NEXT SOUND

1832048
D. SHREYAS

1832049
S.SIDDHESH

PROJECT DEFINITION AND OBJECTIVE:

One of the great promises of the internet and Web 2.0 is the opportunity to expose people to new types of content. Companies like Amazon and Netflix provide customers with ideas for new items to purchase based on current or previous selections. For instance, someone who rented “Star Wars” at Netflix might be presented with “The Matrix” as another movie to rent. The challenge in this strategy is to make suggestions in a reasonable amount of time that the user will mostly like based on the known list of what the user already enjoys. People will only pay attention to the recommendations of a service a finite number of times before they lose trust. Repeatedly suggest content that the user hates and the user will look for new content elsewhere. Also, a user will only pay attention to a service’s recommendation if they arrive in a reasonable amount of time. Make the user wait longer than they are willing and they will again turn elsewhere for content suggestions. Accuracy and speed are critical to a service’s success.

With the explosion of the network in the past decades, internet has become the major source of retrieving multimedia information such as video, books, and music etc. People has considered that music is an important aspect of their lives and they listen to music, an activity they engaged in frequently. Previous research has also indicated that participants listened to music more often than any of the other activities (i.e. watching television, reading books, and watching movies). Music, as a powerful communication and self-expression approach, therefore, has appealed a wealth of research.

Additionally, music recommender is to help users filter and discover songs according to their tastes. A good music recommender system should be able to automatically detect preferences and generate playlists accordingly. Meanwhile, the development of recommender systems provides a great opportunity for industry to aggregate the users who are interested in music. More importantly, it raises challenges for us to better understand and model users preferences in music.

Why are we doing this?

Music Experts have been trying for a long time to understand sound and recommend the next similar songs for the users based on genre played that's our project Next sound.

A successful music recommender needs to meet users various requirements. However, obtaining user information is expensive in terms of financial costs and human labor . For user-oriented design, lots of efforts on user studies need to be investigated. User modelling, as the one of the key elements, it models the difference in profile. For example, the difference in geographic region or age, their

music preferences might be different. Interestingly, other factors such as gender, life styles, and interests could also determine their choices of music. Recent research has revealed that intelligence, personality and the users preference in music are linked . According to Researchers who had investigated the relationship between music preference and Big-Five Inventory (BFI: openness, conscientiousness, extraversion, agreeableness, and neuroticism), their studies showed a highly extraverted person would tend to choose the music which is energetic, while a greater preference for rhythmic and energetic music was associated with greater extraversion and agreeableness.

DATASET DESCRIPTION:

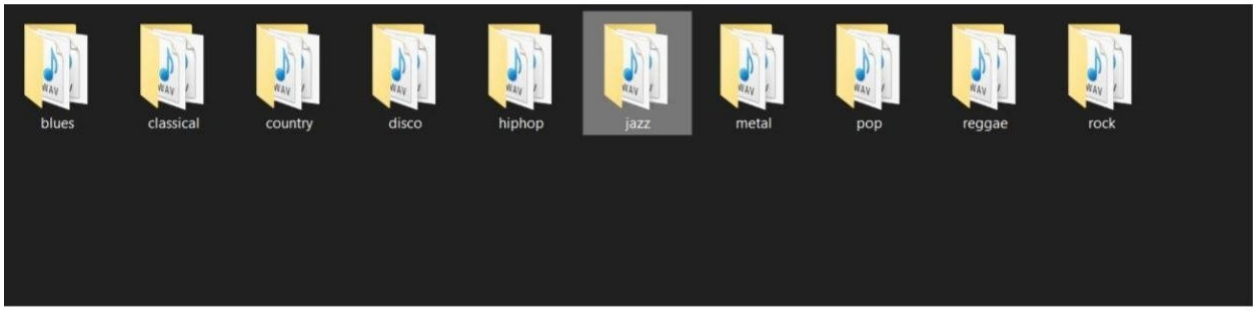
In the world of Music, genres are categorical labels which are used to characterize pieces of music. The genre of the music is characterized by the common characteristics . These are typically related to the instrumentation, rhythmic structure, and harmonic content of the music.

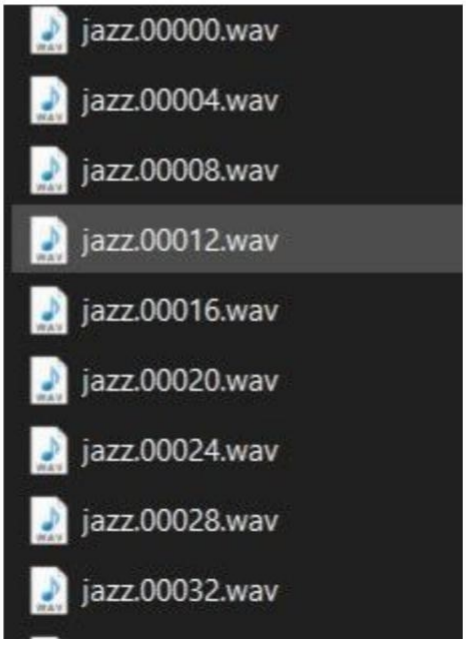
The GTZAN dataset is the most-used public dataset for evaluation in machine listening research for music genre recognition (MGR). It is the real world audio collection of various kinds of music.The files were collected in 2000-2001 from a variety of sources including personal CDs, radio, microphone recordings, in order to represent a variety of recording conditions. The GTZAN dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050Hz Mono 16-bit audio files in .wav format.The genres are

```
['blues', 'classical', 'country', 'disco', 'hiphop', 'jazz', 'metal', 'pop', 'reggae', 'rock']
```

This dataset also contains CSV files consisting of the mean and average value of different extracted features from the audio data.

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521	1773.065032
1	blues.00000.1.wav	66149	0.343065	0.086147	0.112699	0.001450	1816.693777
2	blues.00000.2.wav	66149	0.346815	0.092243	0.132003	0.004620	1788.539719
3	blues.00000.3.wav	66149	0.363639	0.086856	0.132565	0.002448	1655.289045
4	blues.00000.4.wav	66149	0.335579	0.088129	0.143289	0.001701	1630.656199





CONCEPTS USED:

- LIBROSA - Extraction of features is a very important part in analyzing and finding relations between different things. The data provided of audio cannot be understood by the models directly to convert them into an understandable format feature extraction is used. It is a process that explains most of the data but in an understandable way. Feature extraction is required for classification, prediction and recommendation algorithms
- Librosa.display is used to display the audio files in different formats such as wave plot, spectrogram, or colormap. Wave Plots let us know the loudness of the audio at a given time. Spectrogram shows different frequencies playing at a particular time along with it's amplitude. Amplitude and frequency are important parameters of the sound and are unique for each audio. librosa.display.waveplot is used to plot waveform of amplitude vs time where the first axis is an amplitude and second axis is time.
- A spectrogram is a visual representation of the Spectrum of frequencies of sound or other signals as they vary with time. It's a representation of frequencies changing with respect to time for given music signals.
- IPython.display allow us to play audio on jupyter notebook directly. It has a very simple interface with some basic buttons.
  Import IPython.display as
  ipd ipd.Audio(audio_path)

Machine Learning models used in this project are followed as

- Naive bayes
- K-Nearest-Neighbours
- Random Forest

- Decision Tree
- Support Vector Machine
- Logistic Regression
- Neural Network

After the model fitting the best performing models were taken to hyper parameter tuning parts to show more accuracy and improve the model performance .

- ELI 5 is a Python package which helps to debug machine learning classifiers and explain their predictions. Currently ELI5 allows to explain weights and predictions of scikit-learn linear classifiers and regressors, print decision trees as text or as SVG, show feature importances and explain predictions of decision trees and tree-based ensembles.

- Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space

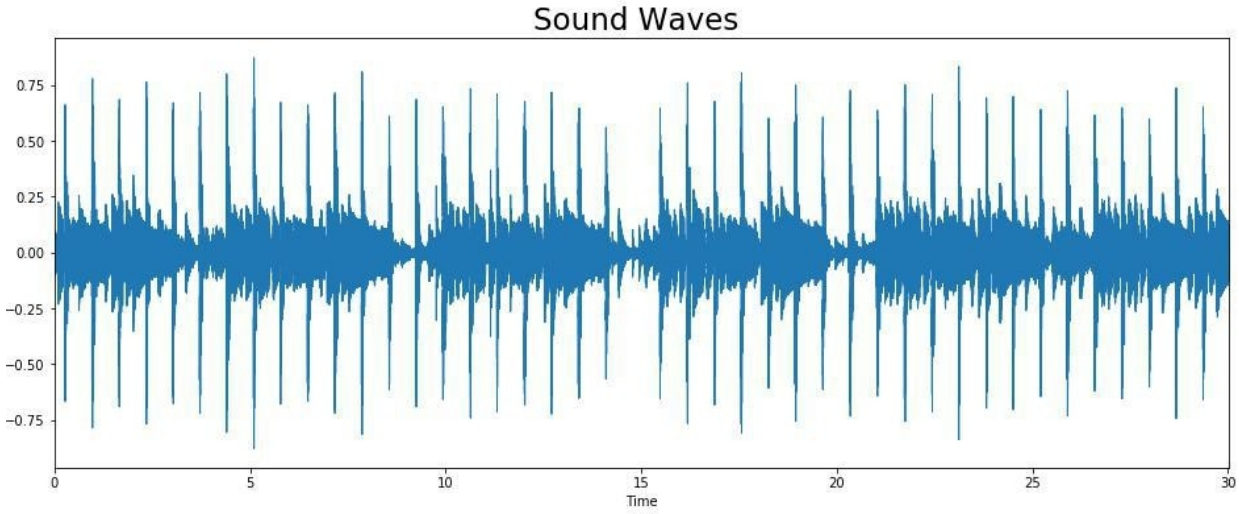
EXPLORE AUDIO DATA:

The audio data is explored by using the Librosa module, with the help of this module initially the sequence of vibration and number of samples of audio carried per second(Sample Rate) are determined.

```
y: [0.02072144 0.04492188 0.05422974 ... 0.06912231 0.08303833 0.08572388]
y shape: (661794,)
Sample Rate (KHz): 22050
```

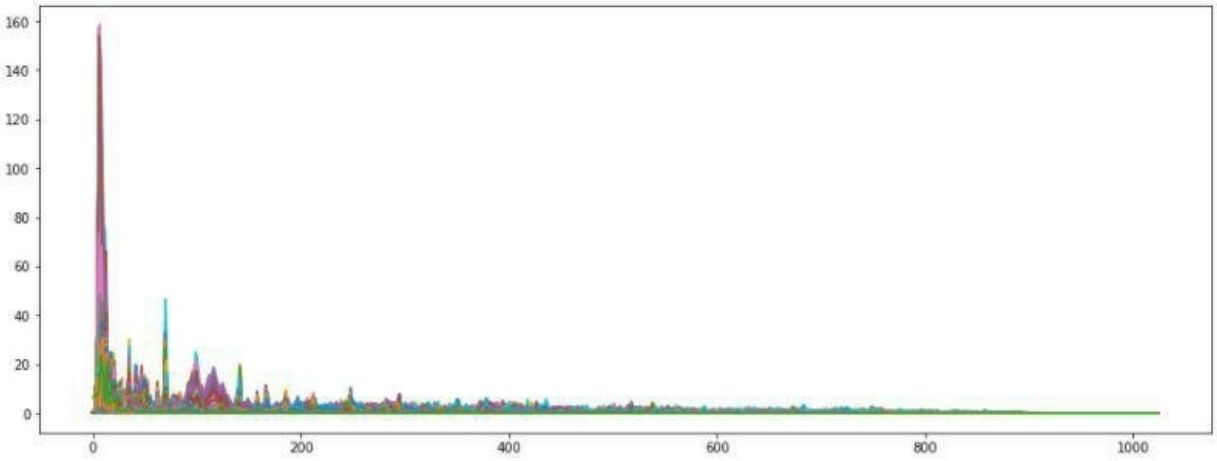
WAVE REPRESENTATION OF AUDIO FILE:

The wave representation of an audio file used to infer information about the frequency of the audio data.



FOURIER TRANSFORMATION:

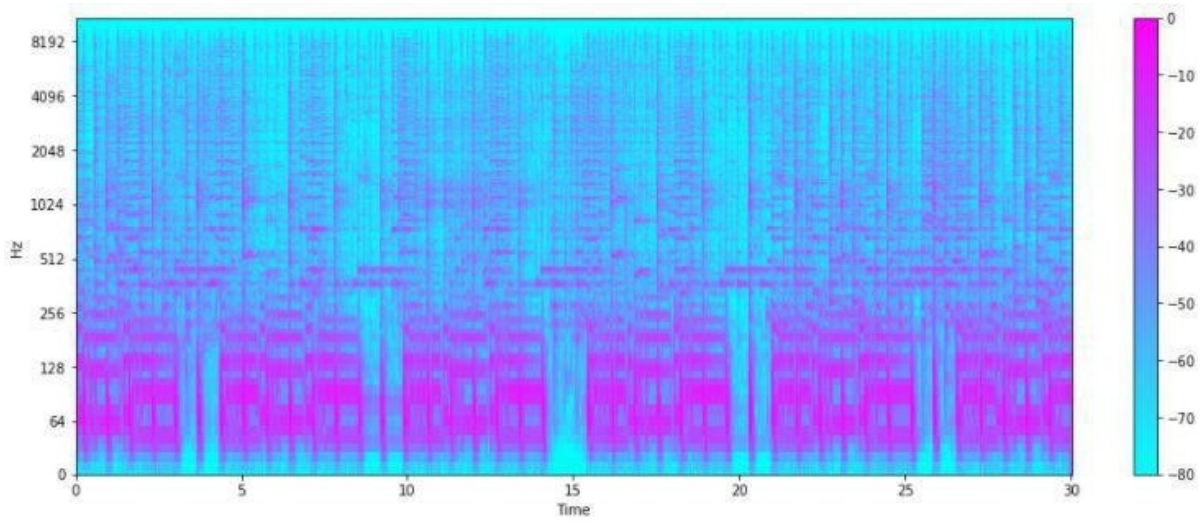
Fourier transform will get a complex periodic sound signal as input and decompose it into a sum of sine waves oscillating at different frequencies.



From the above graph it is inferred that time domain graph is converted into frequency domain graph using short time fourier transform. This computes several FFT at different intervals and gives spectrograms.

SPECTROGRAM:

A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. Here we are converting an amplitude spectrogram into a decimal scaled spectrogram.



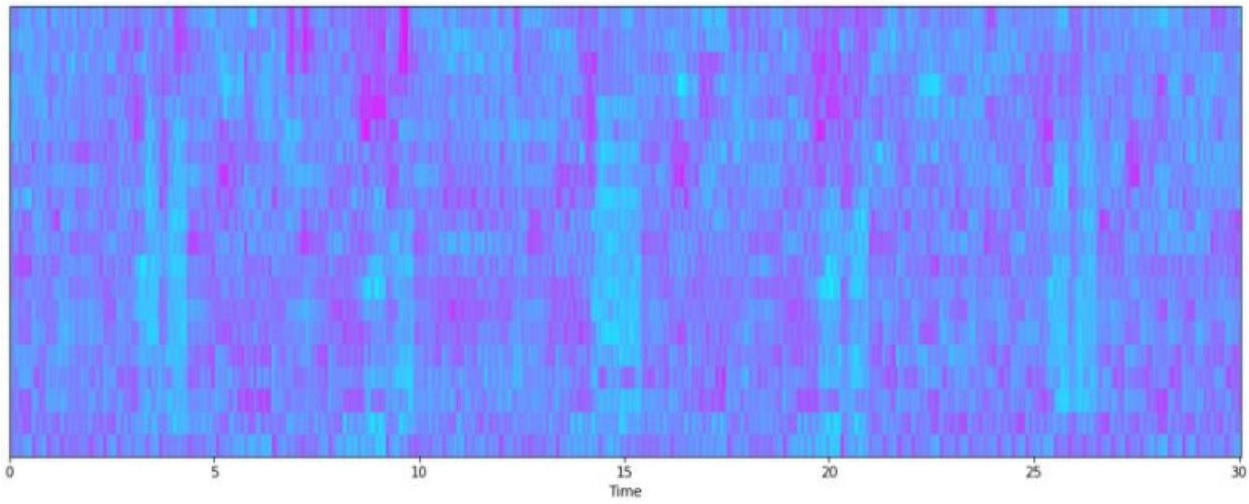
MEL - FREQUENCY CEPSTRAL COEFFICIENTS:

The Mel frequency cepstral coefficient tells the overall shape of a spectrum, which i s derived from the Mel spectrogram. This spectrogram is a non-linear transformation of the frequency.

```
shape: (20, 1293)
```

Using scaling techniques the mean and variance of the data is derived.

```
Mean: 1.032594e-09

Var: 1.0
```



The above figure represents the scaled decimal spectrogram. Hence we have processed the single audio file from which we have derived mean and variance. Further we are going to use machine learning techniques with CSV files consisting of the mean and average value of different extracted features for all audio data.

EXPLORATORY DATA ANALYSIS:

Exploratory Data Analysis is a process of investigating the data to find patterns and anomalies with the help of summary statistics and graphical representations.

Shape:

```
(9990, 60)
```

The dataset has 9990 instances and 60 attributes.

Attributes:

```
Index(['filename', 'length', 'chroma_stft_mean', 'chroma_stft_var', 'rms_mean',
       'rms_var', 'spectral_centroid_mean', 'spectral_centroid_var',
       'spectral_bandwidth_mean', 'spectral_bandwidth_var', 'rolloff_mean',
       'rolloff_var', 'zero_crossing_rate_mean', 'zero_crossing_rate_var',
       'harmony_mean', 'harmony_var', 'perceptr_mean', 'perceptr_var', 'tempo',
       'mfcc1_mean', 'mfcc1_var', 'mfcc2_mean', 'mfcc2_var', 'mfcc3_mean',
       'mfcc3_var', 'mfcc4_mean', 'mfcc4_var', 'mfcc5_mean', 'mfcc5_var',
       'mfcc6_mean', 'mfcc6_var', 'mfcc7_mean', 'mfcc7_var', 'mfcc8_mean',
       'mfcc8_var', 'mfcc9_mean', 'mfcc9_var', 'mfcc10_mean', 'mfcc10_var',
       'mfcc11_mean', 'mfcc11_var', 'mfcc12_mean', 'mfcc12_var', 'mfcc13_mean',
       'mfcc13_var', 'mfcc14_mean', 'mfcc14_var', 'mfcc15_mean', 'mfcc15_var',
       'mfcc16_mean', 'mfcc16_var', 'mfcc17_mean', 'mfcc17_var', 'mfcc18_mean',
       'mfcc18_var', 'mfcc19_mean', 'mfcc19_var', 'mfcc20_mean', 'mfcc20_var',
       'label'],
      dtype='object')
```

Summary Statistics:

The summary statistics give mean, media, mode, etc.

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean
count	9990.0	9990.000000	9990.000000	9990.000000	9.990000e+03	9990.000000
mean	66149.0	0.379534	0.084876	0.130859	2.676388e-03	2199.219431
std	0.0	0.090466	0.009637	0.068545	3.585628e-03	751.860611
min	66149.0	0.107108	0.015345	0.000953	4.379535e-08	472.741636
25%	66149.0	0.315698	0.079833	0.083782	6.145900e-04	1630.680158
50%	66149.0	0.384741	0.085108	0.121253	1.491318e-03	2208.628236
75%	66149.0	0.442443	0.091092	0.176328	3.130862e-03	2712.581884
max	66149.0	0.749481	0.120964	0.442567	3.261522e-02	5432.534406

Info:

Info tells the data type of each variable in the data set.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9990 entries, 0 to 9989
Data columns (total 60 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   filename                 9990 non-null   object
 1   length                   9990 non-null   int64
 2   chroma_stft_mean         9990 non-null   float64
 3   chroma_stft_var          9990 non-null   float64
 4   rms_mean                 9990 non-null   float64
 5   rms_var                  9990 non-null   float64
```

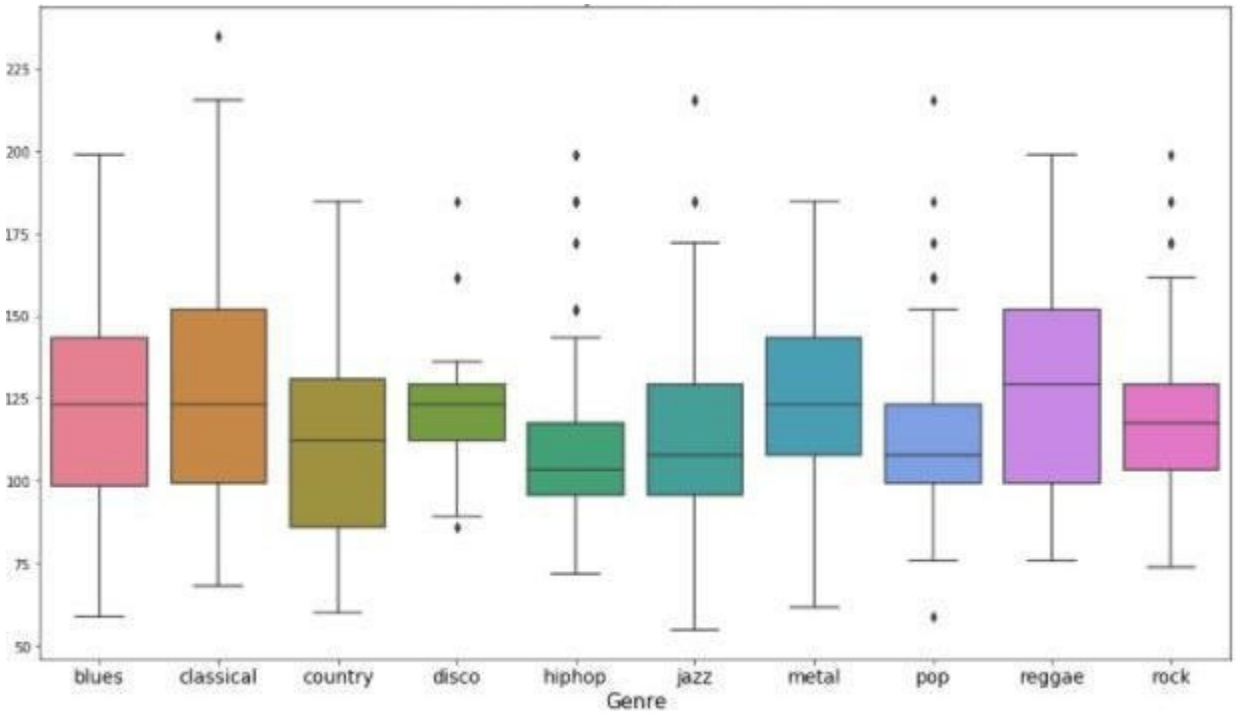
Missing Values:

In this dataset there are no missing values and duplicate values.

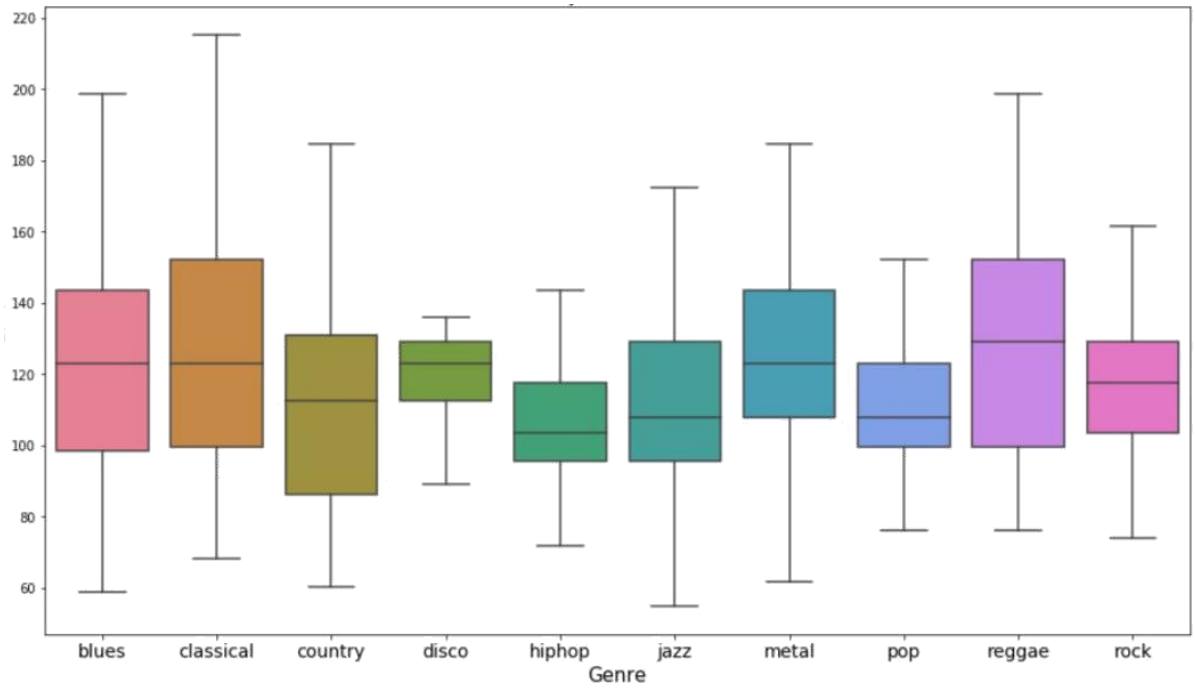
```
filename                 0
length                   0
chroma_stft_mean         0
chroma_stft_var          0
rms_mean                 0
rms_var                  0
```

Outliers:

An outlier is an observation that lies an abnormal distance from other values which affect the statistics like mean, variance, etc.



Here we computed outliers of each genre. This shows the presence of outlier in some genres like hip hop, pop, jazz etc.



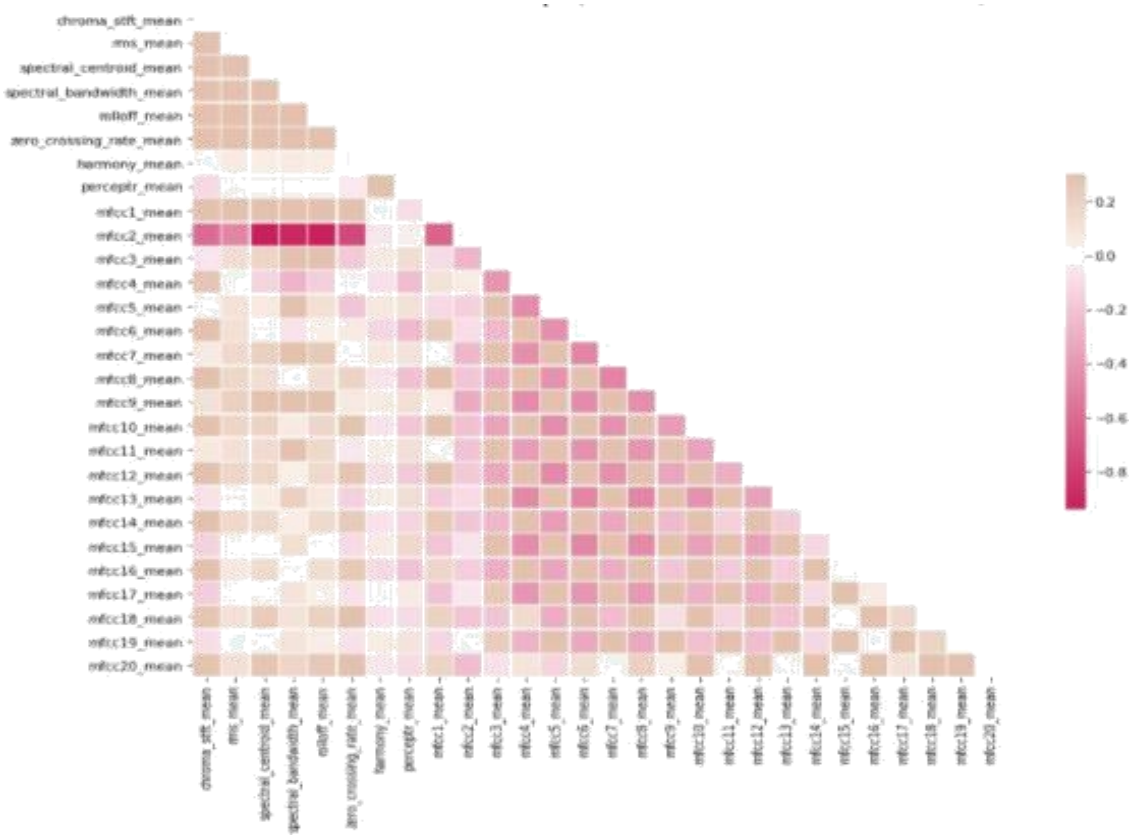
Treating outliers is good for further analysis. Outliers are removed using IQR.The interquartile range (IQR) is a measure of variability.

Principal Component Analysis:

Principal Component Analysis is an unsupervised learning algorithm that is used for the dimensionality reduction .

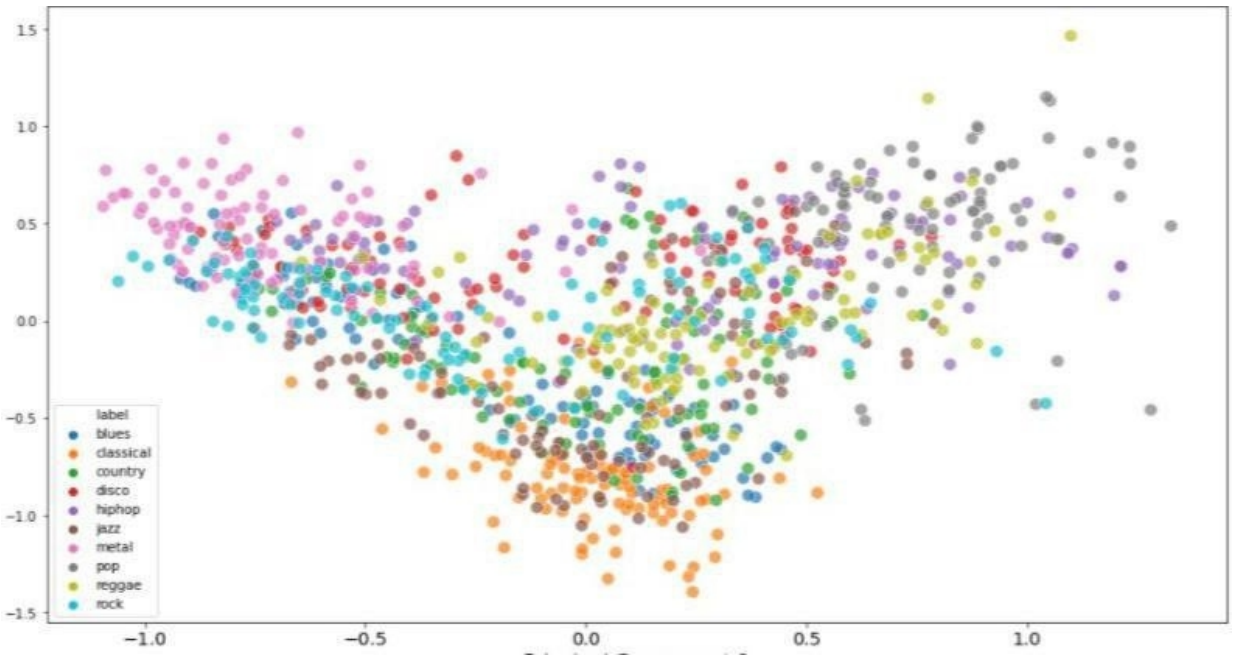
Correlation:

Correlation is a statistical technique that can show whether and how strongly pairs of variables are related. Here we computed the lower triangular matrix for faster computation.



Correlation plays a major role in the PCA analysis as they help to information similarity among the features. The dependency among the features can be inferred from the correlation matrix and the covariance explains how a variable varies with respect to other variables.

PCA Scatter Plot:



The above plot is a visualization graph for possible genre groups. In this graph the labels of each genre are plotted against feature extracted variables.

DATA PREPROCESSING:

We have processed the single audio file and extracted numeric features from the audio data. For further machine learning techniques we need to preprocess the data. Hence the data is numeric it is not necessary to encode the data.

Feature Scaling:

The data is scaled using MinMaxScaler to avoid features with widely different range.

	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean
0	0.0	0.355399	0.716757	0.293133	0.107955	0.262173
1	0.0	0.367322	0.670347	0.253040	0.044447	0.270969
2	0.0	0.373159	0.728067	0.296753	0.141663	0.265293
3	0.0	0.399349	0.677066	0.298024	0.075042	0.238427
4	0.0	0.355668	0.689113	0.322308	0.052149	0.233460

Data Splitting:

The data is splitted into training data and test data for model fitting. The data is splitted in the ratio 70:30.

MODEL FITTING:

After preprocessing, the data is classified using classification algorithms. This classification algorithm tries to draw some conclusions from the input values given for training. It will predict the class labels/categories for the new data. The algorithm fitted are

- Naive Bayes

```
Out[106]: GaussianNB(priors=None, var_smoothing=1e-09)
```

- KNN

```
Out[108]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                               metric_params=None, n_jobs=None, n_neighbors=19, p=2,
                               weights='uniform')
```

- Decision Tree Classifier

```
Out[109]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                 max_depth=None, max_features=None, max_leaf_nodes=None,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=1, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, presort='deprecated',
                                 random_state=None, splitter='best')
```

- Random Forest Classifier

```
Out[110]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                 criterion='gini', max_depth=10, max_features='auto',
                                 max_leaf_nodes=None, max_samples=None,
                                 min_impurity_decrease=0.0, min_impurity_split=None,
                                 min_samples_leaf=1, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, n_estimators=1000,
                                 n_jobs=None, oob_score=False, random_state=0, verbose=0,
                                 warm_start=False)
```

- SVM

```
Out[112]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
              decision_function_shape='ovo', degree=3, gamma='scale', kernel='rbf',
              max_iter=-1, probability=False, random_state=None, shrinking=True,
              tol=0.001, verbose=False)
```

- Logistic Regression

```
Out[113]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                             intercept_scaling=1, l1_ratio=None, max_iter=100,
                             multi_class='multinomial', n_jobs=None, penalty='l2',
                             random_state=0, solver='lbfgs', tol=0.0001, verbose=0,
                             warm_start=False)
```

- Neural Network

```
Out[114]: MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto', beta_1=0.9,
                        beta_2=0.999, early_stopping=False, epsilon=1e-08,
                        hidden_layer_sizes=(5000, 10), learning_rate='constant',
                        learning_rate_init=0.001, max_fun=15000, max_iter=200,
                        momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
                        power_t=0.5, random_state=1, shuffle=True, solver='lbfgs',
                        tol=0.0001, validation_fraction=0.1, verbose=False,
                        warm_start=False)
```

After trying 7 different models we can see the performance of each model the Random Forest is performing pretty much great.

```
Accuracy Naive Bayes : 0.51952

Accuracy KNN : 0.80581

Accuracy Decission trees : 0.6353

Accuracy Random Forest : 0.81415

Accuracy Support Vector Machine : 0.75409

Accuracy Logistic Regression : 0.6977

Accuracy Neural Nets : 0.68068
```

So we are gonna select the best performing three models and apply the hyperparameter tuning to the respective models using a random search cv grid. Svm shows 75% Accuracy so we can take this to consideration for further tuning process similarly Knn and Random forest shows good accuracy(Above 80%) which can be taken into account for the upcoming processes.

MODEL OPTIMIZATION:

1) Tuning Svm model

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:  1.9min finished

Accuracy Support Vector Machine : 0.91225

Best model parameters {'gamma': 1, 'C': 100}

Confusion Matrix:
 [[304   1   4   2   2   2   0   0   2   2]
 [  2 298   0   0   0   7   0   1   0   0]
 [ 11   0 247   0   1   5   0   3   5  14]
 [  5   4   4 268   1   3   1   5   2   8]
 [  1   1   7   6 285   0   1   3   2   5]
 [  3  10   5   1   0 264   0   0   2   1]
 [  4   0   1   1   1   1 290   0   1   4]
 [  0   0   6   3   5   2   0 247   1   3]
 [  0   2   7   4   4   0   0   2 296   1]
 [  6   2  12  19   4   4   9   2   7 235]]
```

The model shows improvement after the tuning 75% to 91% which is really great is't not .

2) Tuning Random Forest Classifier

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed: 19.8min finished

Accuracy Random Forest : 0.89423

Best model parameters {'n_estimators': 1400, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'sqrt', 'max_depth': 80, 'bootstrap': False}

Confusion Matrix:
 [[274   1  19   8   2   4   6   0   5   0]
 [  0 301   0   0   0   5   0   0   0   2]
 [ 12   1 238   7   0  16   2   3   4   3]
 [  1   4   3 264   8   0   1   7   3  10]
 [  2   1   3   3 283   2   4   8   1   4]
 [  3  17   3   1   0 262   0   0   0   0]
 [  0   0   1   0   3   0 288   0   5   6]
 [  0   0   7   2   1   0   0 250   4   3]
 [  1   2   7   4   3   2   0   9 287   1]
 [  2   2  19  20   0   6  11   2   5 233]]
```

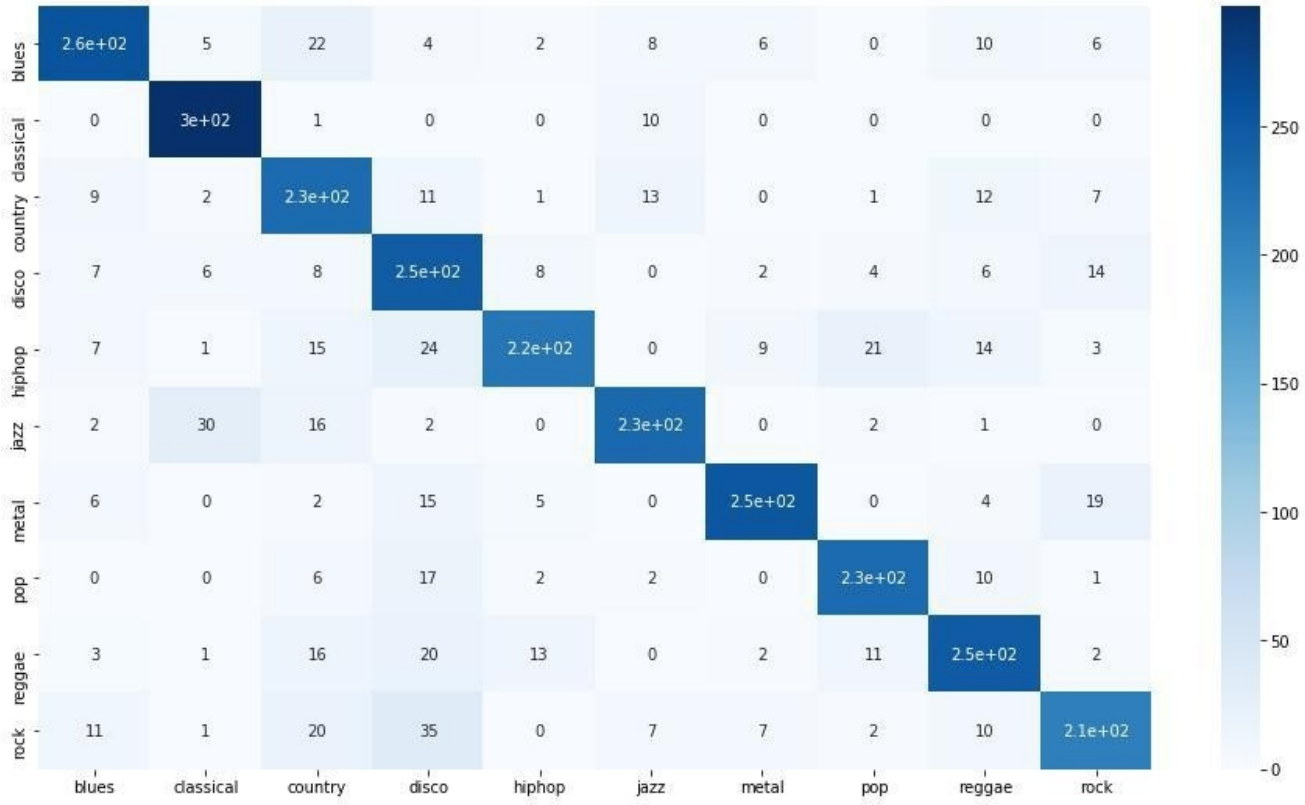
The model has shown increase in the accuracy after the tuning, from 81% to 89%.

3) Tuning KNN MODEL

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done  30 out of  30 | elapsed:  1.9min finished

Accuracy K-Nearest Neighbour : 0.91024

Best model parameters {'weights': 'uniform', 'p': 2, 'n_neighbors': 3, 'metric': 'euclidean', 'leaf_size': 47}

Confusion Matrix:
 [[295   0  12   3   1   1   0   0   4   3]
 [  0 299   1   0   0   8   0   0   0   0]
 [  9   1 253   6   0   5   0   1   6   5]
 [  3   6   1 276   1   2   1   1   2   8]
 [  2   1   9   6 278   1   1   8   3   2]
 [  7  20   6   0   0 253   0   0   0   0]
 [  2   0   1   3   2   0 287   0   3   5]
 [  0   0   4   6   5   2   0 246   2   2]
 [  1   2   7   6   8   1   0   2 289   0]
 [  7   2   8  17   3   3   3   1   4 252]]
```

The model is applied to the tuning after that it shows a good amount of increase in accuracy 80% to 91%.

Confusion Matrix



The Above confusion matrix is obtained using KNN model because it is the best performing model while comparing to the rest of the models ,
For more intuitive about the confusion matrix we are going to explain it
If you see the 2.6e+02 (i.e) approx 260 data that have been correctly predicted that it belongs to the genre Blues and the adjacent value 5 indicates that it actually belongs to the category blue but was falsely predicted as Classical .

Let me come once again to make it clear if you see the category Jazz it has the value around 2.3e+02 (i.e) 230 data that have been correctly predicted that it belongs to the genre jazz if you notice the adjacent value which is zero that implies that the model has not wrongly predicted that it belongs to metal.

FEATURE SELECTION:

Feature selection is the process where you automatically or manually select those features which contribute most to the prediction variable. This can be achieved by using permutation feature importance. Permutation feature importance is a model inspection technique that can be used for any fitted estimator when the data is tabular. This is especially useful for non-linear or opaque estimators .The permutation importance function calculates the feature importance of estimators for a given dataset.

Weight	Feature
0.0903 ± 0.0099	perceptr_var
0.0495 ± 0.0073	perceptr_mean
0.0486 ± 0.0059	harmony_mean
0.0389 ± 0.0047	mfcc4_mean
0.0358 ± 0.0049	chroma_stft_mean
0.0290 ± 0.0045	harmony_var
0.0290 ± 0.0058	rms_var
0.0242 ± 0.0073	mfcc9_mean
0.0206 ± 0.0042	mfcc11_mean
0.0197 ± 0.0028	tempo
0.0173 ± 0.0040	mfcc3_mean
0.0161 ± 0.0030	mfcc6_mean
0.0148 ± 0.0049	mfcc3_var
0.0145 ± 0.0035	spectral_bandwidth_mean
0.0141 ± 0.0029	chroma_stft_var
0.0129 ± 0.0026	mfcc1_var
0.0120 ± 0.0033	mfcc7_mean
0.0103 ± 0.0051	mfcc13_mean
0.0095 ± 0.0072	mfcc1_mean
0.0089 ± 0.0022	mfcc5_var

The above table shows approximate weightage for each feature. This will help the model to find the most contributing feature to the target variables.

NEXT SOUND RECOMMENDATION:

Till now we have classified all the audio files by considering the genres. Now we need to recommend similar songs for next in the music queue. To achieve this similarity of the songs should be derived.

Cosine Similarity:

Cosine similarity finds pairwise similarity for each combination of songs in the data. With the help of cosine similarity, Next Sound Recommendation Systems enable us for any given vector to find the best similarity, ranked in descending order, from the best match to the least best match.

Similarity shape: (1000, 1000)

filename	blues.00000.wav	blues.00001.wav	blues.00002.wav	blues.00003.wav	blues.00004.wav
filename					
blues.00000.wav	1.000000	0.049231	0.589618	0.284862	0.025561
blues.00001.wav	0.049231	1.000000	-0.096834	0.520903	0.080749
blues.00002.wav	0.589618	-0.096834	1.000000	0.210411	0.400266
blues.00003.wav	0.284862	0.520903	0.210411	1.000000	0.126437
blues.00004.wav	0.025561	0.080749	0.400266	0.126437	1.000000

This cause 1000 x 1000 matrix with the help of this our model will recommend the most similar song to the user. Here we are defining a function called find_similarity that takes the name of the song and returns the top 5 best matches for that song.

```
Similar songs to  pop.00019.wav
filename
pop.00023.wav    0.862836
pop.00034.wav    0.860499
pop.00078.wav    0.829135
pop.00088.wav    0.824456
pop.00091.wav    0.802269
Name: pop.00019.wav, dtype: float64
```

These are the similar songs for the uploaded audio file. These songs will add up to the music queue of Next Sound.

INFERENCE:

The data obtained from the audio file have been scaled using a minmax scaler to avoid features with widely different range. Sincs the data is numeric it is not necessary to encode the data. The data is splitted in the ratio 70:30.

Further this data is being fitted with seven classification models and their accuracies are Naive Bayes - 51%, K-Nearest Neighbors - 80%, Decision tree - 63%, Random Forest - 81%, Support vector Machine - 75%, Logistic Regression - 69%, Neural nets - 68%.

By comparing all of these models Support Vector Machine, K-Nearest Neighbors, Random Forest show the best accuracy. For optimization we are hypertunning the 3 best models. After hypertunning we got accuracy higher than before tuning the model. This shows our model has been improved. Along with the accuracy we got some best parameters for each of 3 models.

Support Vector machine:

```
Accuracy Support Vector Machine : 0.91258
Best model parameters {'gamma': 1, 'C': 1000}
```

Random Forest:

```
Accuracy Random Forest : 0.89022
Best model parameters {'n_estimators': 1200, 'min_samples_split': 5,
'min_samples_leaf': 1, 'max_features': 'auto', 'max_depth': 50,
'bootstrap': False}
```

K-Nearest Neighbors:

```
Accuracy K-Nearest Neighbour : 0.92359

Best model parameters {'weights': 'distance', 'p': 1, 'n_neighbors': 5,
'metric': 'manhattan', 'leaf_size': 10}
```

Finally our model can classify the song and recommend some similar song by using cosine similarity. With the help of cosine similarity, Next Sound

Recommendation Systems enable us for any given vector to find the best similarity.

CONCLUSION:

Next Sound is a recommendation model which will fetch you the similar songs that you were listening to. In the initial part, the data were in audio files then we converted the audios into frequency using a fantastic library librosa to deliver the numeric data for our predictions .

The data is preprocessed and exploratory data analysis was carried out for more insights about data. we used machine learning to identify which model suits our dataset precisely so we took bunch of classification and regression models and fitted the data to the model.

Further we used hyper parametric tuning techniques to improve the model further extend and it worked very well and in that KNN model is more powerful/accurate in predicting the correct Genre of the music it belongs to ,the models accuracy was around 92% Respectively.

The recommendation part was quite hectic. We used cosine similarity for finding the similar songs which the listener's may like And it worked pretty well . Next Sound will help the people to listen the similar music,In this project we came across many new packages and functions Like librosa IPython Feature importance technique using ELI5 etc .