International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 12 | December - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

# MUSIC GENRE CLASSIFICATION USING MACHINE LEARNING TECHNIQUS

Sai Ravi Teja Gangavarapu, Aditya Kanbargi, Keerthi Aluvala, Arvapally Sai Sandesh, Abhishek Patil,
Meka Sri Sai Venkat, Allu Chaitanya Swaroop Kumar , Sai Teja N

## ABSTRACT

The objective of this project is to do a comparative study to detect and classify music files automatically based on its genre by using various classification algorithms. Music genre classification is a popular problem in the domain of Music Information Retrieval (MIR) used in many music streaming platforms such as Pandora which is an automated music recommendation service based on the Music Genome Project, that suggests songs to users based on similarity of songs that the user is interested in. In this project we have done a comparative study using various machine learning classification algorithms to classify music files based on its genre.

## 1. INTRODUCTION

Music is described as an art form and a cultural activity whose medium is sound. It is not only for entertainment or pleasure but is being used for a wide variety of purposes due its social and physiological effects. Today there are more than 35 million songs available online and thus it becomes very important to categorise them. Several music industries are making efforts to provide filtered and categorized content to their customers and the public, be it spotify, gaana, amazon music etc. Music genres are created in order to describe and categorise music but, there are no strict boundaries available. Thus the classification of music has become a wide area of research to understand patterns in each of these genres and classify them. This area also plays an important role in music retrieval systems and song suggestion systems. Efficient and accurate systems which can intelligently classify the vast amount of audio data available has become extremely important and there is much research being done. This paper aims to understand patterns in the music in order to classify them accurately. Content and audio signal will be used as major parameters in order to perform classification

### 1.1 System Overview

#### A. Dataset

The GTZAN dataset is the most-used public dataset for evaluation in machine listening research in Music Genre Recognition (MGR). The files were collected in 2000-2001 from various sources that included personal CDs, radio, microphone recordings, in order to represent various recording conditions. It is a collection of 1, 000 music with 30-second, 22050 Hz sampling frequency and 16 bits. Following are the specifications of the dataset :

- Genres Original - Genres in the GTZAN include blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae and rock and all of these genres have 100 music files each.

- Images original - A visual representation for each audio file which helps as an input for the various neural network models.
- CSV files - The dataset contains 2 CSV files that contain features of the audio files.

## B. Audio Features

The following is a detailed description about the features that were used from the dataset. Time and frequency domain digital signal processing was done. Also mean, median, standard deviation were used in order to find out useful features.

- Zero Crossing Rate
- Harmonics and Perceptual
- Tempo BPM (beats per minute)
- Chroma Frequencies
- EDA
- Spectral Centroid
- Spectral Rolloff: It could be defined as the frequency value that corresponds to a certain ratio of the distribution in the spectrum. This rate is usually considered to be 85%.
- Mel Frequency Coefficient of Cepstrum-MFCC

## C. Music Genre Classification using various Algorithms

**1. SVM:** That is used for various Classification as well as Regression problems. The goal of the algorithm is to create the best line or a decision boundary that segregates an n-dimensional space into classes such that a new data point can be easily put into the correct category in future.

**2.K-Nearest Neighbors:** Its performance depends on three factors: the distance metrics, the distance rule, and the value of K. The distance metrics give the measure to locate the nearest neighbors of any incoming data point. The distance rule helps us classify the new data point into a class by comparing its features with that of data points in its neighborhood

**3. Convolutional Neural Networks:** It is a deep learning algorithm that takes an image input and assigns importance (weights/ biases) to various objects in the input image such that it can differentiate one from another. The role of CNN is to reduce the RGB images of an object into a form that is easier to process without losing any feature that is critical to obtain a good prediction.

**4. Stochastic Gradient Descent:** Slope of a function is known as Gradient and it is used to measure the change of one variable with respect to the changes in another variable. Gradient Descent is a convex function for which the output is taken as the partial derivative for the set of parameters of its inputs.

**5. Logistic Regression:** Logistic regression is a statistical model that makes use of a function that models a binary dependent variable, even though many more complex extensions exist.

**6. Recurrent Neural Networks:** RNN is a class of artificial neural networks where the connection between multiple nodes helps in forming a directed graph along a given temporal sequence. Thus, it exhibits a temporal and dynamic behavior.

**7. Random Forest:** The Random Forest is a popular algorithm that belongs to supervised learning techniques. Random Forest is a great model which can be used for dual purposes. Classification and Regression Problems in ML. A random forest algorithm creates decision trees on data samples and then gets the prediction from each one of them

**8. Decision Trees:** The Decision Tree Algorithm is a data mining induction technique that recursively shares a set of records. This algorithm is used to solve regression and classification problems using tree representation

**9. K- means clustering:** K-means clustering is an unsupervised learning algorithm where the model divides or classifies the data into various clusters or classes based on certain parameters. This is an iterative algorithm and the data will be partitioned into K predefined non overlapping clusters.

**10. Cross Gradient Booster:** Cross Gradient Boost is an ensemble of decision trees and uses gradient boosting as a framework. Cross Gradient boost is a variation to the basic gradient boost model and has better computational efficiency and often the model performs better.

**11. Cross Gradient Booster (Random Forest):** Cross Gradient Boost as discussed is used for training Gradient Boosted Decision Trees and other Gradient Boosted Models. Random Forest is a different training algorithm but it also uses the same inference and model representation as decision trees.

**12. Naive Bytes:** Naive Bayes classifier works on the Bayes Theorem. This is a great classification algorithm which is very useful in the case of millions of records and even real time data. This model specifically gives great results in terms of textual data and thus can be a good model in case of content-based music genre classification. It works on conditional probability.

## 1.2 Objective

Music experts have been trying for a long time to understand sound and what differentiates one song from another, how to visualize sound and what makes a tone different from another. In this project, we will go through an in-depth analysis of sound and how we can visualize, classify and ultimately understand it. Through this project we would like to

1. Understand what an Audio file is, what features we can visualize on this kind of data.
2. Perform EDA (exploratory data analysis)
3. Genres Classification on the 3 seconds CSV file (trying multiple models and identify which has the best accuracy)
4. Create a recommender system: given a song, give me top X songs most similar

## 1.3 Applications

This project can be used in music streaming services like Gaana, Saavn, Spotify that are using data collected from the user's music listening history to provide recommendations to their listeners. These music recommendation systems are part of a broader class of recommender systems, which filter information to predict a user's preferences when it comes to a certain item

1) Spotify uses an AI system called Bandits for Recommendations as Treatments or simply known as BaRT.

Two concepts come into play with BaRT — exploit and explore. The combination of 'exploit' and 'explore' is the key to Spotify's recommendation. While exploiting, Spotify makes use of every activity produced by a user. Exploit usually makes use of the user's listening history, songs skipped, playlists the user has created, social media activity on platforms and even one's location to recommend music.

Spotify studies the rest of the world. It starts searching for playlists and artists similar to your listening taste and also looking at the popularity of the artists in that area whom you haven't heard of or any other related works.

By taking all this data that has been collected from you over the decade, or as long as you have been on the Spotify, it presents their user with 'Decade Wrapped Playlists', and by taking the data set of different genres one has listened to over the decade will become the 'Best of the Decade For You'.

Spotify's algorithm looks at the duration of the time one has spent on a song, and if it is for more then 30 seconds, then the platform takes it as a check on their recommendations. The longer one spends on a song or a playlist, the better their suggestions will get.

To recommend a new artist Spotify analysing the audio itself by training the audio analysis algorithm to learn to recognise different desirable characters to music. Spotify experiments could identify various aspects of songs like distorted guitars and recommend based on that rather than Just the genre For automatic playlist continuation the feature analyses the songs in your playlists and tries to predict the music that can be played next after your song finishes.

2) Saavn augments classical collaborative filtering approaches, such as Matrix Factorization and Word2Vec, with deep learning based models and uses using a hybrid system of audio similarity and users' listening patterns to recommend music

3) Last.fm creates a "station" of recommended songs by observing what bands and individual tracks the user has listened to on a regular basis and comparing those against the listening behavior of other users. Last.fm will play tracks that do not appear in the user's library, but are often played by other users with

similar interests.

As this approach leverages the behavior of users, it is an example of a collaborative filtering technique.

4) Pandora uses the properties of a song or artist (a subset of the 400 attributes provided by the Music Genome Project) to seed a "station" that plays music with similar properties.

User feedback is used to refine the station's results, deemphasizing certain attributes when a user "dislikes" a particular song and emphasizes other attributes when a user "likes" a song. This is an example of a content-based approach.

### 1.4 Limitations

1. The genre is strictly restricted to limited number so beyond those other genre classifications is not possible

2. Classification is not strictly attained Some Song may be classified under more than one genre

3. The algorithms used are limited and many more efficient algorithms are available which can be incorporated for better accurate classification.

4. Certain datasets used are beyond 10 years old hence the latest genres are not included hence the classification of the same would be a constraint

## 2. SYSTEM ANALYSIS

### 2.1 Existing System

**1. A Comparative Study on Content-Based Music Genre Classification@ Tao Li, Mitsunori Ogihara and Qi Li**

The Authors was this paper proposed a new method of feature extraction, DWCHs which can capture the local and global information of music signals simultaneously by computing histograms on their Daubechies wavelet coefficients

**2. Machine Learning Evaluation for Music Genre Classification of Audio Signals by Chetna Dabas, Adtya Agrawal**

The authors of this paper proposed a model for audio signal-based music classification onvarious genres of music like pop, blues, metal, country, classical, disco, jazz and hip-hop. They considered different audio features like MFCC (Mel Frequency Spectral Coefficients), Delta, Delta-Delta and temporal aspects for

processing the data. They found that their proposed model performed better than existing models in terms of accuracy by 95%.

**3. Comparing Fuzzy Rule Based Approaches for Music Genre Classification by Heerde, F., Vatolkin, I., & Rudolph, G**

The authors of the paper did research on the fuzz rules-based music genre classification which offers advantage of understandability for end users, in particular in combination with carefully designed semantic features. They performed study on three approaches which operate on fuzzy rules: a complete search of primitive rules, an evolutionary approach, and fuzzy pattern trees. They found out that these methods have high error rates. Fuzzy Trees outperformed any other approach with a balanced validation error rate of 36.15% across all genres.

**4. Music Genre Classification using Machine Learning Techniques@ Bahuleyan, Hareesh**

The author used various models to perform music genre classification and The CNN based model and XGBoost were determined to be the best feature-based classifiers along with reporting the most important features.

**5. Music Genre Classification and Recommendation by Using Machine Learning Techniques By A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk and N. Aydin**

The authors of the paper extracted the acoustic features of music using digital signal processing methods and using various methods, performed music genre classification. The features extracted are used in classification in order to determine the most efficient algorithm. The algorithms used are KNN, Naive Bayes, Random Forests and SVM. They make use of the GTZAN dataset. According to the study, SVMs attained the highest performance accuracy as compared to the rest.

## 2.2 Proposed System

Aim is to compare various existing classification models which work using both supervised and unsupervised learning techniques in order to understand which model performs better. We have taken into consideration some widely used models such as SVM, KNN, XGBoost, Logistic Regression, Random Forest, Decision Trees, Naive Bayes and compared their accuracy to understand which one of them performs better. We have used GTZAN dataset for this purpose which has an extensive collection of music belonging to various categories like pop, blues, metal, country, classical, disco, jazz and hip-hop. We also then want to apply this and create a recommender system that first classifies a song into a genre and then recommends songs most similar to it

### 2.2.1 Benefits of Proposed System

- Our project involves comparison of the most prominent machine learning algorithms which ensures

the accuracy in the result

- Our project has used various regression techniques to find which is the best model and this provides an opportunity for better comparison and ensuring better accuracy in the result
- Our project would cut off the need to manually enter the genres and it automatically classifies music into its genres-based on various different features.
- The accuracy of our project is also way high compared to the pre-existing models
- Our datasets are vast in its genre collection
- Our project results are well supported with illustration accompanied by its implementation
- Our project has vast scope for future collaboration with any of the music systems or any music associated Systems involving genre classification

## 3. Requirement Specification

### 3.1 Hardware Requirements

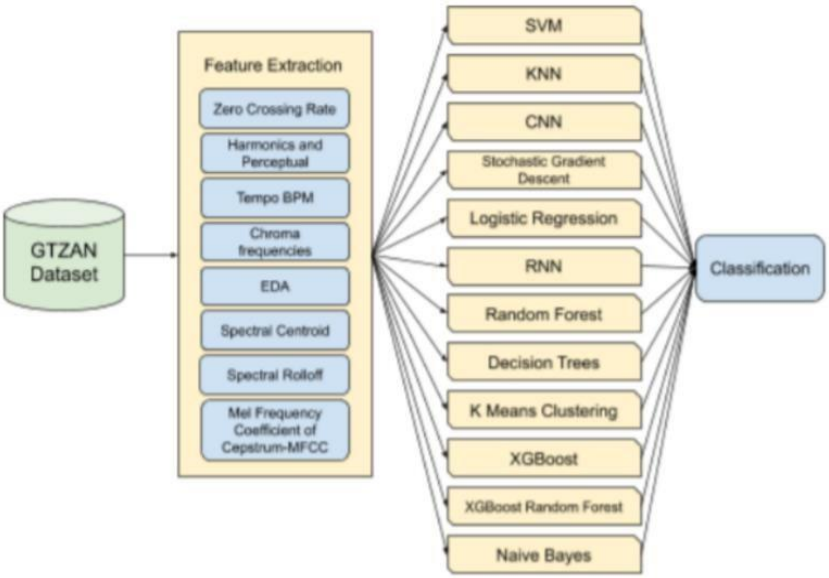
- Laptop

### 3.2 Software Requirements

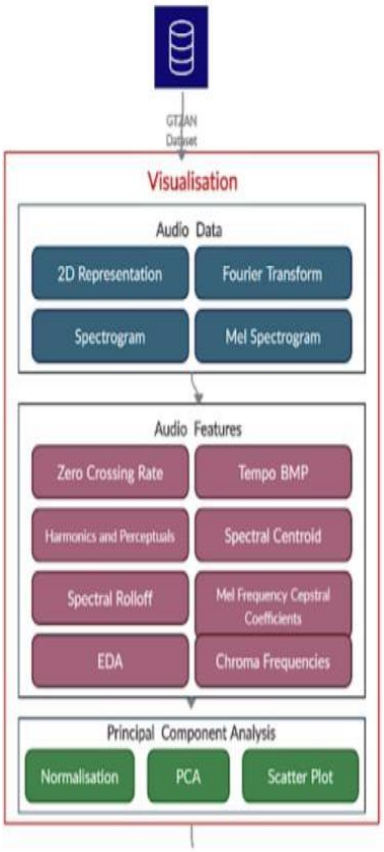
- Jupyter Notebook
- Anaconda
- Music Data

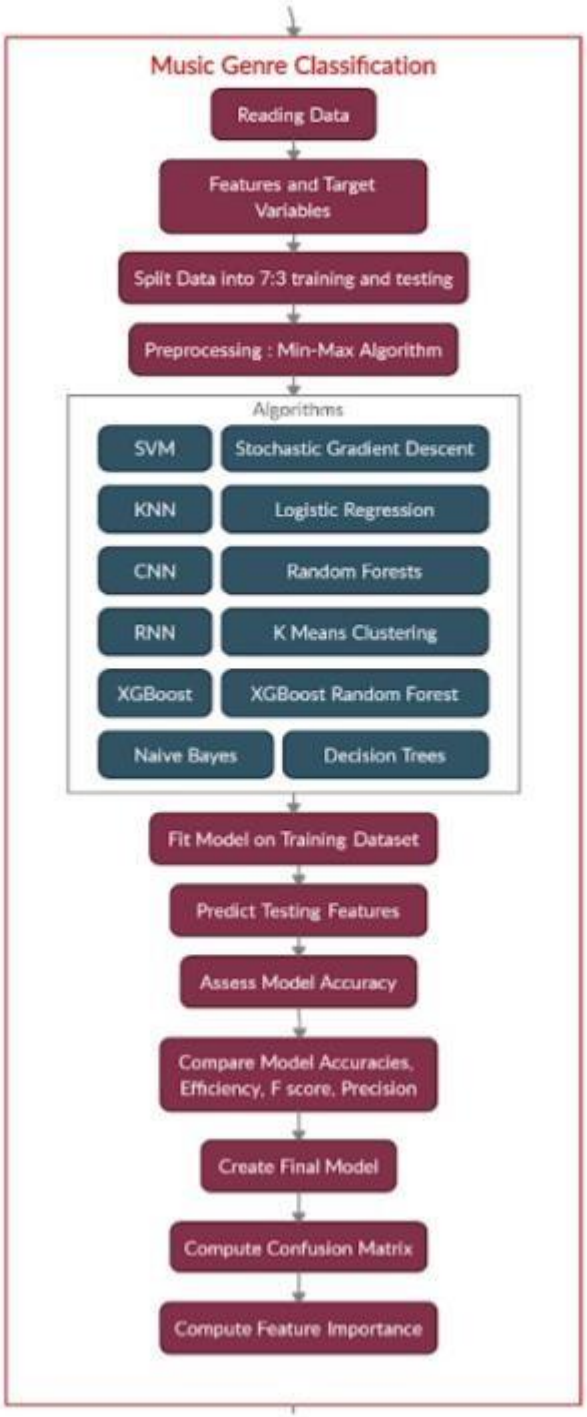
## 4. SYSTEM DESIGN AND SPECIFICATION

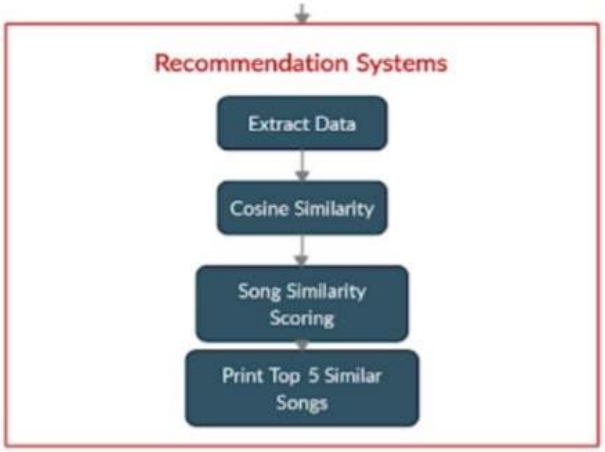
### 4.1 System Architecture



## 4.2 Detailed Design




Music Genre Classification
Reading Data
Features and Target Variables
Split Data into 7:3 training and testing
Preprocessing : Min-Max Algorithm
Algorithms
SVM
Stochastic Gradient Descent
KNN
Logistic Regression
CNN
Random Forests
RNN
K Means Clustering
XGBoost
XGBoost Random Forest
Naive Bayes
Decision Trees
Fit Model on Training Dataset
Predict Testing Features
Assess Model Accuracy
Compare Model Accuracies, Efficiency, F score, Precision
Create Final Model
Compute Confusion Matrix
Compute Feature Importance




The above diagram depicts the detailed architecture diagram which covers the detailed flow/architecture of -

1. Visualization
2. Music Genre Classification
3. Recommendation Systems

## 5. SYSTEM IMPLEMENTATION

### 5.1 Module Description

**There are 3 major parts of our project:**

**1. Visualization**

Audio data and it's representation as

- 2D representation
- Fourier Transformation
- Spectrogram

Audio Features

- Zero Crossing Rate
- Harmonics and Perceptual
- Tempo BPM (beats per minute)
- Chroma frequencies
- EDA: It performs analysis on the features_30_sec.csv file. This file contains the mean and variance for each audio file.
- Spectral Centroid: Spectral centroid is a feature used on a frequency domain and indicates the point of the center of gravity of the frequencies in the frequency bin.

- Spectral Roll off: It could be defined as the frequency value that corresponds to a certain ratio of the distribution in the spectrum.
- Mel Frequency Coefficient of Cepstrum-MFCC: The purpose of an MFCC is to adapt the cepstral coefficients to the human hearing system. Cepstral coefficients are linear scale.

Principal Component analysis
- Normalization
- PCA
- Scatter Plot

**2. Music Genre Classification**
- Reading data
- Split data to 7:3 (training and testing)
- Processing by different algos (KNN, XGBoost, Decision Trees etc.)
- Access accuracy of model
- Compare model accuracy, Efficiency, Precision
- Create Final Model
- Make confusion matrix

**3. Recommendation system**
- Extract data
- Cosine similarity
- Songs similarly scoring
- Print top 5 similar songs

## 5.2 Screen-Shots of the Modules

**1. Understanding the Audio Files**

We Explore the data and use the data reggae.00036.wav file to understand the features and visualize them. We calculate the sound value and sample rate and trim the audio file so that anything that leads and trailing silence are removed.

In [10]:
```
print('y:', y, '\n')
print('y shape:', np.shape(y), '\n')
print('Sample Rate (KHz):', sr, '\n')
```

```
y: [0.02072144 0.04492188 0.05422974 ... 0.06912231 0.08303833 0.08572388]

y shape: (661794,)

Sample Rate (KHz): 22050
```

In [11]:
```
print('Check Len of Audio:', 661794/22050)
```

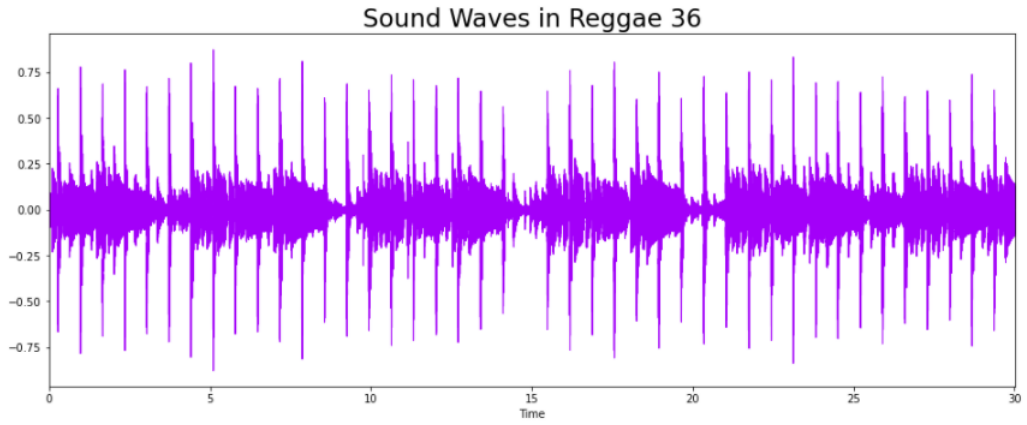
```
Check Len of Audio: 30.013333333333332
```

In [12]:
```
audio_file, _ = librosa.effects.trim(y)
```

In [13]:
```
print('Audio File:', audio_file, '\n')
print('Audio File shape:', np.shape(audio_file))
```

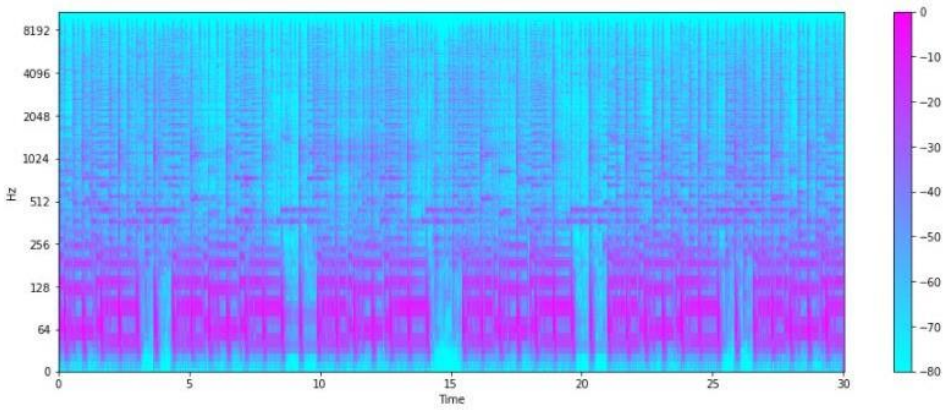
```
Audio File: [0.02072144 0.04492188 0.05422974 ... 0.06912231 0.08303833 0.08572388]

Audio File shape: (661794,)
```

In [14]:
```
plt.figure(figsize = (16, 6))
librosa.display.waveplot(y = audio_file, sr = sr, color = "#A300F9");
plt.title("Sound Waves in Reggae 36", fontsize = 23);
```



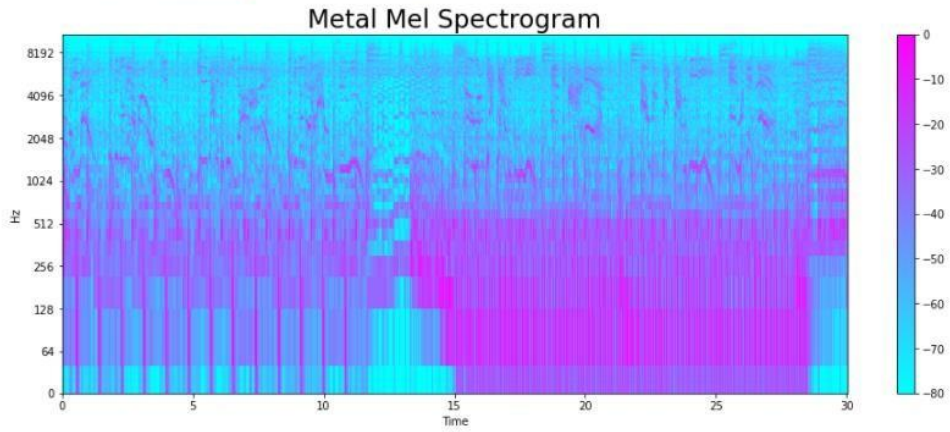
The Audio File is Visualized in the spectrogram, which is a representation of the spectrum of frequencies of signal with respect to the time We are using the same audio file mentioned above. Here, The amplitude spectrogram is converted into a decibel scale spectrogram.

```
In [17]: DB = librosa.amplitude_to_db(D, ref = np.max)
         plt.figure(figsize = (16, 6))
         librosa.display.specshow(DB, sr = sr, hop_length = hop_length, x_axis = 'time', y_axis = 'log',
                                  cmap = 'cool')
         plt.colorbar();
```



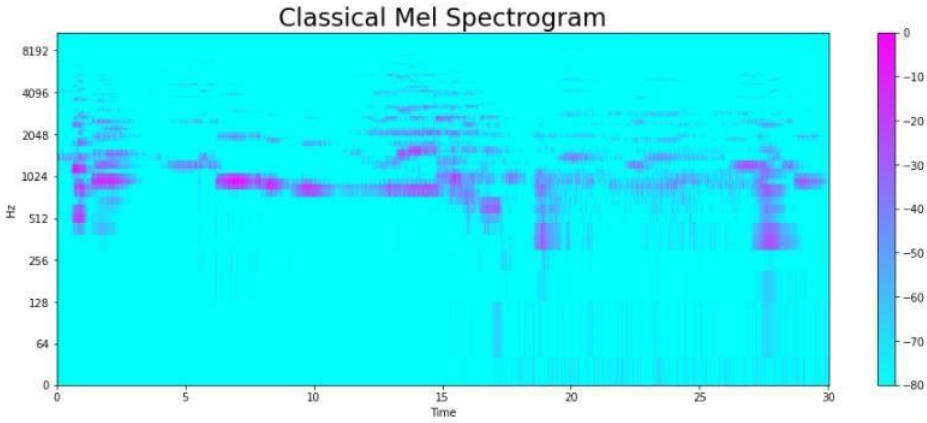

It is now Visualized in the Mel Spectogram.It is the same as a regular Spectrogram but the MEl scale is on the y axis. As mentioned in the code , We use metal.00036.wav.

```
In [18]: y, sr = librosa.load(f'{general_path}/genres_original/metal/metal.00036.wav')
         y, _ = librosa.effects.trim(y)


         S = librosa.feature.melspectrogram(y, sr=sr)
         S_DB = librosa.amplitude_to_db(S, ref=np.max)
         plt.figure(figsize = (16, 6))
         librosa.display.specshow(S_DB, sr=sr, hop_length=hop_length, x_axis = 'time', y_axis = 'log',
                                  cmap = 'cool');
         plt.colorbar();
         plt.title("Metal Mel Spectrogram", fontsize = 23);
```




The above Diagram Visualize metal music while on the below image we have visual classical music by using the audio file of classical.000360.wav

```
In [19]: y, sr = librosa.load(f'{general_path}/genres_original/classical/classical.00036.wav')
         y, _ = librosa.effects.trim(y)


         S = librosa.feature.melspectrogram(y, sr=sr)
         S_DB = librosa.amplitude_to_db(S, ref=np.max)
         plt.figure(figsize = (16, 6))
         librosa.display.specshow(S_DB, sr=sr, hop_length=hop_length, x_axis = 'time', y_axis = 'log',
                                  cmap = 'cool');
         plt.colorbar();
         plt.title("Classical Mel Spectrogram", fontsize = 23);
```

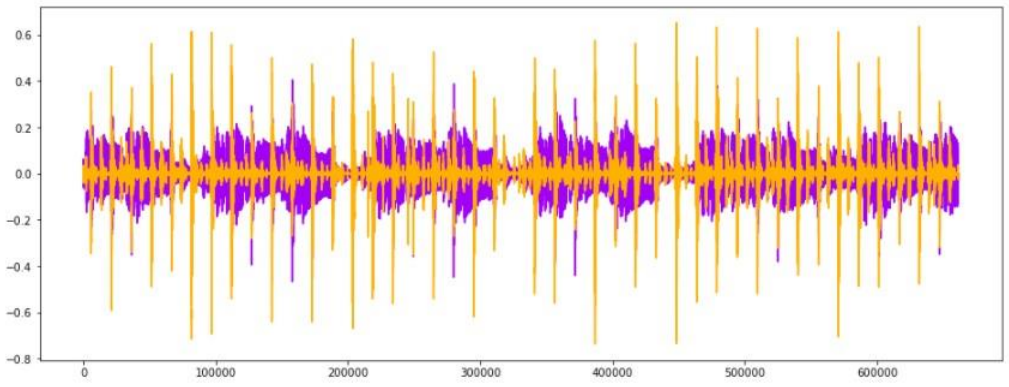



The following visualise the features of the audio file such as Zero Crossing Rate, Harmonics and Perceptual, Tempo BPM, Spectral Centroid, EDA Spectral Rolloff and the Mel-Frequency Cepstral Coefficients.

```
In [20]: zero_crossings = librosa.zero_crossings(audio_file, pad=False)
         print(sum(zero_crossings))
```

```
39232
```

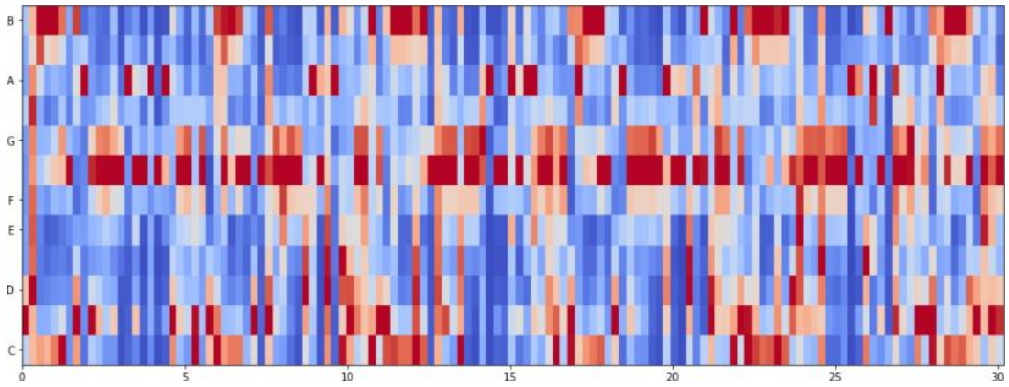
```
In [21]: y_harm, y_perc = librosa.effects.hpss(audio_file)

         plt.figure(figsize = (16, 6))
         plt.plot(y_harm, color = '#A300F9');
         plt.plot(y_perc, color = '#FFB100');
```



In [ 28] :

```
hop_length = 5660
chromagram = librosa . feature . chroma_stft {audio_f11e, sr=sr, hop_1ength=hop_1ength)
print( 'Chronogram shape :, ' chromagram . shape )
plt.title('BPM Boxplot for Genres', fontsize = 25)
plt.figure(figsize=(10, 6))
librosa.display.specshow(chromagram, x_axis='time', y_axis='chroma', hop_length=hop_length, cmap='coolvarm');
plt.yticks(fontsize = 10);
```

```
Chromogram shape: (12, 133)
```




In [ 29] :

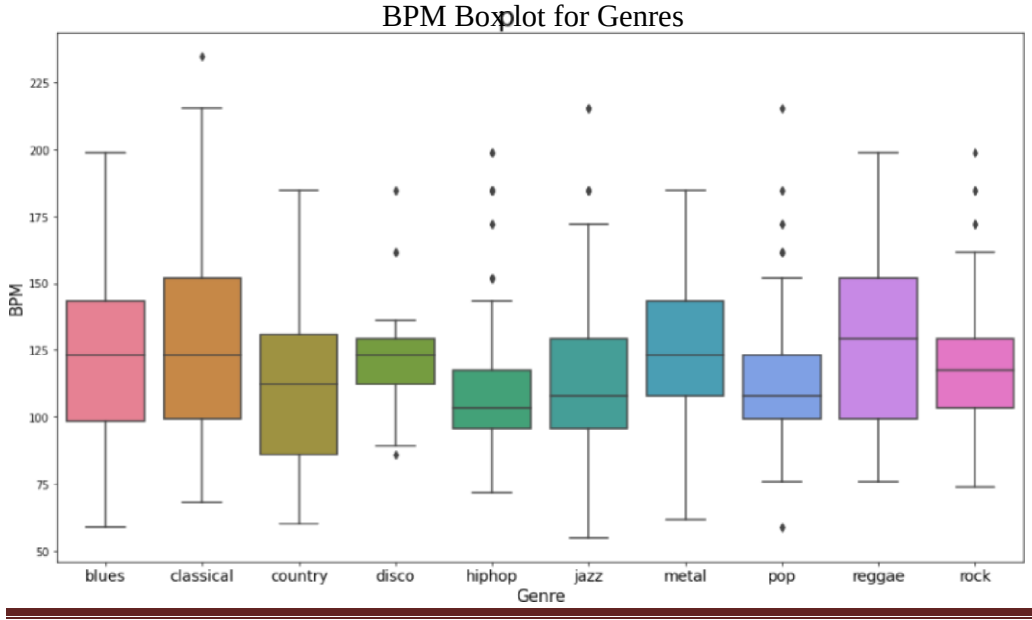
```
data = pd. read_csv(-fi ' (genera1_path},'features_30_sec . cs'v ' )
data . head( )
```

| filename length | chronia_atft_mean | chrorna_atft_var | rms_mean | rma_var | apectral_centroid_mean | apectral_centroid_var | apectral_bandwidth_mean | i |
|---|---|---|---|---|---|---|---|---|
| 0 blues.00000.wav 661794 | 0.H088 | 0.088757 | 0.130228 | 0.002B27 | 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.vyav 661794 | 0.365637 | 0.085275 | 0.175570 | 0.002746 | 1552.811865 | 156467.64336B | 1747.702312 | |
| 3 blues.00003.wav 661794 | 04M785 | 0.093999 | 0.¥41093 | 0.006346 | 1070.106615 | 184355.942417 | 1596.412872 | |
| 4 blues.00004.vyav 661794 | 0.308526 | 0.08784¥ | 0.09¥S29 | 0.002303 | 1835.004266 | 343399.939274 | 1748.172116 | |

5 royrs 60 columns

In [31]:

```
x = data[["label", "tempo"]]

I, ax = plt . s u b p lots(*igsiz e = (1fi, 9)) j
sns.boxplot(x = "label", y = "tempo", data = x, palette = 'husl');



p It . xlabel (”Ge n re" , fontsiz e = US)
p It . y label (”B PPi”, 4°ontsiz e - 15)
p It . savei ig (” B PPi Boxplot . jpg” )
```



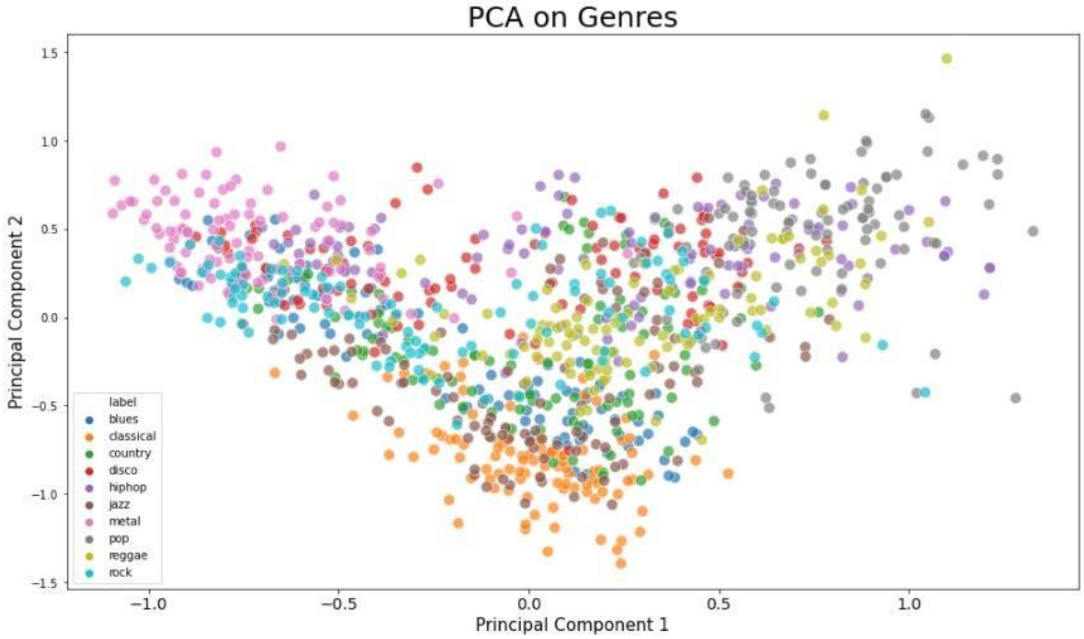
```
In [32]: from sklearn import preprocessing

         data = data.iloc[0:, 1:]
         y = data['label']
         X = data.loc[:, data.columns != 'label']
```

```
In [35]: cols = X.columns
         min_max_sc a ter = preproc ess i ng. FtinNaxSc al er ( )
         np_scaled = min_max_scaler.fit_transform(X)
         X = pd.DataFrame(np_scaled, columns = cols)


         pca = PCA(r_components=2)
         principalComponents = pca.fit_transform(X)
         principalDf = od.DataFrame(data = principalComponents, columns = ['principal comooment 1', 'principal component 2'])

         finalDf = pd.concat([principalDf, y], axis = 1)

         pea . explain en va riance ratio
```

```
In [36]: plt.figure(figsize = (16, 9))
         sn s . scatterplot (x - " p min ci p a l conponent 1", y - " p ni n ci p a l component 2" data - Min a lOf, hue " l a b e l", alpha - B. 7
                             s = i 8Jd ) ;

         plt.title('PCA on Genres, fomtsize = 25)
         pit.xticLs(fortsize = *4)
         plt.yticks(fontsize = 10);
         plt.xlabeI("Principal Component 1", fomtsize = 15)
         plt.ylabeI("Principal Component 2", fortsize = 15)
         plt.savefig("PCA Scattert.jpg")
```

Principal Component Analysis(PCA) is performed to be able to visualize the possible group of genres.

```
In [36]: plt.figure(figsize = (16, 9))
         sns.scatterplot(x = "principal component 1", y = "principal component 2", data = finalDf, hue = "label", alpha = 0.7,
                         s = 100);

         plt.title('PCA on Genres', fontsize = 25)
         plt.xticks(fontsize = 14)
         plt.yticks(fontsize = 10);
         plt.xlabel("Principal Component 1", fontsize = 15)
         plt.ylabel("Principal Component 2", fontsize = 15)
         plt.savefig("PCA Scattert.jpg")
```



## 2. Preprocessing using MIN-MAX Algorithm

The data is Normalised before further usage. The Min-Max Normalization algorithm was used to reach a normalised data set.This algorithm normalises the values such that the most minimum value becomes 0 and the most maximum value is transformed to 1. Other values that occur are transformed into decimal values between 0 and 1.

## 3. Music Genre Classification

Data is read from the features_3_sec.csv file to build a classifier that accurately predicts the genre for any audio file input. Create the target and feature variables, normalise the data and then split it into a 70% to 30% ratio as training : test data. On executing the algorithms and assessing the accuracy, we found that XGBooster was the best performing model.

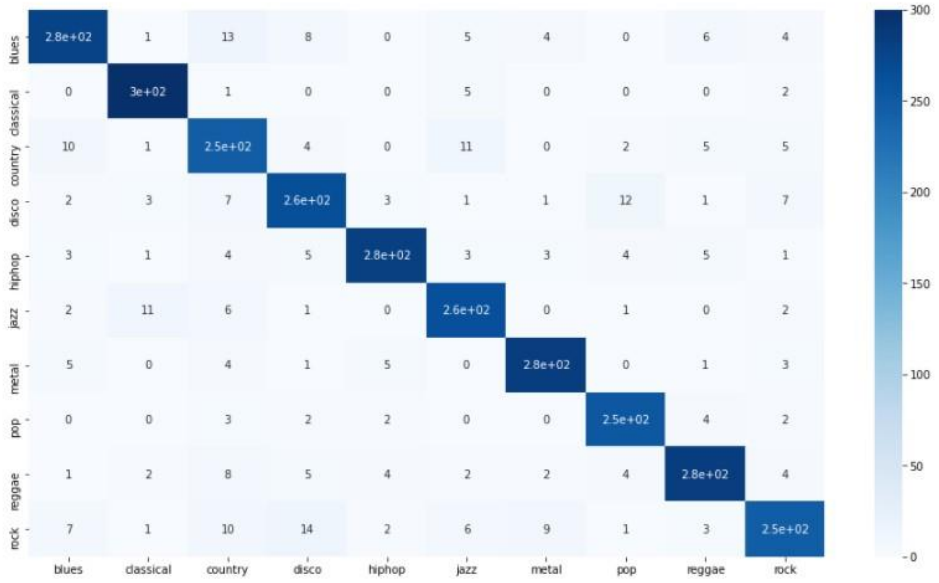
```
In [41]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

In [42]: def model_assess(model, title = "Default"):
             model.fit(X_train, y_train)
             preds = model.predict(X_test)
             print('Accuracy', title, ':', round(accuracy_score(y_test, preds), 5), '\n')

In [43]: nb = GaussianNB()
         model_assess(nb, "Naive Bayes")

         sgd = SGDClassifier(max_iter=5000, random_state=0)
         model_assess(sgd, "Stochastic Gradient Descent")

         knn = KNeighborsClassifier(n_neighbors=19)
         model_assess(knn, "KNN")

         tree = DecisionTreeClassifier()
         model_assess(tree, "Decission trees")

         rforest = RandomForestClassifier(n_estimators=1000, max_depth=10, random_state=0)
         model_assess(rforest, "Random Forest")

         svm = SVC(decision_function_shape="ovo")
         model_assess(svm, "Support Vector Machine")

         lg = LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinomial')
         model_assess(lg, "Logistic Regression")

         nn = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5000, 10), random_state=1)
         model_assess(nn, "Neural Nets")

         xgb = XGBClassifier(n_estimators=1000, learning_rate=0.05)
         model_assess(xgb, "Cross Gradient Booster")

         xgbrf = XGBRFClassifier(objective= 'multi:softmax')
         model_assess(xgbrf, "Cross Gradient Booster (Random Forest)")

         Accuracy Naive Bayes : 0.51952

         Accuracy Stochastic Gradient Descent : 0.65532

         Accuracy KNN : 0.80581

         Accuracy Decission trees : 0.64531

         Accuracy Random Forest : 0.81415

         Accuracy Support Vector Machine : 0.75409

         Accuracy Logistic Regression : 0.6977

         Accuracy Neural Nets : 0.67334
```

XGBooster to be the model with highest accuracy - 90%(approx)

This Diagram below is the Confusion Matrix for XGBooster Algorithm

```
In [44]: xgb = XGBClassifier(n_estimators=1000, learning_rate=0.05)
xgb.fit(X_train, y_train)


preds = xgb.predict(X_test)

print('Accuracy', ':', round(accuracy_score(y_test, preds), 5), '\n')
confusion_matr = confusion_matrix(y_test, preds)
plt.figure(figsize = (16, 9))
sns.heatmap(confusion_matr, cmap="Blues", annot=True,
            xticklabels = ["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop", "reggae", "rock"],
           yticklabels=["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop", "reggae", "rock"]);
plt.savefig("conf matrix")
```

```
[17:44:50] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.5.0/src/learner.cc:1115: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly s
et eval_metric if you'd like to restore the old behavior.
Accuracy : 0.90224
```



Feature importance along with their weights-screenshots. These are the Top 5 Most Features that determine the Music Genre.

```
In [48]: eli5.show_weights(estimator=perm, feature_names = X_test.columns.tolist())
```

Out[48]:

| Weight | Feature |
|---|---|
| 0.1205 ± 0.0095 | perceptr_var |
| 0.0416 ± 0.0031 | perceptr_mean |
| 0.0390 ± 0.0049 | mfcc4_mean |
| 0.0345 ± 0.0044 | chroma_stft_mean |
| 0.0339 ± 0.0062 | harmony_mean |
| 0.0280 ± 0.0065 | harmony_var |
| 0.0228 ± 0.0049 | mfcc9_mean |
| 0.0208 ± 0.0049 | mfcc6_mean |
| 0.0181 ± 0.0024 | rms_var |
| 0.0174 ± 0.0026 | mfcc3_mean |
| 0.0148 ± 0.0031 | spectral_bandwidth_mean |
| 0.0147 ± 0.0056 | mfcc11_mean |
| 0.0137 ± 0.0046 | tempo |
| 0.0116 ± 0.0036 | chroma_stft_var |
| 0.0113 ± 0.0026 | mfcc7_mean |
| 0.0109 ± 0.0038 | mfcc1_var |
| 0.0101 ± 0.0029 | mfcc3_var |
| 0.0089 ± 0.0057 | mfcc8_mean |
| 0.0089 ± 0.0020 | mfcc5_mean |
| 0.0072 ± 0.0038 | mfcc18_mean |
| … 38 more … | |

## 4. Recommender Systems

We scale the input data: features_30_sec.csv file. Using the cosine_similarity library, we find the best similar matches ranked in descending order for any given vector input. It calculates the similarity of two audio files, pairwise and generates a 1000 x 1000 matrix where every cell depicts the similarity of the corresponding audio files. We use the features_30_sec.csv file to predict the same. We write a function named find_similar_songs that takes the name of a song and returns the top five best matches for the input. 22

```
In [ 50] :
data = pd.read_csv(f'{gemeral_path}/features_30_sec.csv',index_col='filename')


labels = data[['label']]

$ Orop lodels from originot doto{rome
data = data.drop(columns=['length','label'])
data.head()

$ Srole tñe doto
data_scaled=preprocessimg.scale(data)
print('Scaled data type:', type(data_scaled))
```

```
Scaled data type: <class 'numpy.mdarray'>
```

```
In [ 52] :
similarity = cosine_similarity(data_scaled) print("Similarity shape:", similarity.shape)



sim_df_nwe s . he ad( )
```

```
Similarity shape: (1000, 1000)
```

Out[51]:

| Sitename | btuea.00000.way | bl | ues.00004.way | bluea.tOD2.nay | blues.00003.way | blues.00004wav | bluea.00005.way | bl ues.00006.way | blues.00007.wav | blues |
|---|---|---|---|---|---|---|---|---|---|---|

| i"itename | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| blues.00000.wav | 1.000000 | 0.049231 | 0.589618 | 0.284862 | 0.025561 | -0.346688 | -0.219483 | 4.167626 |
| 0.049231 | | 1.000000 | -0.086834 | 0.520803 | 0.080749 | 0.307856 | 0.318286 | 0.415258 |
| 0.58g6 T 8 | | -0.096834 | 1.000000 | 0.2104 T T | 0.400266 | -0.082019 | -0.028061 | 0.104446 |
| 0.284862 | | 0. 520903 | 0.21041T | 1.000000 | 0.126437 | 0.T34786 | 0.300746 | 0.324566 |
| 0.02556 T | | 0.080749 | 0. 400266 | 0.T26437 | 1.000000 | 0.556066 | 0.482195 | 0.623455 |

5 rows 1000 columns

On searching for songs similar to pop.00019.wav, we get the following list:

In [52]:
```
def find_similar_songs(name):
    # Find songs most similar to another song
    series = sim_df_names[name].sort_values(ascending = False)

    # Remove cosine similarity == 1 (songs will always have the best match with themselves)
    series = series.drop(name)

    # Display the 5 top matches
    print("\n*******\nSimilar songs to ", name)
    print(series.head(5))
```

In [53]:
```
find_similar_songs('pop.00019.wav')

ipd.Audio(f'{general_path}/genres_original/pop/pop.00019.wav')
```

```
*******
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
```

Out[53]: ▶ 0:00 / 0:30

In [54]:
```
ipd.Audio(f'{general_path}/genres_original/pop/pop.00023.wav')
```

Out[54]: ▶ 0:00 / 0:30

In [55]:
```
ipd.Audio(f'{general_path}/genres_original/pop/pop.00034.wav')
```

Out[55]: ▶ 0:00 / 0:30

On searching for songs similar to metal.00002.wav, we get the following list:

```
In [59]: find_similar_songs('metal.00002.wav')

         ipd.Audio(f'{general_path}/genres_original/metal/metal.00002.wav')

         *******
         Similar songs to  metal.00002.wav
         filename
         metal.00028.wav    0.904367
         metal.00059.wav    0.896096
         rock.00018.wav     0.891910
         rock.00017.wav     0.886526
         rock.00016.wav     0.867508
         Name: metal.00002.wav, dtype: float64
Out[59]: ▶ 0:00 / 0:30 ——— 🔊 ⋮

In [60]: ipd.Audio(f'{general_path}/genres_original/metal/metal.00028.wav')
Out[60]: ▶ 0:00 / 0:30 ——— 🔊 ⋮

In [61]: ipd.Audio(f'{general_path}/genres_original/metal/metal.00059.wav')
Out[61]: ▶ 0:00 / 0:30 ——— 🔊 ⋮

In [62]: ipd.Audio(f'{general_path}/genres_original/rock/rock.00018.wav')
Out[62]: ▶ 0:00 / 0:30 ——— 🔊 ⋮
```

### 5.3 Result Analysis

| Algorithm | Accuracy | Efficiency | F-Score | Precision |
|---|---|---|---|---|
| SVM | 0.75409 | 0.75410 | 0.75210 | 0.75209 |
| K-Nearest Neighbors | 0.80581 | 0.80567 | 0.79432 | 0.79338 |
| Convolution Neural Networks | 0.67734 | 0.67723 | 0.66478 | 0.66473 |
| Stochastic Gradient Descent | 0.65532 | 0.65527 | 0.64213 | 0.64205 |
| Logistic Regression | 0.69970 | 0.69966 | 0.68778 | 0.68764 |
| Recurrent Neural Networks | 0.69012 | 0.69034 | 0.68982 | 0.68991 |
| Random Forest | 0.81415 | 0.81427 | 0.80126 | 0.80113 |
| Decision Trees | 0.64631 | 0.64660 | 0.63990 | 0.63984 |

| | | | | |
|---|---|---|---|---|
| K-Means Clustering | 0.72146 | 0.72123 | 0.71291 | 0.71278 |
| Cross Gradient Booster | 0.90224 | 0.90222 | 0.90121 | 0.90125 |
| Cross Gradient Booster( Random Forest) | 0.74875 | 0.74863 | 0.73245 | 0.73249 |
| Naive Bayes | 0.51952 | 0.51940 | 0.50349 | 0.50412 |

Accuracy, efficiency, f-score and precision are few of the most important parameters that are used to evaluate the performance of the algorithms used and decide which one to finally incorporate in the project. We can deduce from the above table that the algorithm with the highest accuracy, efficiency, f-score and precision is the Cross Gradient Booster Algorithm and the one with the lowest is Naive Bayes Algorithm. Hence it is ideal to choose the Cross Gradient Booster for building our model.

## 6. CONCLUSION AND FUTURE ENHANCEMENTS

As per the accuracy results, we find that XGBooster has the highest accuracy of 90%, and then Random Forests and KNN have an accuracy of 81% and 80% respectively. We build our final model using the XGBooster algorithm and perform feature importance.We find perceptr_var, perceptr_mean and mfcc4_mean to be features that have the highest weights of importance. Finally, using the cosine_similarity library we find the best similar matches to the given input audio.

## 7. References

[1] Li, T., Ogihara, M., & Li, Q. (2003, July). A comparative study on content-based music genre classification. In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 282-289).

[2] Dabas, C., Agarwal, A., Gupta, N., Jain, V., & Pathak, S. (2020). Machine Learning Evaluation for Music Genre Classification of Audio Signals. International Journal of Grid and High Performance Computing (IJGHPC), 12(3), 57-67.

[3] Heerde, F., Vatolkin, I., & Rudolph, G. (2020, April). Comparing Fuzzy Rule Based Approaches for Music Genre Classification. In International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar) (pp. 35-48). Springer, Cham.

[4] Bahuleyan, Hareesh. (2018). Music Genre Classification using Machine Learning Techniques.

[5] A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk and N. Aydin, "Music Genre Classification and Recommendation by Using Machine Learning Techniques," 2018 Innovations in Intelligent Systems and Applications Conference (ASYU), Adana, 2018, pp. 1-5, doi: 10.1109/ASYU.2018.8554016.

[6] Costa, Y.M., Oliveira, L.S., Koerich, A.L., Gouyon, F. and Martins, J.G., 2012. Music genre classification using LBP textural features. Signal Processing, 92(11), pp.2723-2737. 26

[7] Chun Pui Tang, Ka Long Chui, Ying Kin Yu, Zhiliang Zeng, Kin Hong Wong (2018). Music Genre classification using a hierarchical Long Short Term Memory (LSTM) model. ICMR 18, 11- 14 June 2018, ,Yokohama,Japan