Music Classifier – Singer Identifier

G1-MusicDeciphers

**Data Science Capstone Project   
Data Acquisition and Pre-Processing Report**

Date:

11/02/2020

Team Members:

Likhil Kumar Rachuri - [lkr46@drexel.edu](mailto:lkr46@drexel.edu)

Nandakishor Reddy Pulagam - [np689@drexel.edu](mailto:np689@drexel.edu)

Nirupam Kumar Velagapudi – [nkv27@drexel.edu](mailto:nkv27@drexel.edu)

Soujanya Navuluru - [sn863@drexel.edu](mailto:sn863@drexel.edu)

Vuthej Krishna Reddy Verama Reddy - [vv334@drexel.edu](mailto:vv334@drexel.edu)

[The purpose of this report is to describe the data of your project. It includes three major sections: Data Sources, Data-Processing, and Appendix]

**Identifying Data**

**Data Sources:**

[Identify the data sources of your project. It may have more than one data source. Describe each of them and explain why you select the data sources.]

Initially, we have identified six different data sources to collect the data from and they are as follows.

* Jio Saavn – https://www.jiosaavn.com/
* Deezer – https://developers.deezer.com/api/explorer
* Audio DB – https://www.theaudiodb.com/api\_apply.php
* Last FM – https://www.last.fm/api
* Discogs – https://www.discogs.com/developers
* MixCloud – https://www.mixcloud.com/developers/

However, owing to various limitations as outlined below, we had to narrow done our data sources to two websites namely ‘Jio Saavn’ and ‘Deezer’.

Issue with Audio DB:

* Non-availability of the audio files (Song details were present but the audio file is not available)

Issue with Last FM:

* The requirement to have premium access to the websites

Issue with Discogs

* Retrieving preview audio files and very few details of the song rather than complete audio file and complete details of the song

Issue with MixCloud:

* Restrictions on the number of API calls for free accounts like for one day only 25 API requests can be made

Founded in 2007, Jio Saavn is an online music streaming service catering its services to users across the globe by having its headquarters in Mumbai (India) and New York (USA). Jio Saavn is available to the users on Android, Apple, and Microsoft Windows platforms apart from the option to use it as a web-application. It provides its users with songs in English and Indian regional languages. One does not need to have an account with ‘Jio Saavn’ in order to browse the songs from its mobile or web application. Reliance Industries Limited is the parent organization for Jio Saavn.

Being founded in 2007, Deezer is also an online music streaming service. With its headquarters in Paris (France), services are being provided to the users in more than 180 different countries. Similar to Jio Saavn, Deezer is also available on Android, Apple, and Microsoft Windows platforms and can also be accessed as a web-application. We need to create an account to access the songs and playlists on the Deezer website or mobile application.

Access Industries is the parent organization of Deezer.

Below are the reasons for selecting or finalizing on ‘Jio Saavn’ and ‘Deezer’ as our data sources.

* Ease of access to its API’s (Application Programming Interface) for collecting data related to the songs.
* It is very important to have a well-outlined API in order to provide useful responses while collecting the data related to songs. When we send a GET Response to its API,
* Availability of numerous features for each song such as ‘Song Name’, ‘Singer Name’, ‘Song Genre’, ‘Album Name’, ‘Album URL’, ‘Song URL’, ‘Release Year’, ‘Song Duration’, ‘Download Link’, ‘Song ID’ and many more attributes are being sent back as a response from its API.
* Availability of downloadable link for the audio file.

This is one of the major criteria used for deciding on the data sources. In order to build our datasets, we will be required to have the audio files in either MP3 or MP4 format and we faced minor hurdles which we were able to overcome with online resources in accessing them in ‘Jio Saavn’ and ‘Deezer’.

* Free access to songs across multiple languages without being restricted only to Premium Account.
* English and two other Indian regional languages (Hindi and Telugu) are our main focus while building our dataset and we were able to access all three languages through ‘Jio Saavn’.

**Acquisition Process:**

[Describe the data acquisition process. Is the dataset ready for download? How do you download the data? Do you need to write your own code to acquire the data from a public or private source? Describe how you do it. Are there multiple data sources? How do you integrate the data from multiple sources? Any other process involved in the acquisition process?]

The approach that is being followed to acquire data from ‘Jio Saavn’ and ‘Deezer’ are both similar. However, we had to follow some additional steps for ‘Deezer’ in order to register our code as an application and generate the ‘App ID’ along with ‘App Secret’ which are later used to connect to the API.

Below are the steps involved in the data acquisition process.

1. Base URL for the song details and the playlist details is used from Jio Saavn
2. Specific song URL or the Playlist URL is then added to the base URL
3. Song details and Playlist details are extracted in the JSON format
4. The JSON response is looped through to extract relevant features
5. URL to the Audio file is used from this JSON response to download the file
6. Relevant features along with the audio file link form the .CSV file
7. MP3/MP4 format of the audio file is converted to .WAV format for further extraction and modelling.

Downloading Procedure using a script:

While collecting the data from Deezer and Jio Saavn, we have also included the url of the song/preview for download. Few songs are only having preview and used the same in our dataset. The url collected is different for each song by using which we are downloading the mp3 or mp4 format of the preview and song respectively. The downloading of the mp3/mp4 from the link is done using a requests package and url retrieved package from python. Once these urls obtained from above are provided into this request package, obtained mp3/mp4 files are stored in the different folders for convenience at a later stage.

Code used while extraction:

API for Deezer is straight forward and we are having our own code to connect to the API and to extract the data. For ‘Jio Saavn’, we had to develop our own code for the API as there isn’t any official API provided by them. Because of the absence of a well-documented API, **we have leveraged the help from the GitHub community** to build one.

Description

Initially, we get the base url for the Deezer and Jiosaavn, this base url contains the information regarding specific playlists that are required or interested. This base url is the home for our requirement, from this, we are further interested in fetching details for each individual song, to obtain the same, we loop over the songs present and add each song specific url at the end of the base url so it forms the complete url for a song.

From this complete url, we obtain the song details and playlist details in a JSON format, there are many features present in this extraction which we are not interested in, so further we implemented a filter which loops over this JSON and then extracts only the relevant features according to our requirement. We are also extracting the url to download the mp3 file as well at this stage. With all these features including the name, singer, url a csv file is formed. Further to this, we are extracting audio as mentioned above and storing all mp3 files separately. Individually a csv file is obtained from each source, merging all this creates a final csv file.

Handling multiple sources:

Since we are having two different data sources, we need to perform an extra step to integrate the data. To do this, we are following a uniform structure while building the .csv files for both ‘Jio Saavn’ and ‘Deezer’ which will ease the operation for integration.

By the end of acquisition, we are generating a final csv file with all features from the source, and also a folder with all the audio files downloaded using a script.

We were able to create Google notebooks through Google VM and are working towards storing the data in Google cloud block storage so that the data would be easily accessible for further works.

**Issues:**

[Are there any potential issues in data acquisition that have not be solved yet?]

* The main potential issue that we faced is in obtaining a complete song from Deezer APIs. We are only getting a preview of audio file of 30 seconds for the songs from this website but, we are able to get complete audio file from Jio Saavn

**Data-Processing**

[Examine the data you have acquired and understand the data properties. Is there any pre-processing you need to do before you can start analyzing the data? For example, missing data, sparsity, noise, veracity, ambiguity, interoperability, etc. Describe each data issue in a sub-section and explain how you clean up the data.]

Handling different features from different sources:

Data sources are more than one, each individually giving their respective attributes. All these data sources are merged and only a few important features are selected from them which includes ‘Song Name’, ‘Singer Name’, ‘Song Genre’, ‘Album Name’, ‘Album URL’, ‘Song URL’, ‘Release Year’, ‘Song Duration’, ‘Download Link’, ‘Song ID’.

All the above features are common in all individual datasets.

From the data obtained above using the extractors and the end goal that is required for our project, we decided to build two different datasets one for the singer identification and the other for the genre identification.

These two datasets will have exactly similar features, but the songs used in both the datasets are entirely different. No song is used for both the datasets and splitted the songs with precision. The reason for doing this is we donot have a singer representing in all genres and it was making complex. So, we entirely removed these inter dependencies and made two sets with different songs.

Converting Mp3/Mp4 to Wav extension:

The audio data available is in the form of mp3 or mp4 versions from various sources, to make it uniform, and to extract all the features from the song, it needs to be converted into (.wav) format. Converting to wav through python had created an issue for the FFmpeg package as this package has decommissioned recently. Implemented the functionality to convert to wav using av package from R.

5-Sec window approach:

After the conversion to wav format, than giving the entire song to the model we are thinking to split the song with 5 sec interval, this splitting gives more records as output and gives intrinsic level of understanding for the song. This also gives us more records as one song will split itself into many sub sets, and considering all these into the modelling will have more chances of accurate modelling.

Splitting Song file to separate vocals and music:

Song audio is been split into vocal and musical parts separately using the spleeter package from python. This package accepts the song and divides it into voice and music. There were options to divide the music into various frequencies, but they were not accurate, so we have limited the input file to split into two parts of frequencies.

The above splitting functionality is applied onto two datasets created but for further use we only considered the musical part of the genre dataset and vocals part of the artists dataset.

Features Extracted:

spectral contrast: Spectral contrast is defined as the difference of peaks and valleys in the spectrum. Mean and variance for all the values are collected.

spectral bandwidth: It is the range of values in which the signal is not less than the half of its maximum value. Mean and variance of all the values are collected from here.

spectral centroids: It is a metric in digital signal processing, where spectral centroids are defined as the points where the center of mass for the spectrum is located. Mean and variance are collected for all the values.

spectral rolloff: Spectral rolloff is the range where in 85% of the spectrum energy falls into. Mean and variance are collected.

spectral flatness: It measures the amount of noise that is present in the input audio rather than the presence of toned data. Mean and variance are collected for all the values.

zero crossings: It is defined as the number of times the signal in audio is crossing the zero line.

Mel-Frequency Cepstral Coefficients  (20) features: Mel-Frequency Cepstrum is a combination of many Mel-Frequency Cepstral Coefficients (in short MFCC’s) derived from the non-linear spectrum of spectrums. Mean and variances are considered for all the values individually and 20 such features are extracted.

Chroma Short-Time Fourier Transform (24 chroma stft features) : Chroma feature describes the presence of tonal content in the audio data by classifying input into twelve different pitch classes. Mean and variances are considered for all the values individually and 24 such features are extracted.

Chroma Constant-Q Transform (24 chroma cqt features):The frequency obtained by the Constant-Q Transform directly corresponds to the central frequencies of the musical notes. Mean and variances are considered for all the values individually and 24 such features are extracted.

Chroma Energy Normalized Statistics (24 chroma cens features): This is another variant of the chromagram feature and is robust to dynamics, timbre and articulation. Mean and variances are considered for all the values individually and 24 such features are extracted.

Music file Processing for Genre Dataset:

The music file obtained is in the wav format from spleeter. To extract the important features from this file, we used the librosa library from python and fetched the spectral centroids, spectral rollover, zero crossings, mfcc (20) features, chroma stft (24 features comprising means and variances), chroma cqt (24 features comprising means and variances), chroma cens(24 features comprising means and variances) and beat(tempo). These are further normalized using a librosa.util.normalize normalizer to restrict the values between -1 and 1. This was the better standardization technique in comparision with the other standardization or normalization like standard analyzer. For each of the above-mentioned features mean and variance are considered as features for our dataset. One csv file for each dataset is generated with the number of songs vs 126 features including target variable such as genre for this dataset.

Vocal file Processing for Artist Dataset:

The vocals file obtained is in the wav format from spleeter. To extract the important features from this file, we used the librosa library from python and fetched the spectral centroids, spectral rollover, zero crossings, mfcc (20) features, chroma stft (24 features comprising means and variances), chroma cqt (24 features comprising means and variances), chroma cens(24 features comprising means and variances) and beat(tempo). All these are further normalized using a min-max scaler to restrict the values between -1 and 1. For each of the above-mentioned features mean and variance are considered as features for our dataset. One csv file for each dataset is generated with the number of songs vs 126 features including target variable such as artist for this dataset.

The preprocessing for music file and vocal file are almost similar but the end goal is to give two datasets which represent respective songs.

Table of Contributions

The table below identifies contributors to various sections of this document.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Section** | **Writing** | **Editing** |
| **1** | **Data Sources** | **All Team Members** | **All Team Members** |
| **2** | **Data Pre-Processing** | **All Team Members** | **All Team Members** |
| **3** | **Appendix** | **All Team Members** | **All Team Members** |