

Music Recommender System

REPORT SUBTITLE

Atharva Shekatkar | 20BDS0067 | Date

Arnab Banik | 20BDS0067 | Date

Parshva Maniar | 20BDS0067 | Date

Tejas Rokade | 20BDS0067 | Date

Abstract

Recommendation systems are widely used in various fields, whether it be on music and video streaming websites or for businesses. The inspiration for this project came from having an overview of clustering machine learning algorithms and the curiosity about how various recommendations systems worked on various streaming sites. Our team’s goal is to analyze and create such a recommendation system, specifically for music, using unsupervised clustering machine learning algorithms like BIRCH, K-means and Mean-shift algorithms. The most accurate model will be used to give recommendations based on a few songs the user has already listened to. The data being used will be Spotify’s dataset available on Kaggle, which gives information about a large number of songs that are available on Spotify. The data will be preprocessed, visualized and then classified into clusters while training the model.

Contents

[INTRODUCTION 3](#__RefHeading___Toc151_3335915281)

[Purpose 3](#__RefHeading___Toc153_3335915281)

[Scope 3](#__RefHeading___Toc155_3335915281)

[ACQUIRING THE DATASET 4](#__RefHeading___Toc157_3335915281)

[EXPLORATORY DATA ANALYSIS 4](#__RefHeading___Toc159_3335915281)

[Checking the dataset 4](#__RefHeading___Toc118_1708921397)

[Data cleaning 6](#__RefHeading___Toc120_1708921397)

[Data visualization 6](#__RefHeading___Toc122_1708921397)

[TRAINING THE MODELS 9](#__RefHeading___Toc124_1708921397)

# INTRODUCTION

This document lays out a project plan for the development of “Music Recommender System”.

The plan will include a summary of the system functionality, scope of the project from the perspective of the team, the approach to developing the project and metrics and measurements that will be recorded throughout the project.

This document will also cover a detailed analysis of the procedure of development of the project, including obtaining the dataset, preprocessing, analysis of various algorithms, evaluating the best algorithm and then obtaining the final output.

## Purpose

The goal of this project is to make a music recommender system, which will recommend new songs to the user based on a few songs the user has listened to previously.

## Scope

The “Music Recommender System” is a machine learning model which will help users find new music which is similar to the user’s preference in music. The model will be hosted on a website online.

A user must call the model and provide the names of the songs they listened to. The model then identifies and finds each song in the database and then recommends songs similar to it based on various calculations.

The model should be free to call by anyone.

# ACQUIRING THE DATASET

The dataset used for training various models for the music recommender system is taken from Kaggle. The dataset contains multiple files, which have data arranged according to genre and year and have the main training dataset. Each one contains various attributes for each entry such as the name of the song, the artist, whether the song is explicit, whether the song is suitable for dancing, etc.

The dataset has both categorical as well as continuous variables and is thus suitable for making a recommendation system which the team is aiming for.

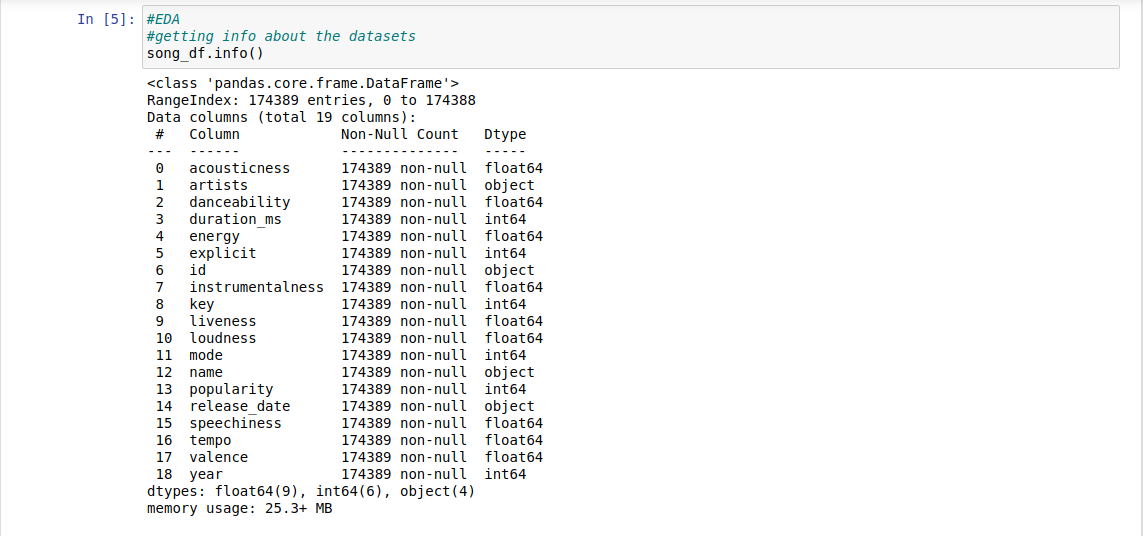
The dataset has been uploaded on the github repository for this project, the link for which can be found in the Links section.

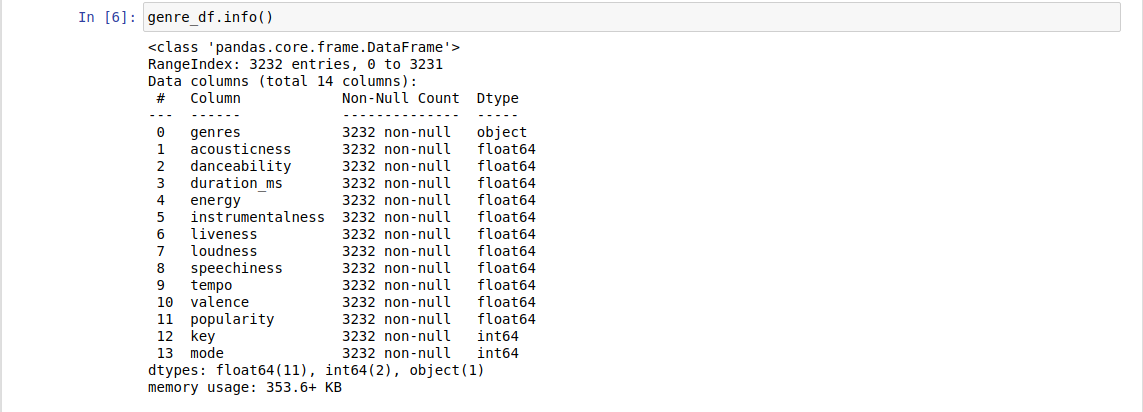
# EXPLORATORY DATA ANALYSIS

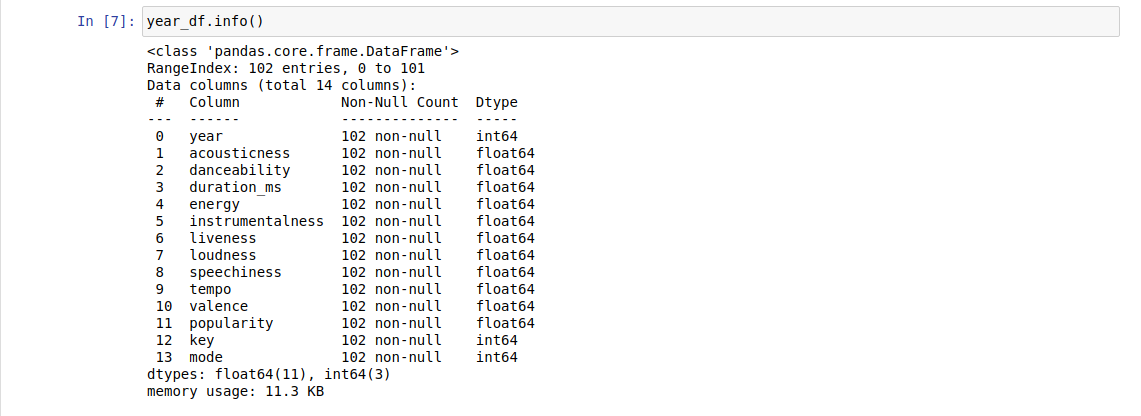
Under this section, various stages of exploratory data analysis are discussed. We first discuss data cleaning, followed by data visualization.

## Checking the dataset

Here, we first check the dataset to see what type of values it contains and whether there are any null entries in any column. This can easily be done using the info function from pandas library.



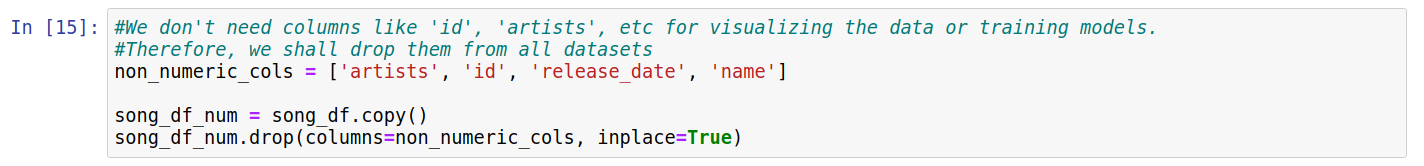




As we can see, out of the total entries in each dataset, none are null. Therefore, we do not need to drop any rows or replace any entry for NA values.

## Data cleaning

We saw that there are string values present in the **song\_df** dataset, which will pose a hindrance while training the dataset. Therefore, we make a copy of the dataset and drop those columns.

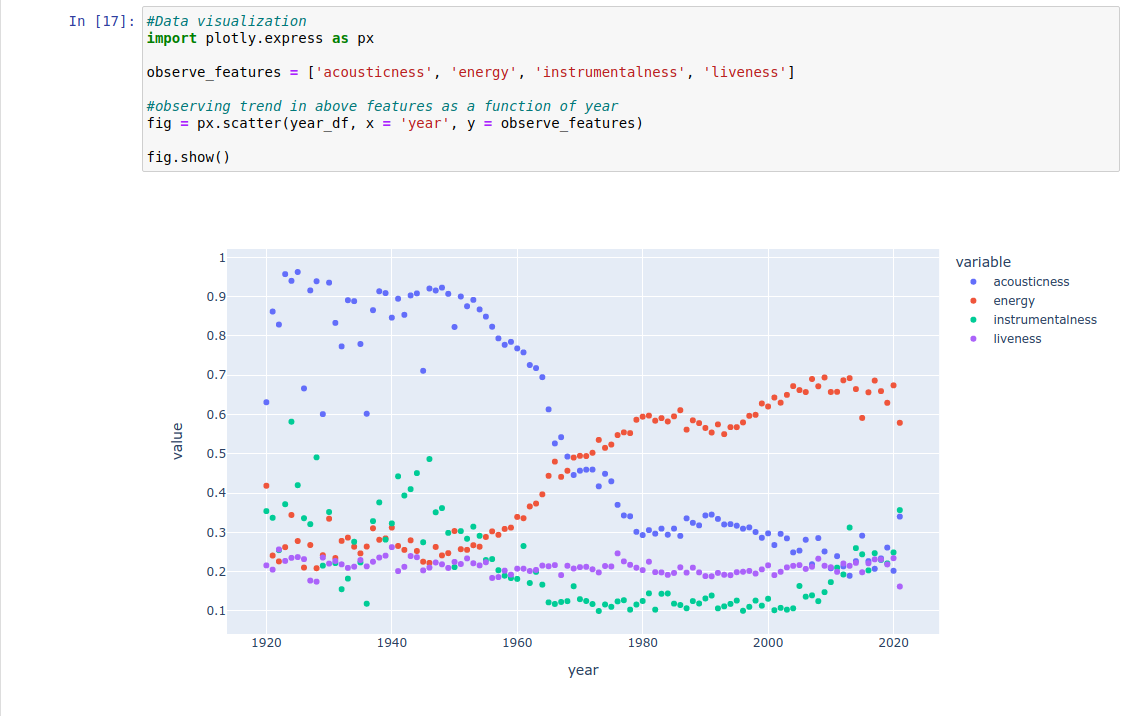


We do not drop any columns from the **genre\_df** and **year\_df** since those dataframes are only going to be used for visualization.

## Data visualization

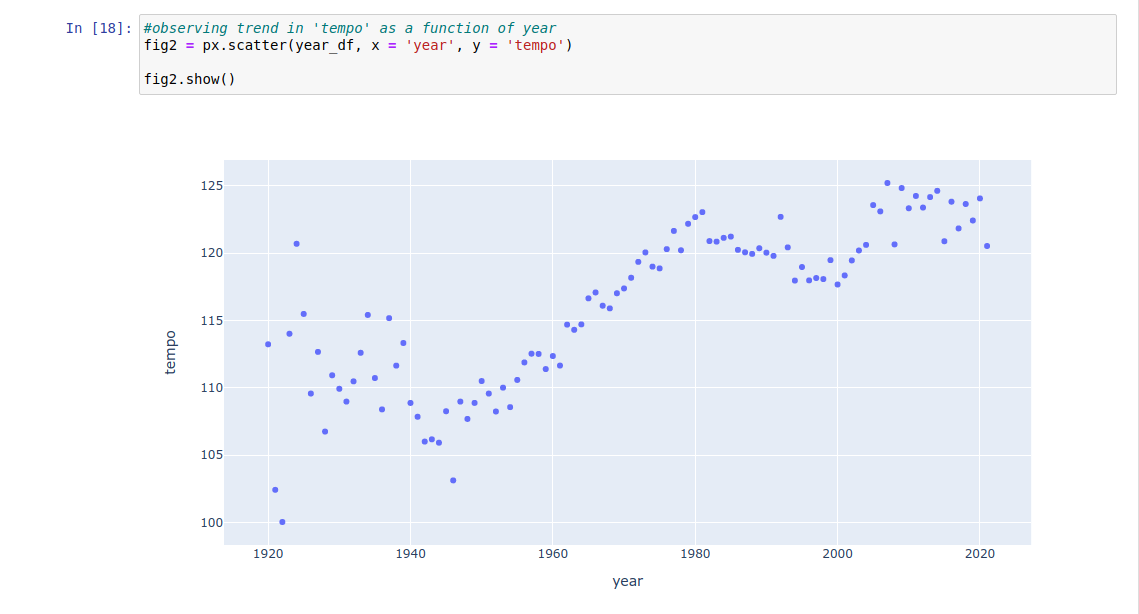
We try to visualize the data in an attempt to understand various trends in music over the period of time and how various features can be correlated to each other.

First, let us see how ‘acousticness’, ‘energy’, ‘instrumentalness’ and ‘liveness’ have varied in the past 100 years.



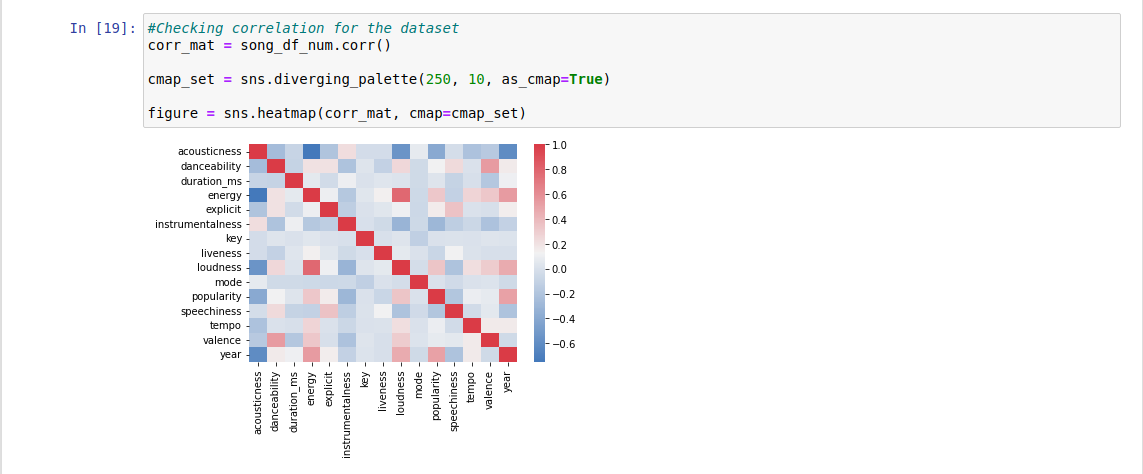
We see that in the past 100 years, acousticness in music has gone down, with a rise in energy in music. This may be because of advancements in audio engineering and various software available to create and edit music digitally. We also notice that instrumentalness and liveness have remained the same in music.

Next, let us see how tempo has varied in the past 100 years



We see that tempo of music has increased considerably as well. This is in accordance with the increase in energy of the music, as usually energizing songs have higher tempo.

Finally, let’s see the correlation matrix as a heatmap to see which features are correlated with each other.



From the above figure, we can see that acousticness is inversely correlated with energy, loudness as well as year. We can also make out that energy and loudness are positively correlated with each other.

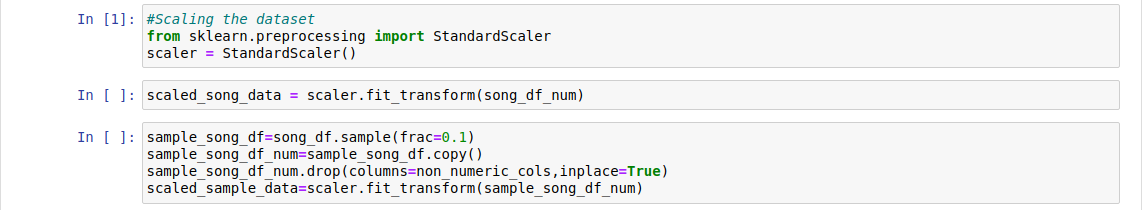
This concludes the exploratory data analysis section. We shall now move onto training the models for song recommendation.

# TRAINING THE MODELS

In this section, we first scale our dataset. Then, we train various models based on different algorithms on the scaled dataset and test them.

## Scaling the dataset

We scale the dataset using **StandardScaler()** function available in the sklearn.preprocessing library.



We also make a smaller dataset out of the larger dataset as some of the algorithms that we test do not scale well for larger datasets. Hence, we take a random sample of 10% of the total dataset and store it in **sample\_song\_df**. We store the columns with numerical data as before and then, scale both the sample song dataset as well as the original song dataset.

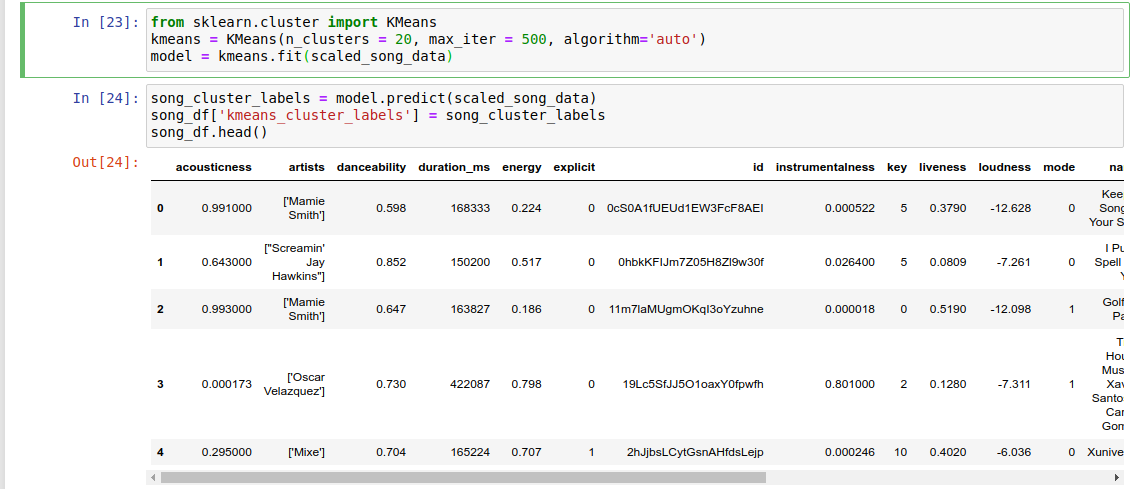
## Implementing the algorithms

### 1. KMeans:

The KMeans algorithm is a clustering algorithm which clusters data by trying to separate samples in n groups of equal variance. It requires the number of clusters to be specified. It also scales well to large number of samples and hence is used across a large number of applications. It is one of the most popular clustering algorithms.

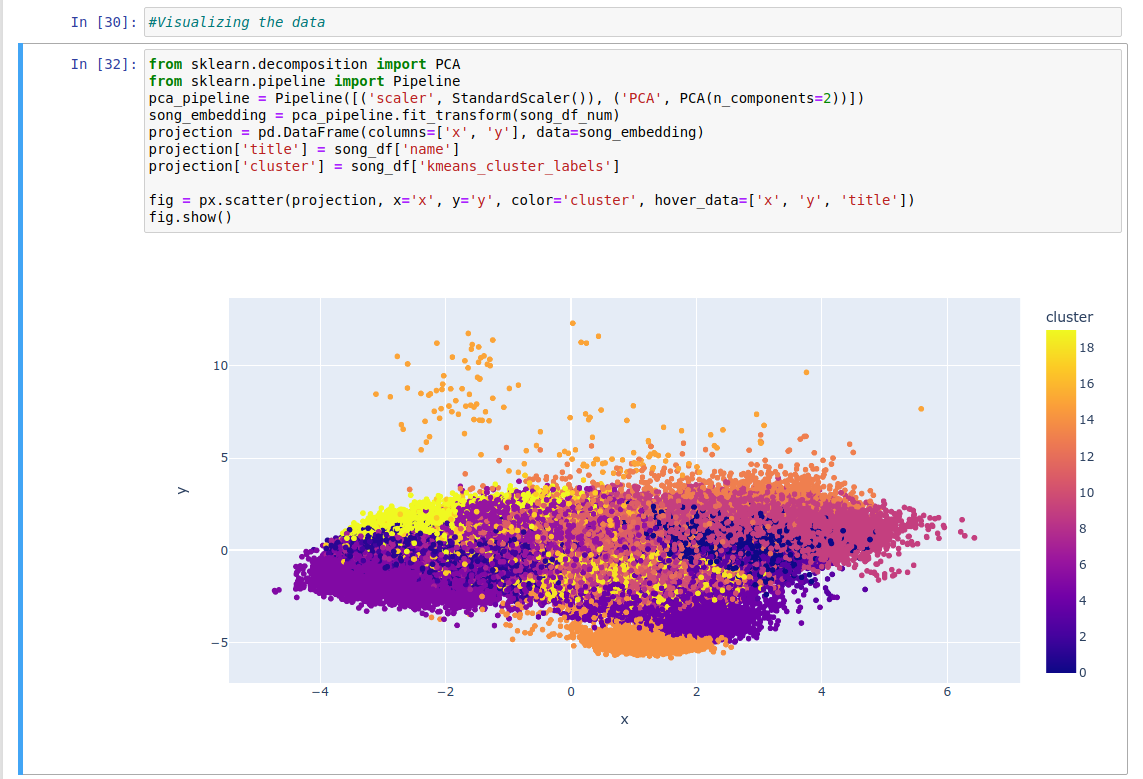
For our project, we use the **KMeans** module from **sklearn.clustering** library and feed it our dataset using the **fit** function. Since the KMeans algorithm scales well to large datasets, we can use our original dataset for fitting the algorithm. We then use the model to make predictions on our dataset itself and add the predicted cluster numbers, in a new column, to the **song\_df** dataframe.

Following is the code



We can visualize the clusters using **PCA** (Principal Component Analysis) and **plotly** modules.

Following is the plot we get:



It should be noted that the plot is interactive and displays the song name and cluster number when the mouse pointer hovers over the dot. However, one must run the notebook to make the plot visible due to its interactive nature.

### 2. BIRCH:

Birch is a clustering algorithm which builds a tree called the Clustering Feature Tree (CFT) for the given dataset. The data is compressed to a set of Clustering Feature nodes (CF nodes). The CF Nodes have a number of subclusters called Clustering Feature subclusters and these subclusters can have CF nodes as children. The CF Subclusters hold the necessary information for clustering which prevents the need to hold the entire input data in memory.

For our project, we use the **Birch** module from the **sklearn.clustering** library and feed it our dataset.