Music Genre Classification and Recommendation System

Pranav Kakkad and Neha Cholera
Northeastern University

Abstract

The goal of this project was to add the new songs to the clusters that have highest similarity and based on clustered songs, the user is recommended songs based on if the user liked that song and on the user history. We applied various methods to complete the task like k-means to cluster the songs and methods like SVD, BaselineOnly method in collaborative and cosine similarity on top tf-idf, word to vector, LDA for content based similarity.

1. Introduction

As the quantity of music being released on a daily basis continues to sky-rocket, especially on internet platforms such as Soundcloud and Spotify – a 2016 number suggests that tens of thousands of songs were released every month on Spotify – the need for accurate meta-data required for database management and search/storage purposes climbs in proportion. Being able to instantly classify songs in any given playlist or library by genre is an important functionality for any music streaming/purchasing service, and the capacity for statistical analysis that correct and complete labeling of music and audio provides is essentially limitless.

It is not possible for anyone to listen to all songs and decide whether they will like the song or not and therefore to increase the market size it is very necessary to display songs to user that they might like thus increasing the revenue for the producer of the song as well it is necessary for the platform like spotify,pandora, apple to make sure that customer stick to their platform, making good recommendation apart from the good quality audio is one of the reason that makes sure that customer comes back.

In this project we have tried to create a whole pipeline right from when the new song is added, which user will like that song and if that user likes that song what are the new possible songs that the user can like.

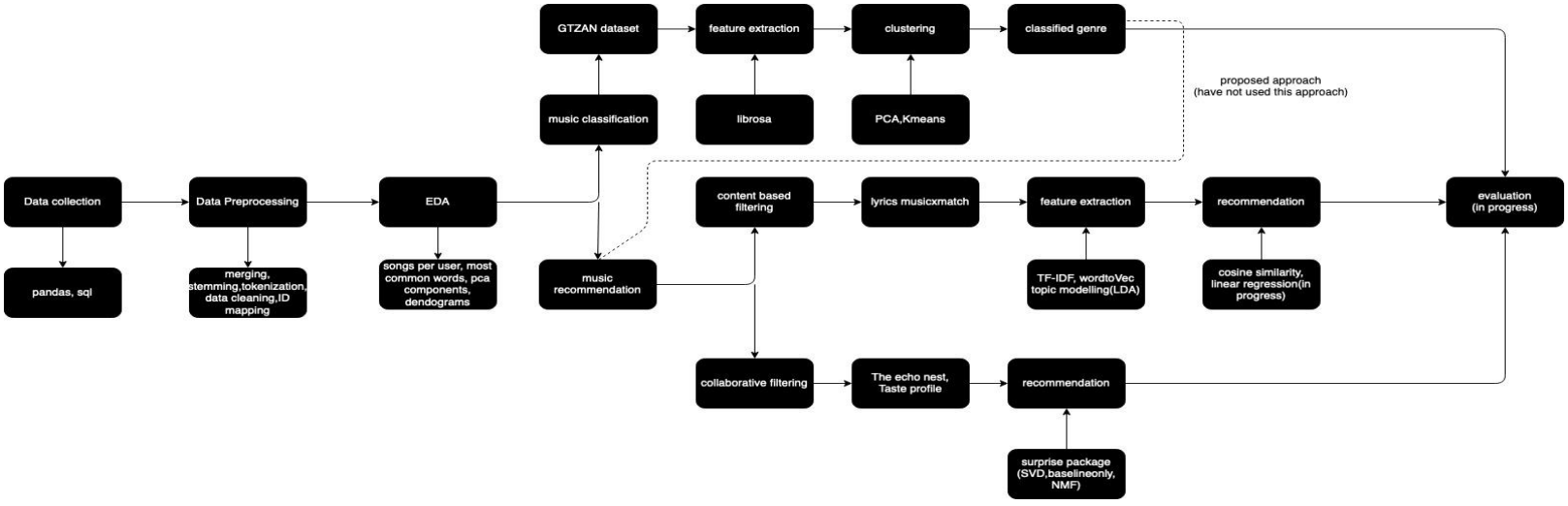


Fig 1: Proposed approach

2. Dataset and Features

Mimicking the whole pipeline was difficult for us because we're not able to find the dataset that contains the audio files as well as the user listening history, therefore we tried to merge two dataset to complete our task.

2.1 GTZAN dataset

It contains 1000 audio files, 100 audio files, 30 seconds long pertaining to 10 genres which are: Blue, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, Rock. The given dataset has 3 folders:

1. **Genre original:** it contains 10 folders for 10 different genres each containing 100 songs.
2. **Images originals:** it contains visual representation of each song, we can classify songs based on image by using Neural network.
   Note: we have not used this and have created images on our own thinking about the future use, but have not used as of now.
3. **2 CSV files:** Each CSV file contains the audio features of each song in the form of mean and variance of each feature of the song. Both CSV files have the same structure but the only difference is one contains features after every 30 seconds and other contains features after every 3 seconds making our data 10 times in size.
   Note: we have not used these csv files and have extracted the features on our own using librosa to make sure that whenever we find the desired dataset we can directly plug into our code and see the working output.

```
feature_30.head()
```

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_mean	spectral
0	blues.00000.wav	661794	0.350088	0.088757	0.130228	0.002827	1784.165850	129774.064525	2002.449060	
1	blues.00001.wav	661794	0.340914	0.094980	0.095948	0.002373	1530.176679	375850.073649	2039.036516	
2	blues.00002.wav	661794	0.363637	0.085275	0.175570	0.002746	1552.811865	156467.643368	1747.702312	
3	blues.00003.wav	661794	0.404785	0.093999	0.141093	0.006346	1070.106615	184355.942417	1596.412872	
4	blues.00004.wav	661794	0.308526	0.087841	0.091529	0.002303	1835.004266	343399.939274	1748.172116	

Fig 2: visual of partial feature dataframe

```
mean_feature_dataframe.columns
```

```
Index(['chroma_stft_mean_0', 'chroma_stft_mean_1', 'chroma_stft_mean_10',
       'chroma_stft_mean_11', 'chroma_stft_mean_2', 'chroma_stft_mean_3',
       'chroma_stft_mean_4', 'chroma_stft_mean_5', 'chroma_stft_mean_6',
       'chroma_stft_mean_7', 'chroma_stft_mean_8', 'chroma_stft_mean_9',
       'chroma_stft_std_0', 'chroma_stft_std_1', 'chroma_stft_std_10',
       'chroma_stft_std_11', 'chroma_stft_std_2', 'chroma_stft_std_3',
       'chroma_stft_std_4', 'chroma_stft_std_5', 'chroma_stft_std_6',
       'chroma_stft_std_7', 'chroma_stft_std_8', 'chroma_stft_std_9',
       'kurtosis', 'mean', 'mfcc_mean_0', 'mfcc_mean_1', 'mfcc_mean_10',
       'mfcc_mean_11', 'mfcc_mean_12', 'mfcc_mean_13', 'mfcc_mean_14',
       'mfcc_mean_15', 'mfcc_mean_16', 'mfcc_mean_17', 'mfcc_mean_18',
       'mfcc_mean_19', 'mfcc_mean_2', 'mfcc_mean_3', 'mfcc_mean_4',
       'mfcc_mean_5', 'mfcc_mean_6', 'mfcc_mean_7', 'mfcc_mean_8',
       'mfcc_mean_9', 'mfcc_std_0', 'mfcc_std_1', 'mfcc_std_10', 'mfcc_std_11',
       'mfcc_std_12', 'mfcc_std_13', 'mfcc_std_14', 'mfcc_std_15',
       'mfcc_std_16', 'mfcc_std_17', 'mfcc_std_18', 'mfcc_std_19',
       'mfcc_std_2', 'mfcc_std_3', 'mfcc_std_4', 'mfcc_std_5', 'mfcc_std_6',
       'mfcc_std_7', 'mfcc_std_8', 'mfcc_std_9', 'name', 'rmse_mean',
       'rmse_std', 'skewness', 'spectral_bandwidth_mean',
       'spectral_bandwidth_std', 'spectral_centroid_mean',
       'spectral_centroid_std', 'spectral_rolloff_mean',
       'spectral_rolloff_std', 'std', 'zero_crossing_rate',
       'zero_crossing_rate_mean'],
      dtype='object')
```

Fig 3: columns in features dataframe

2.2 Million Song dataset

It is a collaborative project between The Echo Nest and LabROSA, is a freely available collection of audio features and meta data for a million contemporary popular music tracks.

We used mainly three sub-dataset :

1. Taste Profile: We used a subset of 10k unique songs and 10k unique users that contain user-song-play count triplets and are tab delimited and thus contain a listening history of each user.

```
triplets.head()
```

	user_id	song_id	count
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1

Fig 4: Taste profile dataset

2. The Million Song subset: This data set contains information about 10k unique songs. It contains information in terms of fields like track id, song id, artist name and song title

```
song_df.head()
```

	song_id	title	release	artist_name	year
0	SOQMMHC12AB0180CB8	Silent Night	Monster Ballads X-Mas	Faster Pussy cat	2003
1	SOVFVAK12A8C1350D9	Tanssi vaan	Karkuteillä	Karkkiautomaatti	1995
2	SOGTUKN12AB017F4F1	No One Could Ever	Butter	Hudson Mohawke	2006
3	SOBNYVR12A8C13558C	Si Vos Querés	De Culo	Yerba Brava	2003
4	SOHSBXH12A8C13B0DF	Tangle Of Aspens	Rene Ablaze Presents Winter Sessions	Der Mystic	0

Fig 5: Million song subset dataset.

3. The musiXmatch dataset: The MXM dataset provides lyrics for many MSD tracks. The lyrics come in bag-of-words format: each track is described as the word-counts for a dictionary of the top 5,000 words across the set. The dataset comes in two text files, describing training and test sets. There are 210,519 training bag-of-words, 27,143 testing ones. It also provides the full list of words with total counts across all tracks so you can measure the relative importance of the top 5,000.

The two text files are formatted as follow (per line):

- comment, ignore

%word1,word2,... - list of top words, in popularity order

TID,MXMID,idx:cnt,idx:cnt,... - track ID from MSD, track ID from musiXmatch,

then word index : word count (word index starts at 1!)

There is another dataset called mxm_779k_matches related to this. It matches the musiXmatch IDs to our superset MSD IDs. The musiXmatch team was able to resolve over **77%** of the MSD tracks. Of these, they have released lyrics for 237,662 tracks

	track_id	mxm_tid	word	count	is_test
0	TRAABJS128F9325C99	5516210	i	5	0
1	TRAABJS128F9325C99	5516210	the	3	0
2	TRAABJS128F9325C99	5516210	you	3	0
3	TRAABJS128F9325C99	5516210	to	3	0
4	TRAABJS128F9325C99	5516210	and	8	0

Fig 6: lyrics dataset

```
meta_data.head()
```

	track_id	title	song_id	release	artist_id	artist_mbid	artist_name	duration	artist_familiarity	artist_hotttnesss
0	TRMMMYQ128F932D901	Silent Night	SOQMMHC12AB0180CB8	Monster Ballads X-Mas	ARYZTJS1187B98C555	357ff05d-848a-44cf-b608-cb34b5701ae5	Faster Pussy cat	252.05506	0.649822	0.394032
1	TRMMMKD128F425225D	Tanssi vaan	SOVFVAK12A8C1350D9	Karkuteillä	ARMVN3U1187FB3A1EB	8d7ef530-a6fd-4f8f-b2e2-74aec765e0f9	Karkkiautomaatti	156.55138	0.439604	0.356992
2	TRMMMRX128F93187D9	No One Could Ever	SOGTUKN12AB017F4F1	Butter	ARGEKB01187FB50750	3d403d44-36ce-465c-ad43-ae877e65adc4	Hudson Mohawke	138.97098	0.643681	0.437504
3	TRMMMCH128F425532C	Si Vos Querés	SOBNYVR12A8C13558C	De Culo	ARNWYLR1187B9B2F9C	12be7648-7094-495f-90e6-df4189d68615	Yerba Brava	145.05751	0.448501	0.372349
4	TRMMMWA128F426B589	Tangle Of Aspens	SOHSBXH12A8C13B0DF	Rene Ablaze Presents Winter Sessions	AREQDTE1269FB37231		Der Mystic	514.29832	0.000000	0.000000

Fig 7: track metadata(partial columns)

3. Classification

The song is first converted into an audio time series with a sampling rate of 22050 Hz which is the default sampling rate in librosa. As seen in the figure 8, audio is converted into an array of values which can be plotted in the time series domain as follows.

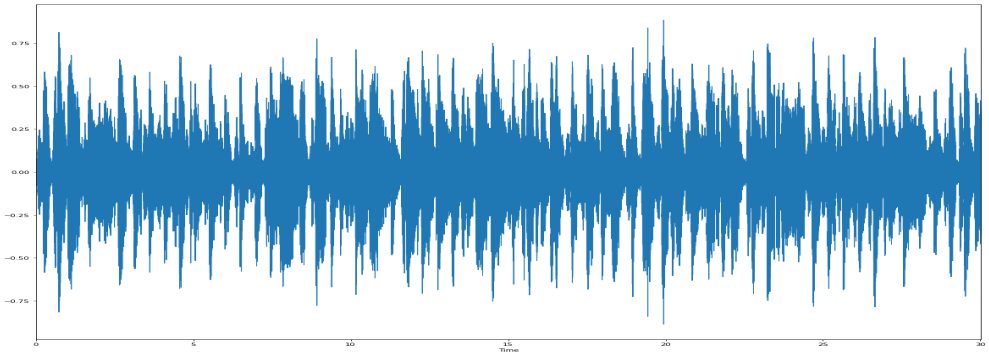


Fig 8: waveplot of audio file from time series

Following are the features that are extracted from the audio:

1. Skewness : It is the degree of distortion present in the audio, basically it measures the symmetry in the audio signal in our case.

if skewness = 0: normally distributed

skewness > 0: more weight in left tail

skewness < 0: more weight in right tail

2. Kurtosis: It is basically used for tail distribution, higher the value of kurtosis it means that there is higher distribution towards its tails which indicates that there are high chances of having a large number of outliers, low value of kurtosis is also a concern because then model will be sensitive to outliers, so need to remove them also if present.

In our case we have acceptable values of kurtosis for all audio.

3. Spectral centroid: It is associated with the measure of brightness of the sound which is obtained by evaluating the center of gravity using the fourier transform. Librosa processes each frame of magnitude spectrogram and treats them as distribution over frequency bins, from which the mean is extracted. It returns a numpy array of centroid frequency,which is then converted by us to a singular value of mean and standard deviation. Centroid is calculated for each time interval t using: centroid[t] = sum_k S[k, t] * freq[k] / (sum_j S[j, t]) where S is a magnitude spectrum and freq is an array of frequencies.

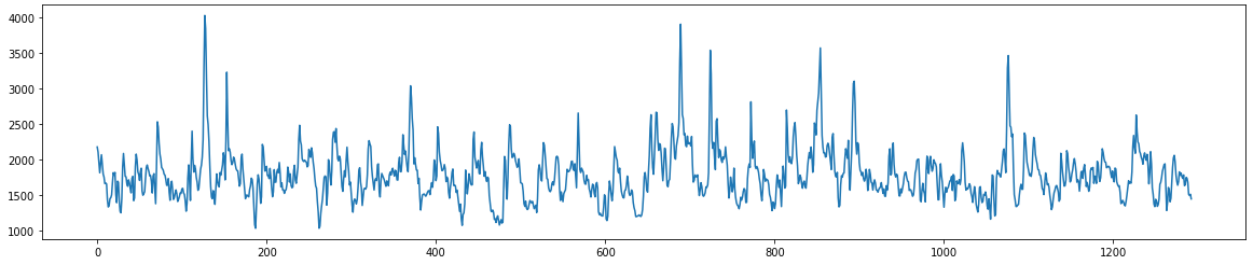


Fig 9: Spectral centroid audio plot

4. Root Mean square (RMS): It is used to calculate the average of values over a period of time. Square root of squared amplitude over the time is being calculated. Librosa can calculate rms value from the time series representation or from spectrogram, both have their pros and cons, calculating from time series is faster but less accurate while calculating from spectrogram is more accurate but is time consuming because it requires calculation of short time fourier transformation is required in this case.

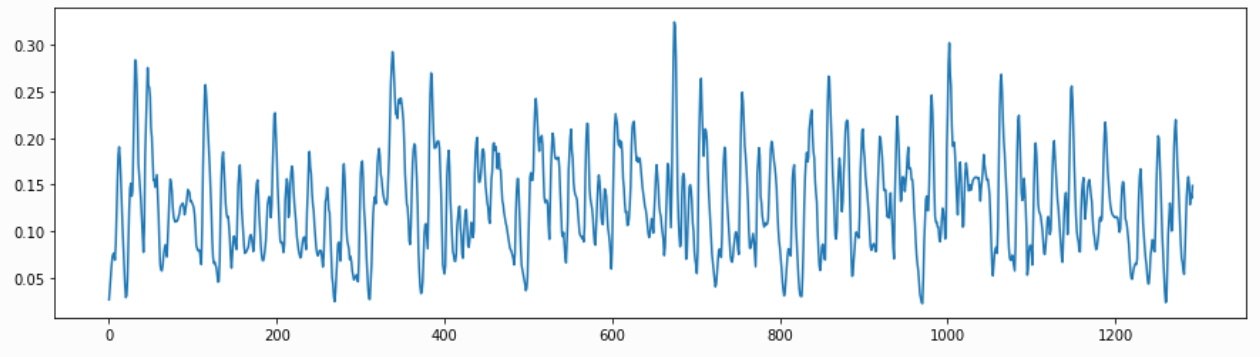


Fig 10: RMS audio plot

5. Spectral rolloff: It is defined for each frame for a spectrogram in a way that at least 85% of the energy of the spectrum in a given frame is contained in that bin and below its bin.
This function returns roll-off frequency for each frame in the form of numpy array of shape (1,t)

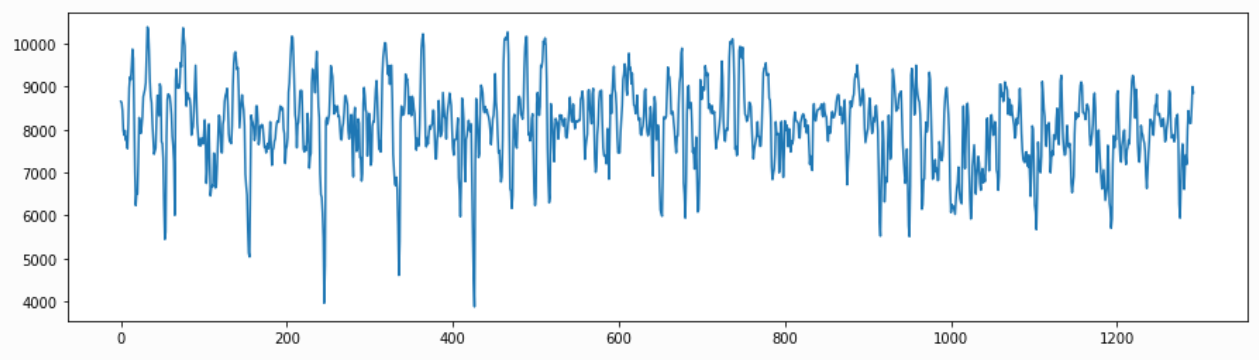


Fig 11: Spectral roll of plot

6. Spectral bandwidth: It can be defined as the extent of power transfer function around the center frequency. It is calculated using this formula: $(\sum kS(k)(f(k) - fc)p)^{1/p}$.Keeping the value of p as 2 will be the same as calculating the weighted standard deviation.

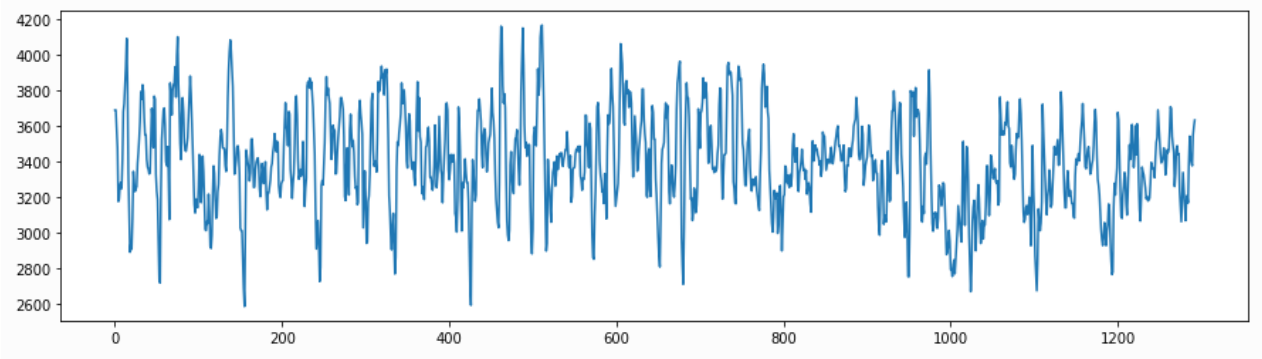


Fig 12: Spectral bandwidth audio plot

7. Zero crossing rate: It is a way to identify the smoothness of the given audio, zero crossing rate is calculated by comparing the sign of each consecutive pair of signals. It returns the numpy array of (0,t).

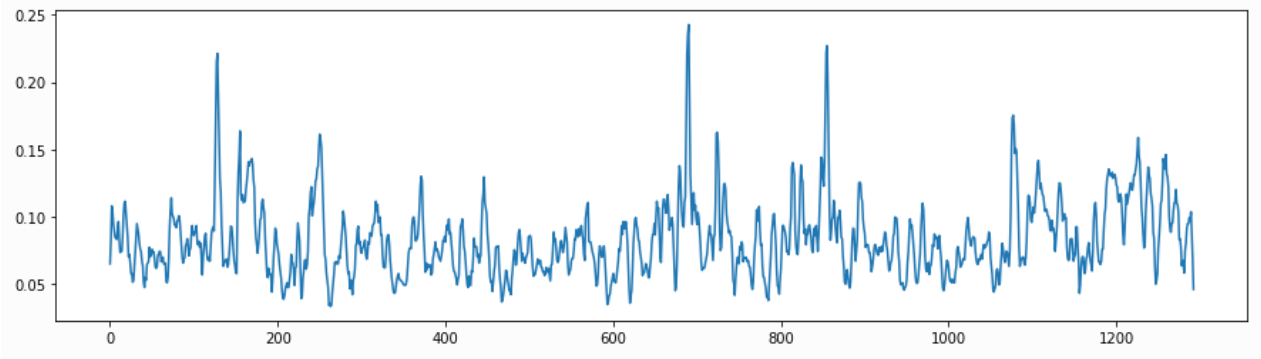


Fig 13: Zero crossing rate plot

8. Mel-frequency cepstral coefficients (MFCCs): it is representation of the short term power spectrum of sound, which is based upon linear cosine transform of a nonlinear mel scale of frequency. MFCC

are the coefficients that make up the mel frequency spectrum. It returns the numpy sequence coefficients.

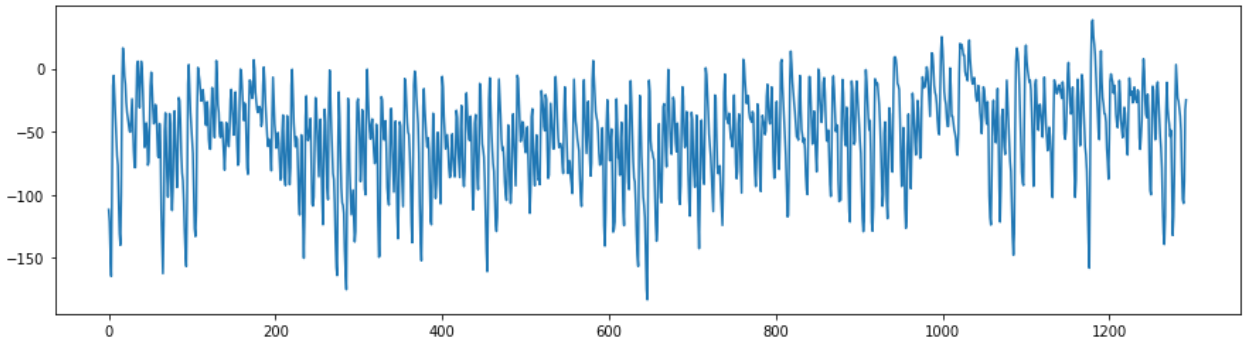


Fig 14: one of the component of mfcc audio plot

9. Chroma STFT: Compute a chromagram from a waveform or power spectrogram. The function returns normalised energy for each chroma bin at each frame.

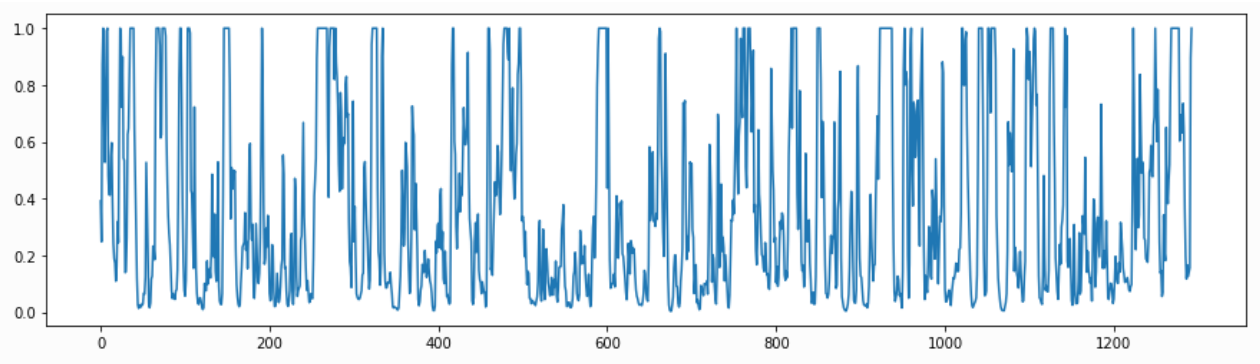


Fig 15: one of the components of Chroma STFT audio plot.

3.1 Methodology

This section gives the insights about the preprocessing of data as well as the clustering techniques that are used to cluster having similar features.

Dataset

The given dataset is labelled dataset classifying each song in specific genre, but as we are using the unsupervised machine learning methods we have jumbled songs and now there is no clear specification in csv file about which song belongs to which cluster.

Feature Extraction

We have extracted a total of 79 different features which are used to classify songs based on its cluster. For each feature that is defined above, mean and standard deviation is being taken for the numpy array that librosa function returns, this is done to make sure that we feed in the singular value to the model and not the whole array to reduce the complexity of code as well as the computational complexity. Mean and standard deviation for each feature also generalises the whole array for us which can be used to easily understand the whole waveplot by single value.

Kurtosis and skewness are also considered to see whether there are any abnormalities found or not but in our dataset we do not observe anything that is unusual and therefore we have not taken into consideration those 2 parameters while predicting.

Correlation

There are 2 possibility that can be performed, either correlation between mean of all features could be compared or either standard deviation between all the features could be compared to see how are they related to each other, we have tried both of them and found that mean makes more sense as it is the average value of whole time series of audio signal while according to definition of standard deviation which states that it is measure of deviation present in the time series, is not much useful while comparing the correlation as we are concerned with the value right now and not in how much standard deviation is present in each song. Therefore correlation between mean of each feature is taken and heat map is being plotted.

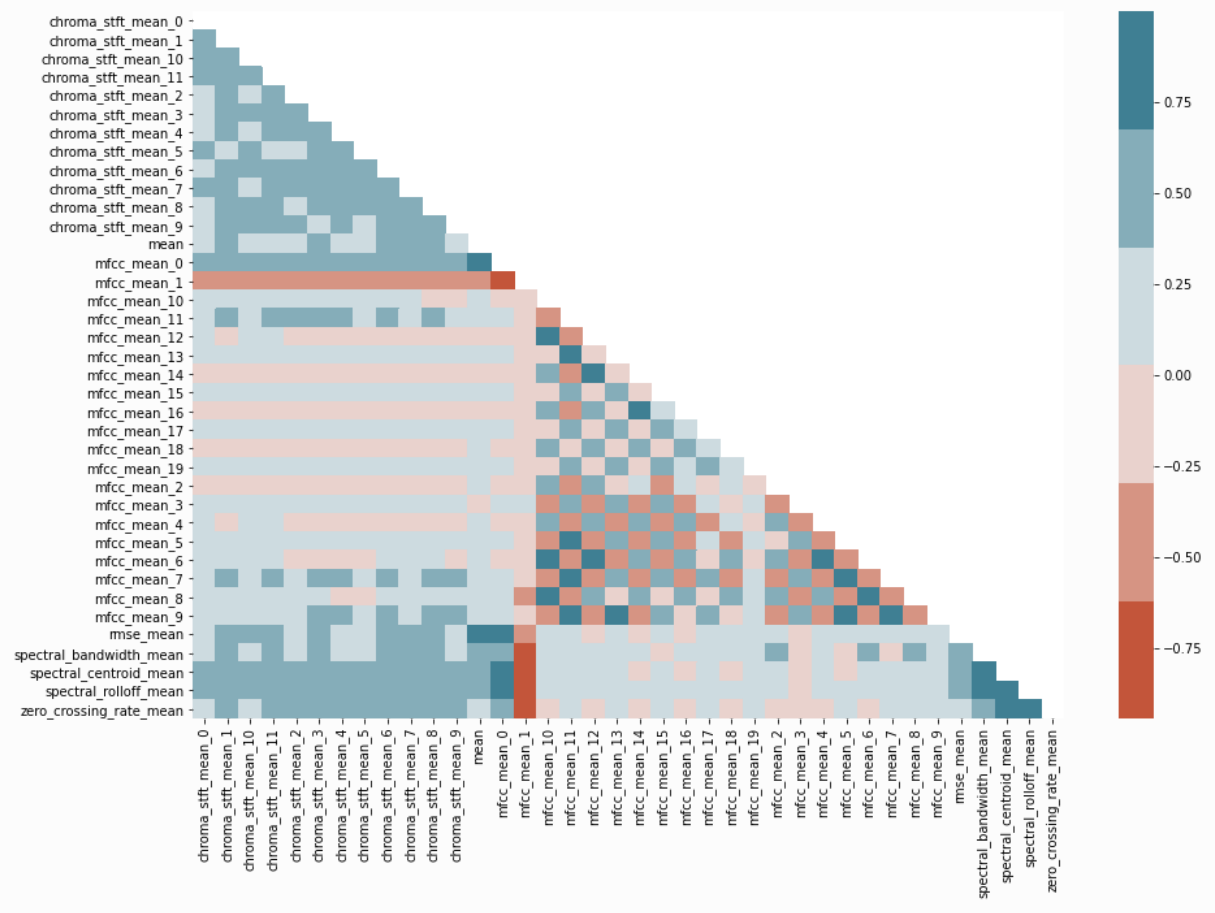


Fig 16: correlation between different features extracted from audio

PCA

From the correlation we have found that every variable does not have strong impact on the other feature present in the dataset and 79 features is large dimension for each song therefore we have performed dimensionality reduction on the feature dataset extracted from song, even though PCA performs normalisation, we have manually normalised the data for our understanding how is that thing done and have dropped column, to make sure that there are only columns having numerical type in it, after the normalising of data is performed it was necessary to make sure that maximum variance is covered using less amount of features and after plotting the cumsum of variance after each feature we found from the graph that if number of components were kept to 30 we were able to regain more than 95% of data, as observed from the graph and text data even after increasing more features there is very less difference in variance covered.

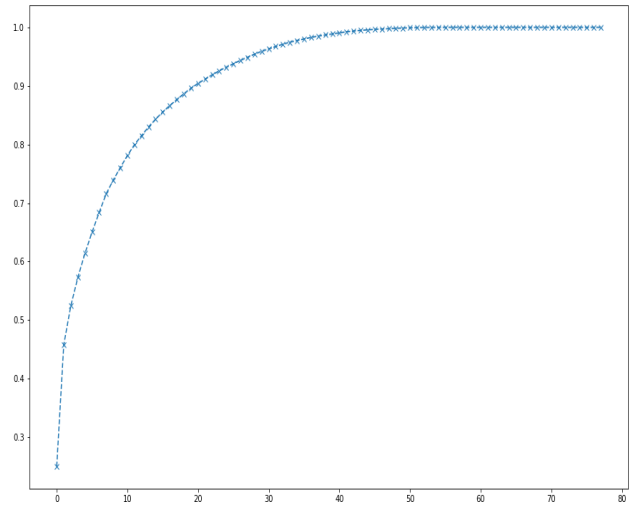


Fig 17: curve representing the cumulative sum of variance covered

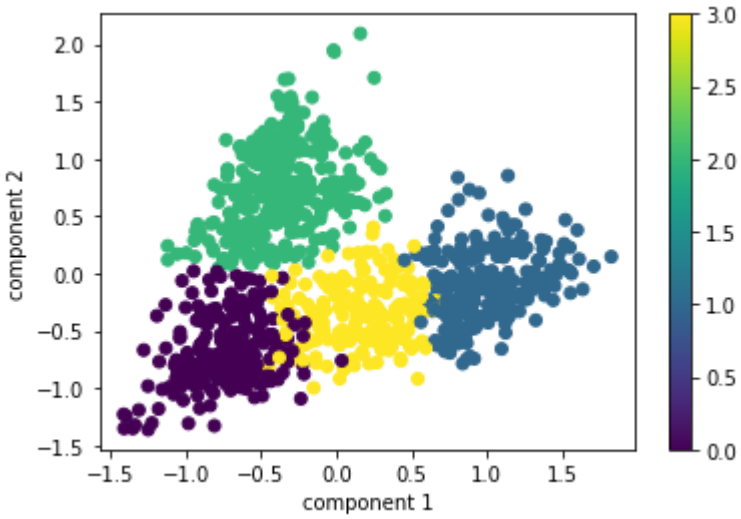


Fig 18: Clustering based on top 2 PCA components

Clustering

We now have 30 features reduced from originally 79 features, clustering is now performed on scaled PCA features that are extracted. We have clustered the song using k means and dendogram method.

For K-means clustering techniques, it was important to decide how many clusters should be performed though we knew that there were 10 genres present, but still we had a loop ranging from 2 cluster to 20 clusters and from the elbow method performed using kneed library we found that optimal number of clusters were 4, that made us ponder upon the reason and after a research we found that there are multiple reasons for this, first being there is lot of similarities between features of different genres for example rock, pop and hip hop have features that are almost similar to each other and on other hand classical and jazz were having features similar to each other and k means being a low level of algorithm compared to CNN was not able to capture this minute details found in different song genre and classified them into the single genre. Therefore we performed the clustering based on the optimal number of clusters as well as on 10 clusters, and evaluated the silhouette coefficients, in fact we calculated silhouette coefficients for clusters 2 to 10 which is as shown below.

If we plot the 2 main components of PCA, that are the first 2 components, we can see that 4 clusters are the optimal way to cluster the audio present in the GTZAN dataset. To cross check whether our hypothesis is true or not we performed clustering using dendogram by applying the ward method so that standard deviation is measured for each audio feature and clusters are formed on the basis of it, even in that we found it is not able to cluster into 10 genres but only 3 clusters.

We can see from the silhouette coefficients also that the highest value is found when the number of clusters are 3 and not 10, though in K means we have an optimal number of clusters as 4, silhouette coefficients are low for it compared to 3 clusters. In one of the dendrogram graphs we have calculated the number of songs that are clusters into that cluster.

Conclusion: K means is very basic algorithm that is used to cluster the song, other methods like CNN can be used on the features as well on the mel spectrogram images to classify the songs into clusters

and they would perform better than the k means as they will be able to extract indepth pattern between each feature and will be able to differentiate between 2 genres even if there very less difference between features.

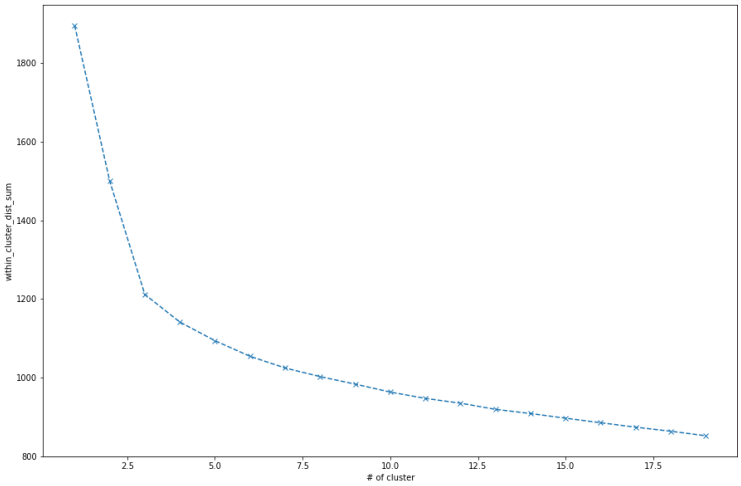


Fig 19: elbow method to find the optimal number of cluster

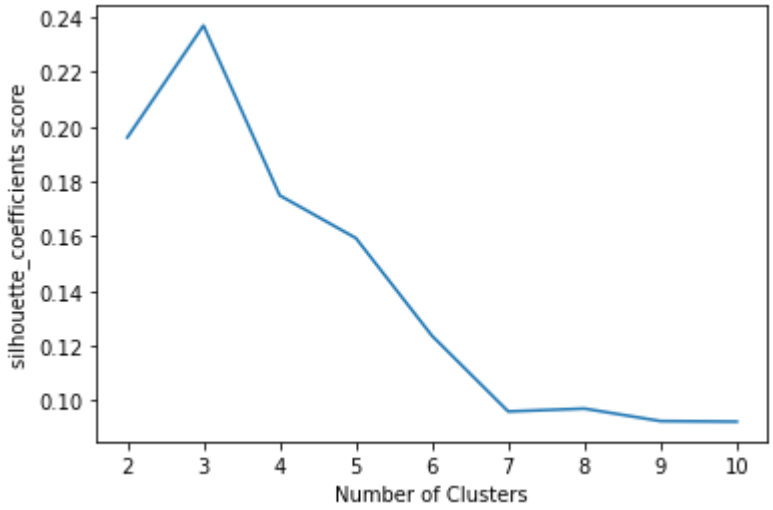


Fig 20: Evaluation score for size of cluster

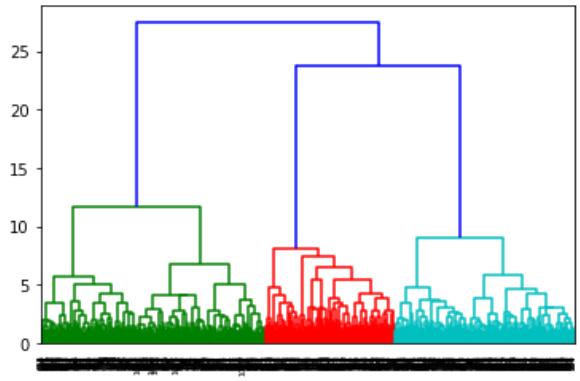


Fig 22: Clustering based on dendrograms

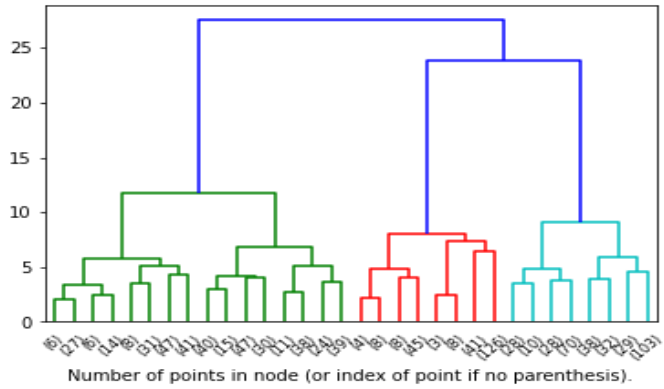


Fig 23: dendrograms representing number of points in each cluster

	clusters	silhouette_coefficients
0	2	0.196134
1	3	0.237020
2	4	0.174958
3	5	0.159349
4	6	0.123633
5	7	0.095989
6	8	0.097032
7	9	0.092469
8	10	0.092269

Fig 21: silhouette coefficient for given cluster size

4. Recommendation

Basically a recommendation system is the system that understands the users past history or object structure so that it can recommend on the basis of which it has learned, in practical case it never stops learning and sees user behavior at each and every point but for the sake of project and from the data which we had we have only considered the past history not considering whether the user iked our new recommended song or not but in real life case, for every recommendation that the model makes, if the user liked the song model has higher confidence and will recommend more song based on what it is able to find from the song features but in case if user dislikes the song, model is punished and would not be allowed to recommend those type of song until model have higher threshold of confidence but in our case we have not gone up to such an extent and kept it very simple. We have implemented collaborative as well as content based recommendation systems to recommend songs based upon the users past listening history and from the content of the song.

4.1 Collaborative filtering

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if person *A* has the same opinion as person *B* on an issue, A is more likely to have B's opinion on a different issue than that of a randomly chosen person.

4.1.1 Methodology

This section gives the insights about the preprocessing of data as well as the filtering techniques that are used to recommend songs.

Dataset

For collaborative filtering we have used a dataset provided by the million song dataset.We have not used the core dataset which is released by echo nest which contains almost 48 million triplets of (userId, songId, count), but instead we have used subset of approximately 1GB of data containing 10K triplets,
In the dataset we are not provided with the timestamp when the song was released or where the song was released, having those values will help us build more robust system by recommending songs that are released in given timestamp or there are high chances that Indian will be listening to the indian music rather than the spanish song. To solve one of the problems we merged this echonest dataset with the song metadata which contains the artist name, song id, year in which it is released and title. The sample of data can be seen in figure 4 and figure 5.

```
merge_df.head()
```

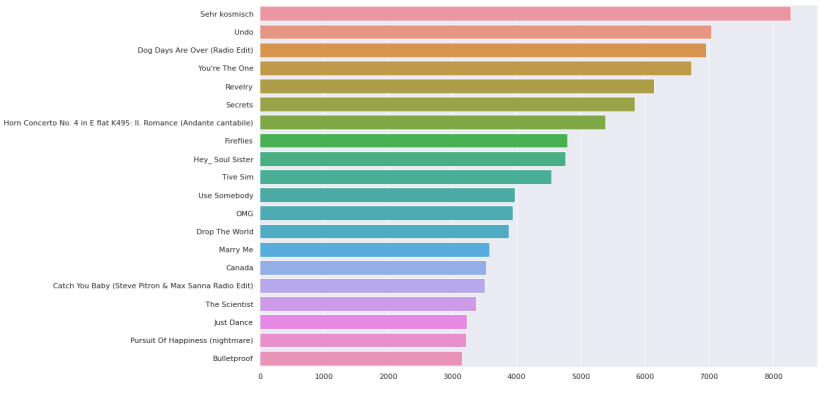
	user_id	song_id	count	title	release	artist_name	year
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	The Cove	Thicker Than Water	Jack Johnson	0
1	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBBMDR12A8C13253B	2	Entre Dos Aguas	Flamenco Para Niños	Paco De Lucia	1976
2	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBXHDL12A81C204C0	1	Stronger	Graduation	Kanye West	2007
3	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOBYHAJ12A6701BF1D	1	Constellations	In Between Dreams	Jack Johnson	2005
4	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SODACBL12A8C13C273	1	Learn To Fly	There Is Nothing Left To Lose	Foo Fighters	1999

Fig 24 : after merging dataset displayed in fig 4 and fig 5

Data preprocessing and analysis

After merging 2 dataset, user taste profile and song metadata we now have around more information about the user listening to given song, but while merging on song id there were problem while merging because there were chances that there may be duplicates record found or what if song id is not found therefore we had made sure that we merge dataset only when song id are present in both the dataset.
After merging the dataset we found that there are 100000 songs present in the data out of which 999056 songs are unique,duplicates have been dropped and after the data cleaning now we have 10000 unique songs,9567 unique song titles, and 76353 unique number of users. We then found the top 20 sings that are listened most number of times as well as top 20 artist whose songs are listened most number of times, this is done because there are high chances that if users are listening to that song it is very popular and other users may like it and same goes with the artist, if artist is popular there are chances that user might have heard of the song sung by the artist and they would like that song.

One thing that we found from the dataset is that the average number of times a song is heard by the user is 1 time, and it is very confusing to tell from that whether the user liked that song or have just tried listening to that song. So to clear this type of confusion we have made user wise segregation, we found that users listen to songs on an average of 3 times and the median number of times that users listen to songs is 16 times. To make sure that user likes the song we are considering that user that has listened to the song more than 16 times. Matrix is being created of shape 10000 x 36561 out of which only 1635302 have ratings stored in it. Sparse matrix is one of the problems in collaborative filtering.



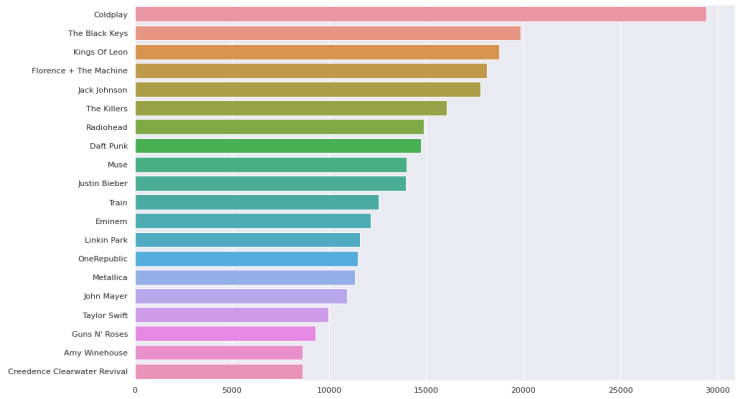


Fig 25: top 20 songs listened by all uses in dataset Fig 26: top 20 artists whose songs are most listened

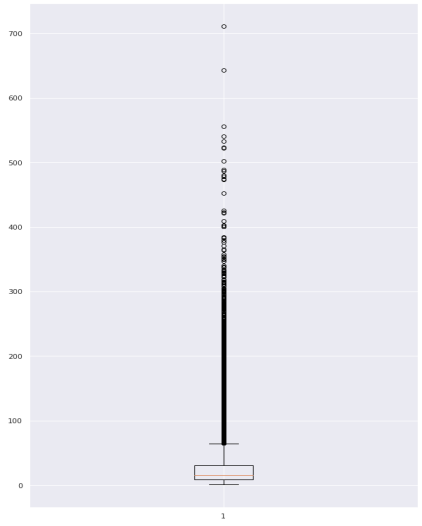


Fig 27: Plot representing the number of times each song is listened to

Recommendation using SVD, BaselineOnly and NMF model

In the user profile taste we are not given the rating and instead we are given counts, which is the explicit form of rating, we can assume that if user listens to the same song multiple times user would have liked that song but again there is an issue over here where it is very difficult to say that when we encounter 0 because it is confusing that whether user did not liked the song or have not came across it yet, we have assumed that 0 means that user did not liked the song.

```
user_song.head()
```

user_id	000e2c2a8c7870ff9121f212b35c8b3a20cc0e67	000ebc858861aca26bac9b49f650ed424cf882fc	000ef25cc955ad5841c915d269432eea41f4a1a5	0012bf7
song_id				
SOAAAGQ12A8C1420C8	0.0	0.0	0.0	
SOAACPJ12A81C21360	0.0	0.0	0.0	
SOAACSG12AB018DC80	0.0	0.0	0.0	
SOAAEJI12AB0188AB5	0.0	0.0	0.0	
SOAAFAC12A67ADF7EB	0.0	0.0	0.0	

5 rows × 36561 columns

Fig 28: matrix of user vs song(it is sparse matrix)

To recommend songs we have used SVD, BaselineOnly and NMF methods of surprise package to recommend songs. We have used RMSE and MAE metrics to compare the results of all the 3 methods.

If we understand the working of methods it is:

SVD: It is matrix factorisation based algorithm, the prediction is performed using the

$$\hat{r}\,ui = \mu + bu + bi + qTipu$$

If the user is unknown than the bias bu and the factors pu are assumed to be 0.

We have tried creating a matrix of user vs song which is a sparse matrix and only few of the data is present.

BaselineOnly: This method tries to minimise the regularised square error which is given by:

$$\sum rui \in Rtrain(rui - (\mu + bu + bi))2 + \lambda(b2u + b2i)$$

BaselineOnly can be calculated using either stochastic gradient or using Alternating least square error.

We have used alternating square error because we wanted to regularise the parameters related to user and items which can be done in alternating square error.

NMF: A collaborative filtering algorithm based on Non-negative Matrix Factorization.It is similar to SVD.

It uses stochastic gradients and the step size is decided in a way that makes sure that there are non-negative factors.

Note: This is just the surface knowledge about the algorithms that we have used, assuming that reader already knows about this algorithm, here only part that is relevant to recommending songs is explained.

From the Results we found that Baseline only after configuring it performed better on our dataset giving us the least RMSE score on the test dataset while we got the least MAE score by using NMF, going forward with RMSE metrics, we have decided to use BaselineOnly method to recommend songs.

```
algo_Results.head()
```

	Algorithm	test_rmse	test_mae	fit_time	test_time
0	BaselineOnly	6.339318	2.274690	18.220139	7.349895
1	NMF	6.431054	2.162662	156.554848	8.114863
2	SVD	6.855565	3.906780	113.476743	8.854725

Fig 29: RMSE and MAE score

Top 5 songs are predicted for the given user by entering the user id, in the background it returns the song id which is matched with the title of the song and title are returned along with the approximate rating that might be given by the user. Basically we can say in a way that it is the confidence of the model of the user liking the given song.

```
Song recommended for given user is/are: with predicted rating
Undo    5
The Want Ad     4.572555799605418
Catch You Baby (Steve Pitron & Max Sanna Radio Edit)    4.498100925361384
The Fear        2.046453909656938
Oh What A Night 1.8725458420842105
Cuando Pase El Temblor  1.5392269821401263
Roll Over Beethoven     1.363804919862884
```

Fig 30: Recommended songs according to the collaborative filtering

Conclusion

The BaselineOnly is used to recommend song with RMSE of 6.34 which is the lowest in all the 3 algorithm but what concerts us is why SVD is not performing better than the BaselineOnly even when it have bias term with it,and after researching more about it we found that it is because algorithm is not tuned properly, even after trying different parameter we were not able to get the hyperparameter with which SVD performs better than BaselineOnly, though it is still something to ponder upon and requires much more effort to find the correct hyperparameters.

4.2 Content based filtering

Content-based filtering methods are based on a description of the item and a profile of the user's preferences.These methods are best suited to situations where there is known data on an item (name, location, description, etc.), but not on the user. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on an item's features.

4.2.1 Methodology

This section gives the insights about the preprocessing of data as well as the filtering techniques that are used to recommend songs.

Dataset

The content based filtering is based on the lyrics and for that we have used the musiXmatch dataset.
This dataset has mxm IDs which are different from the song IDs from our core data set called the MSD IDs. So the first part of preprocessing was to merge both the dataset based on their IDs. This was done by using a supporting dataset in the form of a SQLite DB called the mxm_779k_matches. It matches the Mxm IDs to MSD IDs. Basic ideas about the dataset can be obtained from figure 6 and figure 7.

Data preprocessing and analysis

To recommend the song based on the content, multiple datasets and supporting datasets were used to establish the link between the 2 datasets. It basically works on the similarity between the 2 given item set in our case songs, the features that are given for the song can be used to find the similarity between 2 songs, if seen in larger sense there are high chances that recommended song will come from the cluster that we have formed because they will be having similar features but there are chances that songs not

belonging to that cluster can be recommended because of similar artist, year published or based on lyrics, in our project we have worked upon the lyrics of the dataset and artist to recommend simi;ar song.

Lyrics dataset is present in million song dataset in jumbled form for each song because of the copyright issue, to match the song with the other information we need supporting dataset that is track metadata which is used to establish the link between the song and other information in song metadata dataset.

Lyrics of the song are being cleaned by performing stemming, stopwords removal and tokenization, but before doing this we had created the bar plot of the most common words present in the dataset to get the idea about what lyrics represent.

```
match_merge.head()
```

	user_id	song_id	count	track_id	artist_name	title	mxm_artist	mxm_title	mxm_id
0	b80344d063b5ccb3212f76538f3d9e43d87dca9e	SOAKIMP12A8C130995	1	TRIQAUQ128F42435AD	Jack Johnson	The Cove	Jack Johnson	The Cove	7232629.0
1	7c861769417189 84fed11b7c0674ff04c029b480	SOAKIMP12A8C130995	1	TRIQAUQ128F42435AD	Jack Johnson	The Cove	Jack Johnson	The Cove	7232629.0
2	76235885b32c4e8c82760c340dc54f9b608d7d7e	SOAKIMP12A8C130995	3	TRIQAUQ128F42435AD	Jack Johnson	The Cove	Jack Johnson	The Cove	7232629.0
3	250c0fa2a77bc6695046e7c47882ecd85c42d748	SOAKIMP12A8C130995	1	TRIQAUQ128F42435AD	Jack Johnson	The Cove	Jack Johnson	The Cove	7232629.0
4	3f73f44560e822344b0fb7c6b463869743eb9860	SOAKIMP12A8C130995	6	TRIQAUQ128F42435AD	Jack Johnson	The Cove	Jack Johnson	The Cove	7232629.0

Fig 30: Merging of data

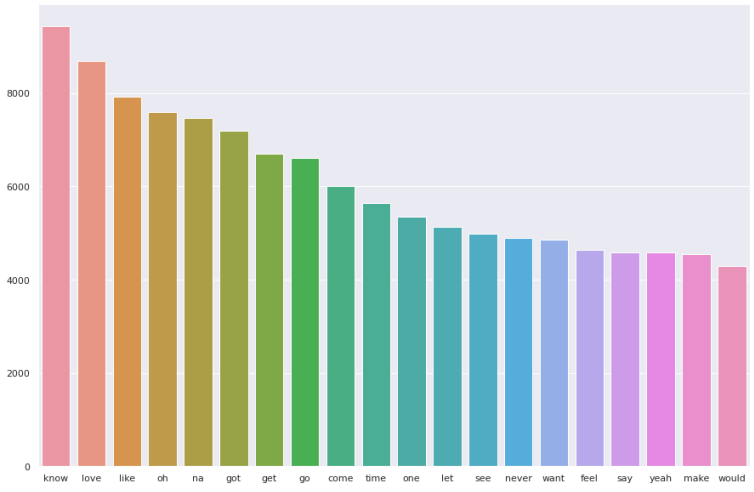


Fig 31: Top 20 words used in lyrics

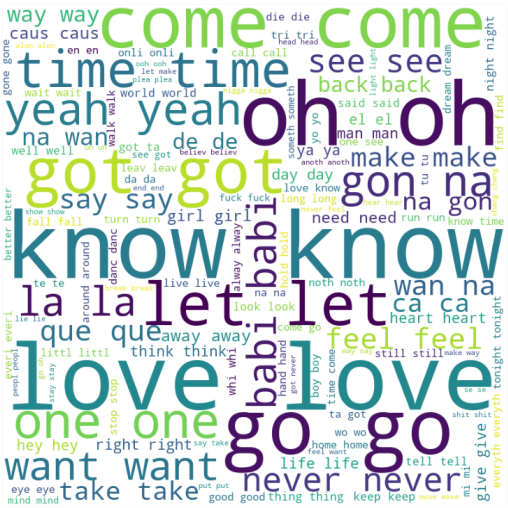


Fig 32: word cloud

```
merge_data.head()
```

	track_id	lyric	lyric_clean
0	TRAABJS128F9325C99	come one one want way say eye tell still still...	[come, one, one, want, way, say, eye, tell, st...
1	TRAACIE128F428495B	love love like like come one never let let say...	[love, love, like, like, come, one, never, let...
2	TRAACPH12903CF5F14	come see eye thing live live everyth word much...	[come, see, eye, thing, live, live, everyth, w...
3	TRAADKW128E079503A	know like like like like go go get get get get...	[know, like, like, like, like, go, go, get, ge...
4	TRAADQL128F427D281	love know like time go get get get get get get...	[love, know, like, time, go, get, get, get, ge...

Fig 33: Lyrics after preprocessing.

Feature Engineering

Content-based filtering generates song playlists based on the cosine similarity calculated for three lyrical features, including (1) TF-IDF, (2) Word2Vec, and (3) Topic Model with Latent Dirichlet Allocation (LDA).

TF-IDF : TF-IDF stands for “Term Frequency – Inverse Document Frequency”. In this case, we vectorized the lyrics and used TF-IDF to quantify the word in song lyrics. We computed a weight to each word which signified the importance of the word in the lyrics. We generated a TF-IDF matrix using tfidf vectorizer from sklearn and it has 200 columns pertaining to 200 features.

Word2Vec : Word2Vec is a two-layer neural net that processes text by “vectorizing” words and turns text into a numerical form. Word2vec groups vectors of similar words in vector space as it detects similarities mathematically. We generated a Word2Vec matrix using gensim and it has 200 columns pertaining to 200 features.

LDA: Topic Modeling is an unsupervised approach used for identifying topics present in a text object and to derive hidden patterns exhibited by a text corpus. Latent Dirichlet Allocation (LDA) is one of the topic models that builds a topic per document model and words per topic model. The big idea behind LDA is that each document can be described by a distribution of topics, and each topic can be described by a distribution of words. We first generated a list containing unique tokens from the lyric_token column of our dataframe. Using that we built a corpus and generated a term document matrix.We took the transpose of that matrix, converted it to a dataframe and used it to fit in a LDA model. The LDA matrix has dimension 5630*100 i.e. 5630 songs and 100 topics.

Note: dataset containing the TFIDF vectors, word2Vec and LDA can be seen in the notebook file because the dataset contains a high number of features and it is not possible to show them over here.

Recommendation using cosine similarity

For recommending songs to the users based on song id and artist name, we first calculated cosine similarity between the feature matrix of TF-IDF, word2Vec and LDA. In this case, the algorithm is designed for new users to search for a specific song. The users enter their favorite song or artist, then the system will match the song’s index with cosine similarity matrix and return 10 songs with similar content. So the final recommendation of the number of songs will be common songs in the top 10 songs of above three methods. For example, 4 similar songs between TFIDF and Word2vec methods; 2 similar songs between TFIDF and LDA methods; 2 similar songs between Word2vec and LDA methods; 1 similar song among three methods. Therefore, this 1 song is the final recommended song.

```
[ ] #Recommendation using tfidf
    recommend_title("Anyone Else But You","Michael Cera & Ellen Page",similarity_tfidf)
```

	artist_name	title
0	Michael Cera & Ellen Page	Anyone Else But You
1	The Moldy Peaches	Anyone Else But You
2	Gene Kelly	Singing in the Rain
3	Markus Krunegård	Genom tunna tyger
4	Alexi Murdoch	Love You More
5	Rammstein	Du Riechst So Gut
6	Suspekt	Kinky Fætter
7	Hot Chip	Look After Me
8	Rammstein	Tier
9	The Killers	Romeo And Juliet
10	Rammstein	Seemann

Fig 34: recommendation using tfidf vecotrs

```
#recommendations using word to Vector
recommend_title("Anyone Else But You","Michael Cera & Ellen Page",similarity_wordToV
```

	artist_name	title
0	Michael Cera & Ellen Page	Anyone Else But You
1	The Moldy Peaches	Anyone Else But You
2	Beyonce	Daddy
3	NEEDTOBREATHE	Shine On (Album Version)
4	Hot Chip	Playboy
5	Jason Mraz	You And I Both (LP Version)
6	Mike Jones	Mr. Jones (Explicit Version)
7	Owl City	Tidal Wave
8	Vampire Weekend	M79 (Album)
9	Bright Eyes	First Day Of My Life (Single Version)
10	Radiohead	How Do You?

Fig 35: recommendation using word2Vec

```
#Recommendation using LDA
recommend_title("Anyone Else But You","Michael Cera & Ellen Page",similarity_lda)
```

	artist_name	title
0	Michael Cera & Ellen Page	Anyone Else But You
1	The Moldy Peaches	Anyone Else But You
2	Suspekt	Kinky Fætter
3	Blur	For Tomorrow
4	Goldfrapp	Ooh La La
5	Mae	Communication
6	A Day To Remember	If It Means A Lot To You
7	Blur	This Is A Low
8	Nirvana	Breed
9	The Grass Roots	Let's Live For Today
10	Lady GaGa	Fashion

Fig 36: recommendation using LDA

Recommendation using linear regression on user data

Linear regression is used to predict the rating of the song that is not being heard by the user and based on the score that we receive, user will be recommended songs, based on matrix factorisation we will be obtaining row vector for each user having number of columns same as the number of unique songs present in the dataset with some value indicating the chances of user liking that song. Top 5 songs are then extracted from those row vectors which will be displayed to the user.

We have tried doing the Linear regression by performing matrix multiplication according to what we are taught in the class but though we faced some issues and are not able to complete it, but if looked upon as the theoretical aspect we have used TF-IDF vectors to use in Linear regression by taking row vectors corresponding to the the user if the user have given rating to that song or not. The extracted matrix is multiplied with the rating that is extracted from the matrix created between user_id v/s song_id, according to the theory they can be multiplied but we are getting different dimensions for multiplication and therefore matrix multiplication is not possible.

Conclusion

We have used cosine similarity on top of 3 algorithms which are tf-idf vectors, word to vector and LDA to predict the top 10 songs based on song id and the artist and we have found that the it works upto some extent by predicting the correct result based on features that given as input, the system can be built in more sophisticated and more accurate way if we have audio of the song as well, if we had the audio for it then we can even extract the audio features like done in the clustering part which also plays an important role in deciding the chances of song being liked by the user or not.

5. Future Work

As per the proposed approach we wish to have the pipeline where dataset should contain the audio as well as the user history, audio can be used to derive the features that can be used for clustering the songs and getting the lyrics from it using the python library which converts audio to text and user history to perform the collaborative filtering, in short having this type of dataset we will be able to create a pipeline as a whole.The other thing that could be worked upon is the model updation at every step, for example if user listens to the song recommended by the model or searches by themself and liked the song then we should add that in real time and predict new songs based on it, if the user is not listening to the songs that are recommended by the system for some period of time then instead of showing recommended songs, random songs should be displayed hoping that user might like one of them.

If we are able to detect the mood of the user based on his facial expression and we are able to capture the location of the user then we can recommend songs according to it as well, for example if user is in sad mood we can recommend song according to it, and if user is happy and is not at usual location and the location that we found is of some party then songs should be recommended based on that, but for everything that is discussed above requires merging of everything that we have done until now and adding few things on top if it, the main concern would be how to weight the recommendation coming from each system because they will be recommending different songs and it is important to decide some weighing factor which can be used to decide the final confidence score. According to us the weighing factor should be dynamic and shall depend upon the system which is another topic to ponder upon and different machine learning models could be designed to predict the weighing score for each system which is personalised for each user.

6. References

1. https://codeburst.io/2-important-statistics-terms-you-need-to-know-in-data-science-skewness-and-kurtosis-388fef94eeaa
2. https://ccrma.stanford.edu/~unjung/AIR/areaExam.pdf
3. https://librosa.org/doc/0.8.0/generated/
4. https://ls11-www.cs.tu-dortmund.de/_media/techreports/tr08-01.pdf
5. https://wiki.aalto.fi/display/ITSP/Zero-crossing+rate
6. https://en.wikipedia.org/wiki/Mel-frequency_cepstrum
7. https://surprise.readthedocs.io/en/stable/index.html
8. https://medium.com/
9. https://towardsdatascience.com/
10. https://www.uvm.edu/~ylin19/files/Music_Recommer_System_Report.pdf
11. http://www.diva-portal.org/smash/get/diva2:1515217/FULLTEXT01.pdf
12. https://portfolios.cs.earlham.edu/wp-content/uploads/2019/08/final_paper_draft_Vuong.pdf
13. https://www.ijert.org/music-recommendation-system-using-content-and-collaborative-filtering-methods