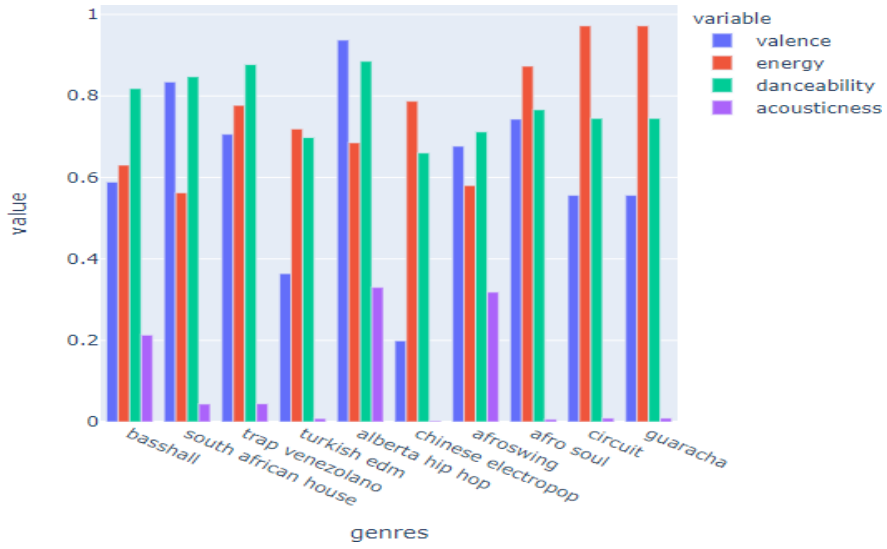


Fig-7

Clustering Genres with K-Means

Here, the simple K-means clustering algorithm is used to divide the genres in this dataset into ten clusters based on the numerical audio features of each genres.

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

cluster_pipeline = Pipeline([('scaler', StandardScaler()), ('kmeans', KMeans(n
_clusters=10, n_jobs=-1))])
X = genre_data.select_dtypes(np.number)
cluster_pipeline.fit(X)
genre_data['cluster'] = cluster_pipeline.predict(X)

# Visualizing the Clusters with t-SNE

from sklearn.manifold import TSNE

tsne_pipeline = Pipeline([('scaler', StandardScaler()), ('tsne', TSNE(n_compon
ents=2, verbose=1))])
genre_embedding = tsne_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=genre_embedding)
projection['genres'] = genre_data['genres']
projection['cluster'] = genre_data['cluster']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'genres']
)
fig.show()
```

```
OUTPUT:
[t-SNE] Computing 91 nearest neighbors...
[t-SNE] Indexed 2973 samples in 0.005s...
[t-SNE] Computed neighbors for 2973 samples in 0.322s...
[t-SNE] Computed conditional probabilities for sample 1000 / 2973
[t-SNE] Computed conditional probabilities for sample 2000 / 2973
[t-SNE] Computed conditional probabilities for sample 2973 / 2973
[t-SNE] Mean sigma: 0.777516
[t-SNE] KL divergence after 250 iterations with early exaggeration: 76.115768
[t-SNE] KL divergence after 1000 iterations: 1.392461
```

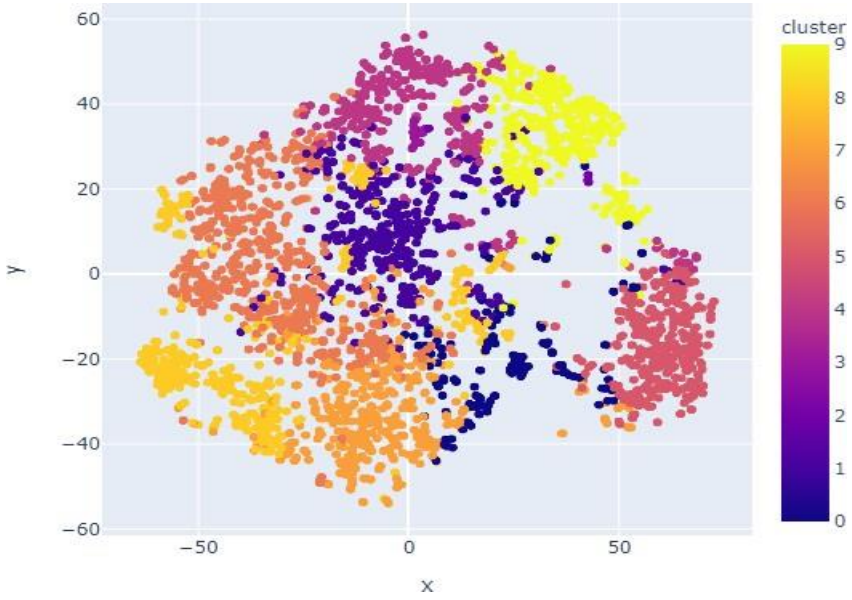


Fig-8

Clustering Songs with K-Means

```
song_cluster_pipeline = Pipeline([('scaler', StandardScaler()),
                                  ('kmeans', KMeans(n_clusters=20,
                                   verbose=False, n_jobs=4))
                                 ], verbose=False)

X = data.select_dtypes(np.number)
number_cols = list(X.columns)
song_cluster_pipeline.fit(X)
song_cluster_labels = song_cluster_pipeline.predict(X)
data['cluster_label'] = song_cluster_labels


# Visualizing the Clusters with PCA

from sklearn.decomposition import PCA

pca_pipeline = Pipeline([('scaler', StandardScaler()), ('PCA', PCA(n_component
s=2))])
song_embedding = pca_pipeline.fit_transform(X)
projection = pd.DataFrame(columns=['x', 'y'], data=song_embedding)
projection['title'] = data['name']
projection['cluster'] = data['cluster_label']

fig = px.scatter(
    projection, x='x', y='y', color='cluster', hover_data=['x', 'y', 'title'])
fig.show()
```

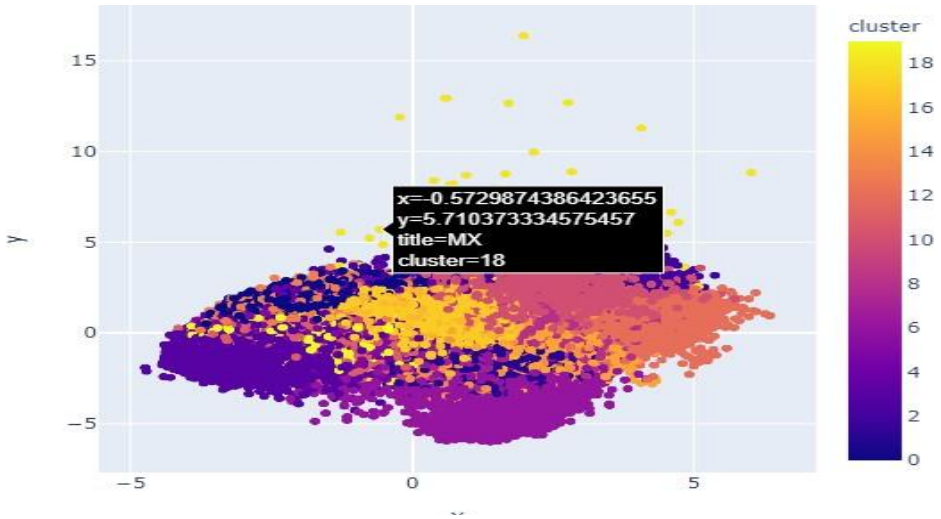


Fig-9

Build Recommender System

Based on the analysis and visualizations, it's clear that similar genres tend to have data points that are located close to each other while similar types of songs are also clustered together. This observation makes perfect sense. Similar genres will sound similar and will come from similar time periods while the same can be said for songs within those genres. We can use this idea to build a recommendation system by taking the data points of the songs a user has listened to and recommending songs corresponding to nearby data points. Spotipy is a Python client for the Spotify Web API that makes it easy for developers to fetch data and query Spotify's catalog for songs. You have to install using pip install spotipy. After installing Spotipy, you will need to create an app on the Spotify Developer's page and save your Client ID and secret key.

```
!pip install spotipy
import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
from collections import defaultdict

sp = spotipy.Spotify(auth_manager=SpotifyClientCredentials(client_id=os.enviro
n["SPOTIFY_CLIENT_ID"],
                                                           client_secret=os.en
viron["SPOTIFY_CLIENT_SECRET"]))

def find_song(name, year):
    song_data = defaultdict()
    results = sp.search(q= 'track: {} year: {}'.format(name,year), limit=1)
    if results['tracks']['items'] == []:
        return None



results = results['tracks']['items'][0]
```

```
    track_id = results['id']
    audio_features = sp.audio_features(track_id)[0]
    song_data['name'] = [name]
    song_data['year'] = [year]
    song_data['explicit'] = [int(results['explicit'])]
    song_data['duration_ms'] = [results['duration_ms']]    track_id = results[
'id']
    audio_features = sp.audio_features(track_id)[0]
```

```
    song_data['popularity'] = [results['popularity']]
    for key, value in audio_features.items():
        song_data[key] = value

    return pd.DataFrame(song_data)

from collections import defaultdict
from sklearn.metrics import euclidean_distances
from scipy.spatial.distance import cdist
import difflib

number_cols = ['valence', 'year', 'acousticness', 'danceability', 'duration_ms
', 'energy', 'explicit',
 'instrumentalness', 'key', 'liveness', 'loudness', 'mode', 'popularity', 'spe
echiness', 'tempo']
def get_song_data(song, spotify_data):
try:
        song_data = spotify_data[(spotify_data['name'] == song['name'])
                                 & (spotify_data['year'] == song['year'])].iloc[0]
        return song_data
    except IndexError:
        return find_song(song['name'], song['year'])
def get_mean_vector(song_list, spotify_data):
    song_vectors = []
    for song in song_list:
        song_data = get_song_data(song, spotify_data)
        if song_data is None:
print('Warning: {} does not exist in Spotify or in database'.format(song['name
'']))
            continue
        song_vector = song_data[number_cols].values
        song_vectors.append(song_vector)

    song_matrix = np.array(list(song_vectors))
    return np.mean(song_matrix, axis=0)


def flatten_dict_list(dict_list):

    flattened_dict = defaultdict()
    for key in dict_list[0].keys():
       flattened_dict[key] = []
```

```
for dictionary in dict_list:

        for key, value in dictionary.items():
  flattened_dict[key].append(value)

    return flattened_dict

def recommend_songs( song_list, spotify_data, n_songs=10):

    metadata_cols = ['name', 'year', 'artists']
    song_dict = flatten_dict_list(song_list)

    song_center = get_mean_vector(song_list, spotify_data)
    scaler = song_cluster_pipeline.steps[0][1]
    scaled_data = scaler.transform(spotify_data[number_cols])
    scaled_song_center = scaler.transform(song_center.reshape(1, -1))
    distances = cdist(scaled_song_center, scaled_data, 'cosine')
    index = list(np.argsort(distances)[:, :n_songs][0])

rec_songs = spotify_data.iloc[index]
    rec_songs = rec_songs[~rec_songs['name'].isin(song_dict['name'])]
    return rec_songs[metadata_cols].to_dict(orient='records')
```

In [ ]:

```
linkcode
recommend_songs([{'name': 'Come As You Are', 'year':1991},
                 {'name': 'Smells Like Teen Spirit', 'year': 1991},
                 {'name': 'Lithium', 'year': 1992},
                 {'name': 'All Apologies', 'year': 1993},
                 {'name': 'Stay Away', 'year': 1993}],  data)
```

This last cell will gives you a recommendation list of songs like this,

```
[{'name': 'Life is a Highway - From "Cars"',

  'year': 2009,

  'artists': "['Rascal Flatts']"},

 {'name': 'Of Wolf And Man', 'year': 1991, 'artists': "['Metallica']"},

 {'name': 'Somebody Like You', 'year': 2002, 'artists': "['Keith Urban']"},

 {'name': 'Kayleigh', 'year': 1992, 'artists': "['Marillion']"},

 {'name': 'Little Secrets', 'year': 2009, 'artists': "['Passion Pit']"},

 {'name': 'No Excuses', 'year': 1994, 'artists': "['Alice In Chains']"},

 {'name': 'Corazón Mágico', 'year': 1995, 'artists': "['Los Fugitivos']"},

 {'name': 'If Today Was Your Last Day','year': 2008,'artists': "['Nickelback']
"},

{'name': "Let's Get Rocked", 'year': 1992, 'artists': "['Def Leppard']"},

 {'name': "Breakfast At Tiffany's",'year': 1995,'artists': "['Deep Blue Someth
ing']"}]
```


You can change the given songs list as per your choice.


```
{'name': 'No Excuses', 'year': 1994, 'artists': "['Alice In Chains']"},

 {'name': 'Corazón Mágico', 'year': 1995, 'artists': "['Los Fugitivos']"},

 {'name': 'If Today Was Your Last Day','year': 2008,'artists': "['Nickelback']
"},

{'name': "Let's Get Rocked", 'year': 1992, 'artists': "['Def Leppard']"},

 {'name': "Breakfast At Tiffany's",'year': 1995,'artists': "['Deep Blue Someth
ing']"}]
```

You can change the given songs list as per your choice.

CHAPTER 5
CONCLUSION

CONCLUSION

The following are our conclusions based on experiment results. The music genre information to increase the quality of music recommendations. The music recommender is able to recommend the songs based on the song features. The music Recommender is able to check the

dataset taken by generating the similarity score for each recommended song. The mood of the song is predicted by examining the lyrics of the given song with all the other songs in the dataset and predicting the mood and similarity scores and recommending the songs based on the mood. Based on our analyses, we can suggest for future research to add other music features in order to improve the accuracy of the recommender system. Designing a personalized music recommender is complicated, and It is challenging to thoroughly understand the user needs and meet their requirements. The future research direction will be mainly focused on user-centric music recommender systems. It indicates the effectiveness of its hybrid structure to extract the music features. Based on our analyses, we can suggest for future research to add other music features in order to improve the accuracy of the recommender system.

CHAPTER 6
REFERENCES

REFERENCES

[1] R. Ragno, C. Burges and C. Herley: “Inferring similarity between music objects with application to playlist generation,” in Proc. of 7th ACM Multimedia, Workshop on MIR, pp. 73–80, 2005.

[2] A. Flexer, D. Schnitzer, M. Gasser and G. Widmer: “Playlist generation using start and end songs,” in Proc. of 9th ISMIR, pp. 173–178, 2008.

[3] E. Pampalk, T. Pohle and G. Widmer: “Dynamic playlist generation based on skipping behavior” in Proc. of 6th ISMIR, pp. 634–637, 2005.

[4] W. W. Cohen and W. Fan: “Web-collaborative filtering: Recommending music by crawling the web,” Computer Network, Vol. 33, pp. 685–698, 2000.

[5] P. Cano, M. Koppenberger and N. Wack: “An industrialstrength content-based music recommendation system,” in Proc. of 28th ACM SIGIR, pp. 673, 2005.

[6] R. Cai, C. Zhang, C. Wang, L. Zhang and W. Ma: “MusicSense: Contextual music recommendation using emotional allocation modeling,” in Proc. of ACM Multimedia, pp. 553–556, 2007.

[7] J. Donaldson: “A hybrid social-acoustic recommendation system for popular music,” in Proc. of the ACM Recommender Systems, pp. 187–190, 2007.

[8] B. Shao, D. Wang, T. Li and M. Ogihara: “Music recommendation based on acoustic features and user access patterns,” IEEE Trans. on Audio, Speech And Language Processing, Vol. 17, No. 8, pp. 1602–1611, 2009.

[9] N. Liu, S. Lai, C. Chen and S. Hsieh: “Adaptive music recommendation based on user behaviour in time slot,” International Journal of Computer Science and Network Security, Vol. 9, pp. 219–227, 2009.

[10] J. Su and H. Yeh: “Music recommendation using content and context information mining,” IEEE Intelligent Systems, Vol. 25, pp. 16–26, 2010.

[11] G. E. P. Box, and D. A. Pierce: “Distribution of residual autocorrelation

s in autoregressive-integrated moving average time series models,” Jour. of the American Statistical Association, Vol. 65, pp. 1509–1526, 1970.

[12] B. Logan: “Music recommendation from song sets,” in Proc. of 5th ISMIR, pp. 425–428, 2004

[13] Stafford SA. Music in the Digital Age: The Emergence of Digital Music and Its Repercussions on the Music Industry. The Elon Journal of Undergraduate Research in Communications Vol. 1 No. 2. 2010.

[14] Aggarwal CC. Recommender Systems: The Textbook. 1st ed.: Springer; 2016.

[15] Oord Avd, Dieleman S, Schrauwen B. Deep content-based music recommendation. Advances in Neural Information Processing Systems 2013.

[16] O'Bryant J. A survey of music recommendation and possible improvements 2017

[17] Labrosa.ee.columbia.edu. (2017). Million Song Dataset | scaling MIR research. [online] Available at: https://labrosa.ee.columbia.edu/millionsong/ [Accessed 10 Oct. 2017].

[18] Linden, G., Smith, B. and York, J. (2003).

[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering.