Spotify Cluster Based Music Recommender

Abtin Mahyar

1. Overview

In this project, I created a terminal-based python application capable of reading a dataset as the input consisting of feature analyses of some music that a specific user like which is provided by Spotify's API. The application uses a classifier trained using the large dataset consisting of available music on Spotify and their genre to predict genres of music in the input dataset. Then music will be divided into different clusters by a clustering algorithm which is trained with the main dataset. After that, five playlists are recommended to the user based on total number of input music in each cluster.

2. Problem Statement

The goal is to create a cluster-based music recommender; the tasks involved are the following:

1. Download and preprocess Spotify music dataset which is provided on Kaggle.
2. Train a classifier that can determine genres of each music and extract importance of every feature.
3. Run the classifier on user's input dataset.
4. Find and train the best clustering algorithm that can divide music in the dataset into well-separated clusters.
5. Run the trained cluster algorithm on input dataset.
6. Choose the top five clusters for each user and recommend related music in those clusters.

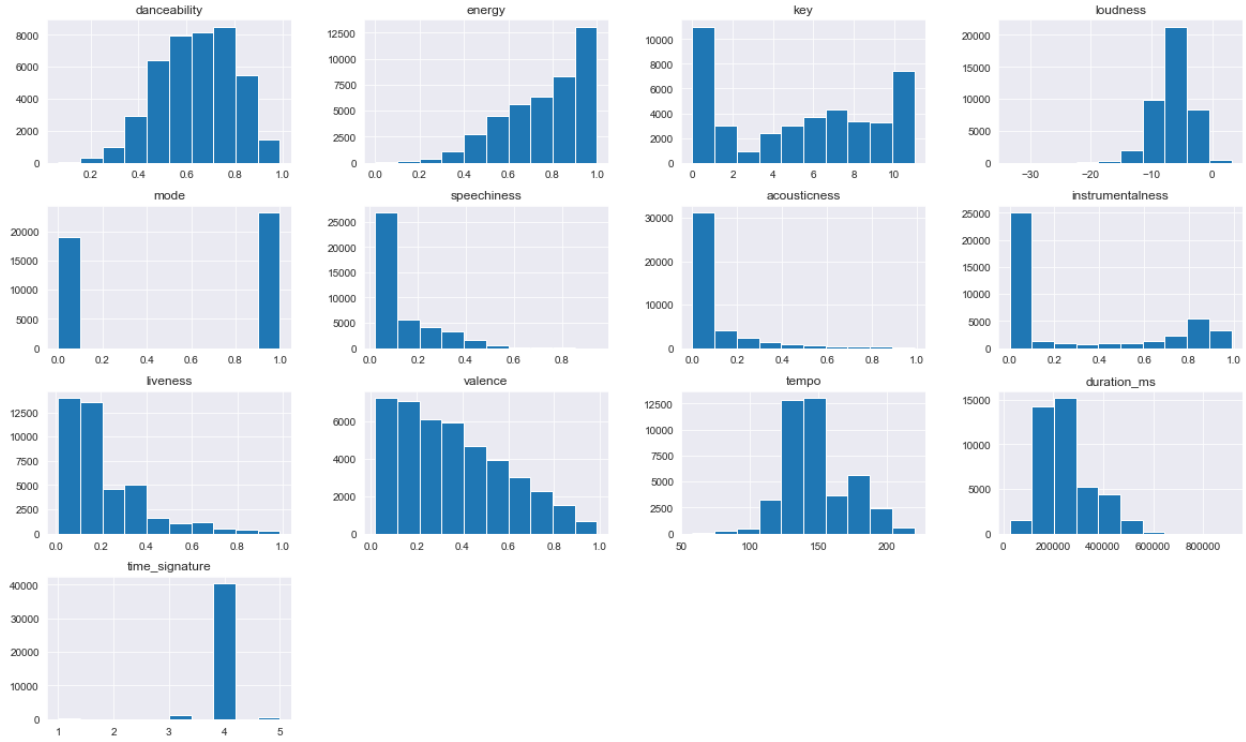
3. Metrics

The silhouette value is used as metric for clustering analysis as a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.

4. Data Exploration

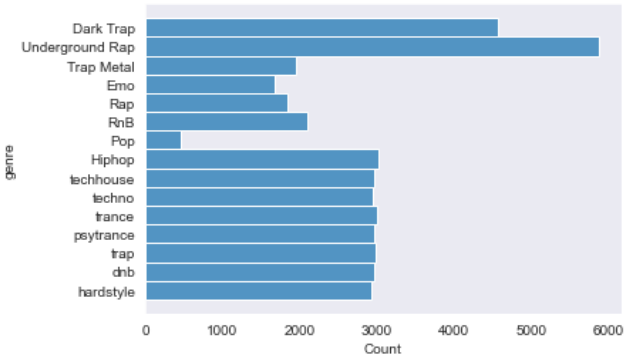
The Spotify's music dataset which is provided on Kaggle has valuable information and feature analyses about each music. This dataset has around 40,000 records with various types of fields which each record refers to a particular music available on Spotify. Each record has 21 attributes which are described completely on Spotify webpage. As it can be suspected based on these attributes, some of the features are useless or illegible, which means they must be discarded during the data preprocessing. Also, dataset has some null values in some of its' features which should be managed during preprocessing.

4.1. Exploratory Visualization



The above plot shows the distribution of numerical features in the dataset. As it can be seen, most of them have skewed distribution and should be normalized by an appropriate scaler during preprocessing section.

following plot shows that how songs are distributed over 15 different genres in the dataset. As it can be seen, this distribution is unbalanced. Underground rap is the most repeated genre in the dataset while pop is the least repeated one.



5. Methodology

5.1. Data Preprocessing

The preprocessing done in the notebook consist of the following steps:

1. Some of unrelated, unvaluable, and redundant columns or columns with null values in them were dropped. The resulted dataset has 14 columns and dropped columns list can be seen at the notebook.
2. Data types are fixed and each attribute convert to its' own datatype. Then each datatype down cast to another data type in order to decrease memory usage and faster ML models processing.
3. Normalization performed on the dataset using a min-max scaler.
4. Unrelated features eliminated using feature selection techniques. The performed technique was random forest classifier which trained with dataset to predict genre of each song and return importance of each score in that prediction. Features with low importance eliminated.
5. Genres encoded with one-hot-encoder to vectors and added to the original dataset.

5.2. Implementation

The implementation process can be split into two main stages:

1. The classifier training stage
2. Performing clustering algorithm
3. The application development stage

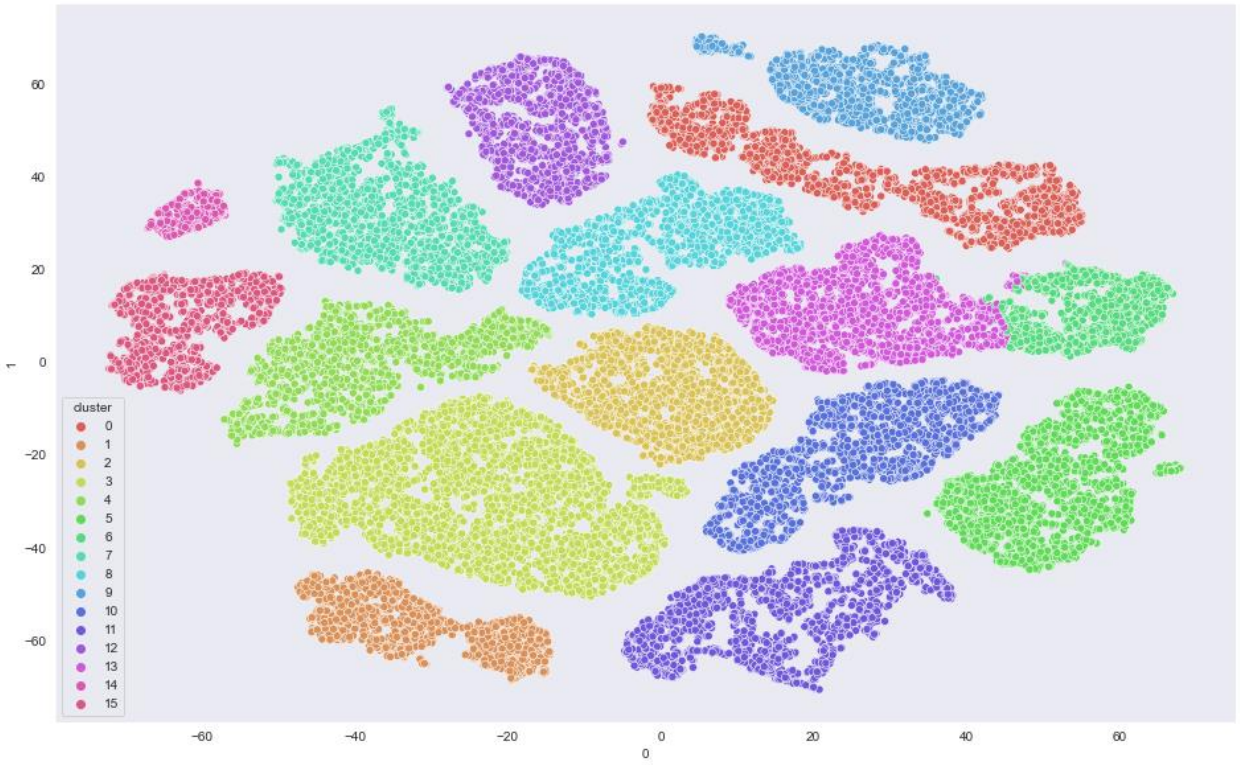
In the first stage, the classifier was trained on the dataset. This was done in a Jupyter notebook (titled "classify"), and can be further divided into following steps:

1. Load training dataset and label encoding genre for the classification.
2. Uncorrelated features to genres were eliminated.
3. Splitting dataset to train and test in order to performing cross validation.
4. Calculating accuracy which was around 77%.
5. Saving model and label encoder for further using in application.

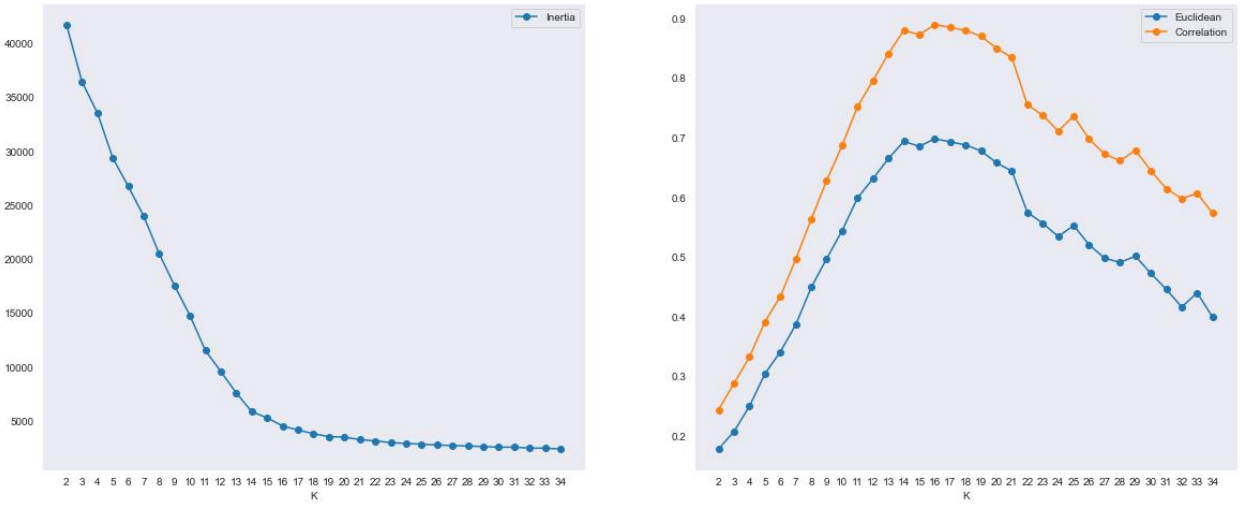
During the second stage, different clustering algorithms trained with the preprocessed dataset which was described previously and silhouette score for every algorithm calculated which is available in the following:

With considering silhouette score for each clustering algorithm and distribution of the dataset after clustering I decided to use "K-Means" as my main clustering algorithm which split data in the dataset into 16 clusters.

ALGORITHM	SILHOUETTE SCORE
K-Means	0.698
DBSCAN	0.654
Agglomerative	0.697
Birch	0.706
Mini Batch K-means	0.68



The above plot is t-SNE visualization of the dataset and each cluster is associated with a specific color. As it can be seen, the clustering algorithm can detect different clusters precisely.



The following plots are plotted in order to determine the optimal number of k (number of cluster). Left figure demonstrates inertia (measures how well a dataset was clustered) of each performed k-means algorithm with different number of clusters. Right figure demonstrates silhouette score using Euclidean and correlation metric of each performed k-means algorithm with different number of clusters. As it can be seen, for k=16, the k-means has the highest silhouette score and the “elbow” symbol perform with that number of clusters in left plot; therefore, k=16 is an appropriate number of clusters for k-means algorithm.

At the end, clustering algorithm, min-max scaler, and feature selector which were described in preprocessing section saved for further uses in the application. Also, clusters which achieved form our model added to the main dataset.

The application development stage can be split into the following steps:

1. User input dataset will be loaded and same preprocessing stage as the train dataset will be performed on them.
2. Since the user input dataset could be without genre, genre for every input song will be predicted using the classification model.
3. Clustering algorithm will predict cluster for each input song.
4. Top five cluster calculated for each user. (Clusters with highest number of repetitions in the input dataset. If size of resulted clusters were less than five then we could have some playlists with the same cluster.)
5. Five songs will be recommended to the user by saving their information in a CSV file. Those songs will be randomly recommended from the chosen clusters in the previous section.
6. All 25 songs which were recommended to the user via five different CSV file will be gathered and saved in a single CSV file (titled “mix_comp”).

6. References

[1]. https://en.wikipedia.org/wiki/Silhouette_(clustering)
[2]. https://towardsdatascience.com/feature-selection-techniques-in-machine-learning-with-python-f24e7da3f36e
[3]. https://machinelearningmastery.com/calculate-feature-importance-with-python/
[4]. **https://bpostance.github.io/posts/clustering-mixed-data/**
[5]. **https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html**
[6]. **https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/**
[7]. **https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/#:~:text=Feature%20selection%20is%20the%20process,the%20performance%20of%20the%20model.**
[8]. **https://machinelearningmastery.com/calculate-feature-importance-with-python/#:~:text=Feature%20importance%20refers%20to%20techniques,at%20predicting%20a%20target%20variable.&text=The%20role%20of%20feature%20importance%20in%20a%20predictive%20modeling%20problem.**
[9]. **https://towardsdatascience.com/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b**
https://machinelearningmastery.com/clustering-algorithms-with-python/
https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd

IMPORTANT NOTE ON COMPILING THE APPLICATION: because file "feature_selector.pkl" which is made in "final_2.ipynb" for selecting features is very huge to upload I deleted that file form folder "resources". So, before compiling "musicRecommender.py" please run the specific cell for feature selection task in "final_2.ipynb" and save the result via pickle (there is a pre-written cell for this purpose in the end of the notebook) and then compile the main application.