**Report on**

**“Music Recommendation System”**

**By**

**Team:Vornoi Practitioners**

| **Sl. No** | **Team Member** | **Email ID** |
| --- | --- | --- |
|  | Mrinal Bhardwaj (Captain) | mrinal.nbd@gmail.com |
|  | Ashutosh Kumar Bharti | ashubharti65@gmail.com |
|  | Ankitha R Desai | ankitha.desai0816@gmail.com |
|  | Muskan A | muskan.af18@gmail.com |
|  | Rutvik S Jaiswal | rutvikjaiswal195@gmail.com |

**Table of Contents**

[**Introduction:**](#_8lnneq1fqtqp) **3**

[**Problem Statement:**](#_wkft4ywqxp98) **3**

[**Requirement gathering:**](#_4vavy0nxoytx) **3**

[1.Dataset Description:](#_szh407h3gd4o) 3

[2. Data Preprocessing:](#_hz0xw1oi7yqc) 3

[3. Data Cleaning:](#_io4h871u8ut2) 4

[4. Nearest Neighbor Algorithm:](#_r5ppq3ne0xgv) 4

[5. Content Based Recommendation:](#_6yz0hreinglk) 5

[6. Functional Requirements:](#_6u56km7ly9hz) 5

[7. Non- Functional Requirements:](#_1s28d9uajg4z) 5

[**Implementation:**](#_3uylid2jhn53) **6**

[Workflow Diagram:](#_k9csi3ksk4kg) 6

[Use Case Diagram:](#_c6j216t2pdpk) 6

[UML Diagram:](#_ffhvk1jkot2y) **7**

[**Results:**](#_sjrnvlv3b7bu) **8**

[**Conclusion and Future Scope:**](#_zeks1z7qwjhu) **8**

[**References:**](#_smdf61p6u5u9) **8**

# **Introduction:**

With the unexpected development of streaming services like Apple's iTunes, Spotify, and Pandora, the music industry is currently in a metaphorical stage where the change to digital distribution is becoming increasingly visible. As a result, automatic music recommendation has become a more pressing issue, allowing users to discover new music while also helping businesses to target the correct audience.

# **Problem Statement:**

The goal of the project is to design and develop a “Music Recommendation System” that will detect preferences and produce a playlist based on them. We are using the spotify dataset provided by Kaggle and building this recommendation system using Content based recommendation and Nearest Neighbor algorithm.

# **Requirement gathering:**

## 1.Dataset Description:

The spotify dataset was acquired from Kaggle.Dataset contains more than 1,60,000 songs collected from Spotify Web API. The features include song, artist, release date as well as some characteristics of the song such as acousticness, danceability, loudness, tempo and so on.

Feature definition of few features:

1. Artist: The list of artists of the song
2. Danceability: Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.
3. Duration\_ms: The duration of the track in milliseconds.
4. Energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
5. Instrumentalness: Predicts whether a track contains no vocals.The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content.
6. Key: The key the track is in. Integers map to pitches using standard Pitch Class notation .
7. Liveness: Detects the presence of an audience in the recording.A value above 0.8 provides strong likelihood that the track is live.
8. Loudness: The overall loudness of a track in decibels (dB).

## 

## 2. Data Preprocessing:

1. The process of transforming raw data into an understandable format.
2. It is done to improve the quality of data in data warehouse
3. Increase efficiency
4. Ease of mining process
5. Removes noisy data, inconsistent data and data with missing values.

## 3. Data Cleaning:

1. Process of removal of incorrect, incomplete, inaccurate data, also replaces missing data
2. It cleans the data by :
   1. Filling in the missing values.
   2. Smoothing noisy data.
   3. Resolving the inconsistency.
   4. Removing the outlier.
3. Ways to handle Missing Data during cleaning :
   1. Manual entry of missing data
   2. Using attribute mean
   3. Using most probable value
   4. Using Global Constant
   5. Ignore the tuple

## 4. Nearest Neighbor Algorithm:

* Nearest Neighbors is one of the most basic yet essential classification algorithms of Machine Learning.
* It belongs to the unsupervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.
* It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data.
* The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point, and predict the label from these.
* The number of samples can be a user-defined constant (k-nearest neighbor learning), or vary based on the local density of points (radius-based neighbor learning).
* The distance can, in general, be any metric measure: standard Euclidean distance is the most common choice. Neighbors-based methods are known as *non-generalizing* machine learning methods, since they simply “remember” all of its training data.

**Steps of Nearest Neighbor Algorithm :**

1. Select the number of the neighbors.
2. Calculate the Euclidean distance of the number of neighbors.
3. Take the nearest neighbors as per the calculated Euclidean distance.
4. Among these neighbors, count the number of the data points in each category.
5. Assign the new data points to that category for which the number of the neighbor is maximum.
6. Model is ready.

**Advantages:**

* They are simple to put into practice.
* They can also produce good outcomes if the features are appropriately selected (and if they are weighted carefully in the computation of the distance).
* During training, only a minimal amount of time is required.
* This algorithm can constantly evolve with the dataset.

**Disadvantages:**

* Nearest neighbors are extremely sensitive to the existence of irrelevant information.
* These methods are also rather slow if the training set has many examples.

## 5. Content Based Recommendation:

* A Content-Based Recommender is based on the data we get from users, either explicitly (ratings) or implicitly (clicking on a link).
* We develop a user profile based on the data, which is then used to provide recommendations to the user. As the user offers additional information or acts on the advice, the engine becomes more accurate.
* **User Profile:**
  + We generate vectors in the User Profile that describe the user's preferences. The utility matrix, which describes the link between user and object, is used to create a user profile. With this information, the best guess we can make about which item a user prefers is a combination of those things' profiles.
* **Item Profile:**
  + We must create a profile for each item in Content-Based Recommender, which will describe the significant attributes of that item.
* **Recommending Items to User Based on Content:**
  + **Method 1:**
    - The cosine distance between the item's and the user's vectors can be used to establish the item's preference for the user.
  + **Method 2:**
    - We may use a classification technique in recommendation systems as well, such as using the Decision Tree to determine if a user wants to watch a music or not, and applying a condition at each level to refine our suggestion.

## 6. Functional Requirements:

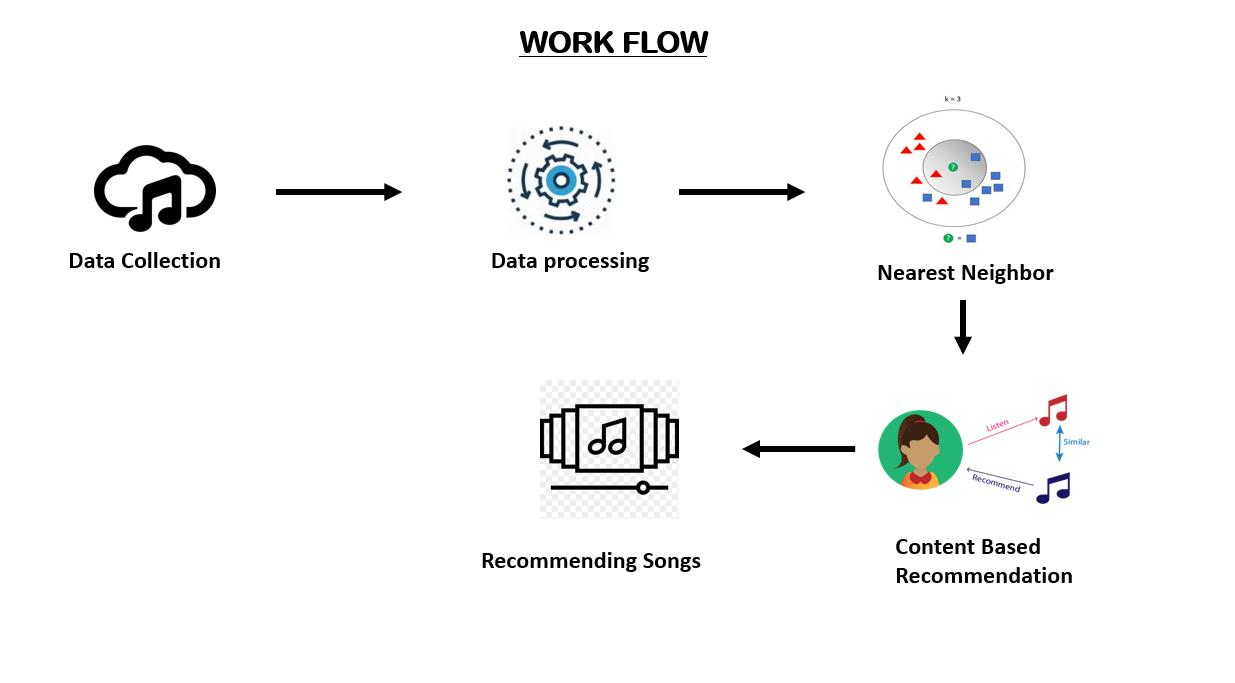
* **Handle requests for recommendations**
  + Requests for recommendations must be obtained and handled by the server application.
* **Store evaluations**
  + The server programme is responsible for receiving and storing music ratings.
* **Data storing**
  + The newly retrieved data must be stored in the database by the server application.
* **Recommend using content based filtering**
  + The server application must be capable of generating recommendations based on the content and user evaluations.

## 7. Non- Functional Requirements:

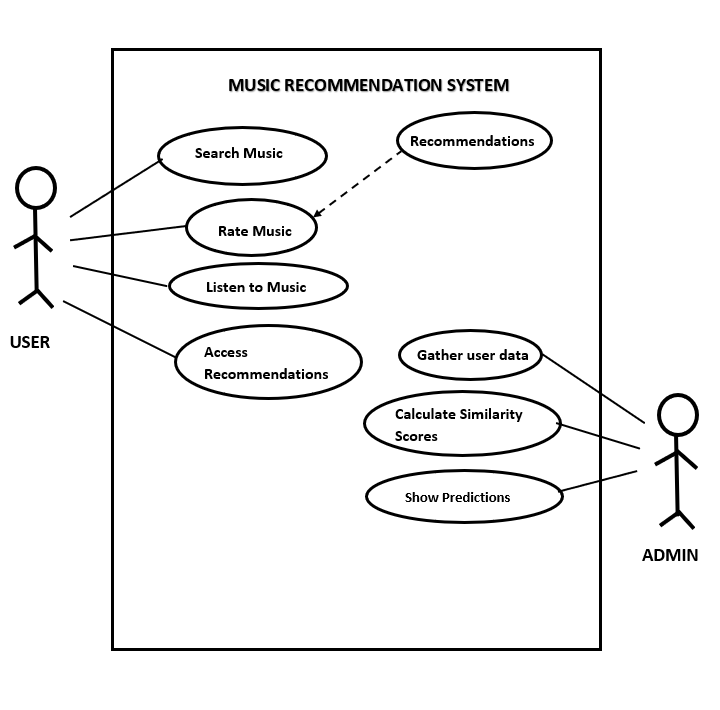
* **Accuracy:** Because we will prioritize software correctness, the Music Recommender's performance will be determined by its suggestion accuracy.
* **Failure handling:** Some system components may fail without affecting others. As a result, system components must be designed to withstand the failure of other components on which they rely.
* **Transparency:** To ensure that the system is usable for a fair period of time, it should be expandable.
* **Security:** Information that is sensitive should be kept secure.

# **Implementation:**

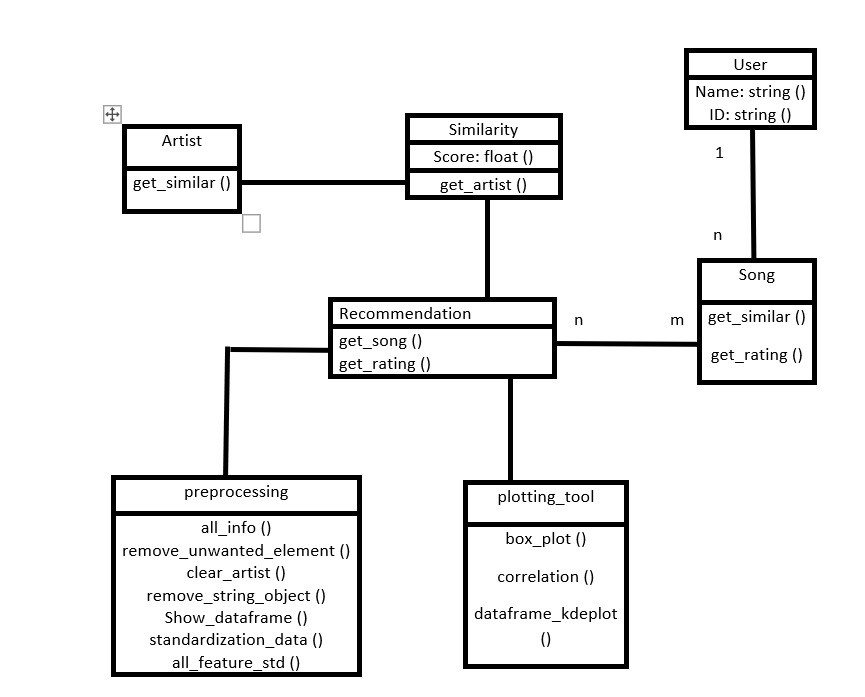
## Workflow Diagram:



## Use Case Diagram:

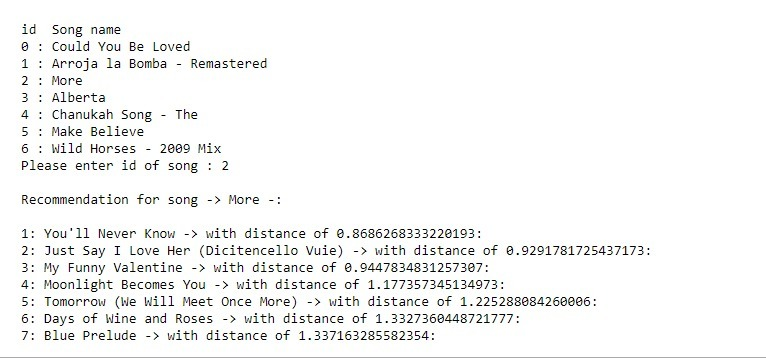


## UML Diagram:



# **Results:**

The recommendation system provides users ten tracks with their IDs. Based on the user's selection of ID, it recommends seven tracks that are similar to the track of the user's choice. Check out the following scenario:

****

# **Conclusion and Future Scope:**

Recommender systems are a powerful new technology that helps organizations to get more value out of their user information. These technologies aid customers in locating items they want to buy from an organization. Users benefit from recommender systems since they can find products that they like.

The attributes of a song used in this recommender system are tempo, energy, liveliness, loudness, danceability, and many other important characteristics are utilized. The recommender system takes into account all the relevant features of a song that practically describes all the qualities of a song.

# **References:**

* <https://www.kaggle.com/code/vatsalmavani/music-recommendation-system-using-spotify-dataset/notebook>
* <https://towardsdatascience.com/part-iii-building-a-song-recommendation-system-with-spotify-cf76b52705e7>
* <https://github.com/caserec/Datasets-for-Recommender-Systems>