Music Genre Classification Using Machine Learning Techniques

*Project Report for *Indian Institute of Technology, Bombay* - CS 419: Introduction to Machine Learning (2023)

N. Kolhe
Department of Physics
Indian Institute of Technology, Bombay
Mumbai, Maharashtra
210260034@iitb.ac.in

P. Aggarwal
Department of Physics
Indian Institute of Technology, Bombay
Mumbai, Maharashtra
210260038@iitb.ac.in

T. Patil
Department of Physics
Indian Institute of Technology, Bombay
Mumbai, Maharashtra
210260058@iitb.ac.in

***Abstract*—A music genre is a conventional category that identifies some pieces of music as belonging to a shared tradition or set of conventions. It is to be distinguished from musical form and musical style. Music can be divided into different genres in many different ways. The popular music genres are Pop, Hip-Hop, Rock, Jazz, Blues, Country and Metal.**
Categorizing music files according to their genre is a challenging task in the area of music information retrieval (MIR). Automatic music genre classification is important to obtain music from a large collection. It finds applications in the real world in various fields like automatic tagging of unknown piece of music (useful for apps like Saavn, Wynk etc.).
Companies nowadays use music classification, either to be able to place recommendations to their customers or simply as a product. Determining music genres is the first step in the process of music recommendation. Most of the current music genre classification techniques use machine learning techniques. The same principles are applied in Music Analysis also. Machine Learning techniques have proved to be quite successful in extracting trends and patterns from the large pool of data.
In this project, we will build a complete music Genre classifier using various machine learning algorithms and then choosing the model which gives best accuracy.

I. Introduction

A. What is Music Genre

A music genre is a classification system that classifies music into different styles. It's the art of incorporating instrumental and vocal tones in a structured manner that gives the music its distinctive character.
As a result, all artistic compositions that belong to the same music genre share some similarities in form or style.
The word genre is used in other forms of art, including literature, television, cinema, and other artistic creation types. It combines pieces of work that fit under a specific category after analyzing and highlighting the most distinctive elements. In addition to pure genres, crossover genres are also popular. These combine several elements from different elements to create a new style that appeals to a wider audience. This technique is widely used in popular music. However, it's quite common for a musical piece to belong to several genres at the same time.
Music genres can refer to when the music was composed, its style, the instruments used and their functions, and the music's geographical origin. In other words, there's no one clear comprehensive definition that explains what different music genres are. Even within a single system, there is some disagreement about the interpretations of the elements used to categorize music composition.

B. Major music genres

- **Art music** :- Art music primarily includes classical traditions, including both contemporary and historical classical music forms.
- **Popular music** :- Popular music is any musical style accessible to the general public and disseminated by the mass media.
- **Electronic music** :- Electronic music is music that employs electronic musical instruments, digital instruments, or circuitry-based music technology in its creation.
- **Metal music** :- It has a rougher style and heavier sound than other forms of rock music, with notable subgenres such as thrash metal, death metal and black metal.
- **Hip Hop music** :- It can be broadly defined as a stylized rhythmic music that commonly accompanies rapping, a rhythmic and rhyming speech that is chanted.
- **Jazz** :- Jazz is characterized by swing and blue notes, complex chords, call and response vocals, polyrhythms and improvisation. Jazz has roots in European harmony and African rhythmic rituals.
- **Punk** :- The aggressiveness of the musical and performative style, based on structural simplicity and the vigorous rhythms of rock'n'roll style, reinforced the challenging and provocative character, within the universe of modern music.
- **Rock music** :- It has its roots in 1940s and 1950s rock and roll, a style that drew directly from the blues and rhythm and blues genres of African-American music and from country music.
- **Religious music** :- Religious music (also sacred music) is music performed or composed for religious use.

C. How to classify different songs

The general classification algorithm is divided into music feature extractor and classifier.
The feature extraction of the system is constructed by timbre feature extraction and rhythm feature extraction. 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. We'll be using **librosa** for analyzing and extracting features of an audio signal.
The classifier is formed by back propagation neural network (BPNN). But in this project, we will build a complete music genre classifier using various machine learning algorithms without using deep learning, and then choose the model which gives the best accuracy.

II. Dataset and Methodology

The dataset we will use is named the GTZAN genre collection dataset which is a very popular audio collection dataset. It contains :

- **Genres original** - A collection of 10 genres with 100 audio files each, all having a length of 30 seconds (the famous GTZAN dataset, the MNIST of sounds)
- **Images original** - A visual representation for each audio file.One way to classify data is through neural networks. Because NNs usually take in some sort of image representation, the audio files were converted to Mel Spectrograms to make this possible.
- **2 CSV files** - Containing features of the audio files. One file has for each song (30 seconds long) a mean and variance computed over multiple features that can be extracted from an audio file. The other file has the same structure, but the songs were split before into 3 seconds audio files (this way increasing 10 times the amount of data we fuel into our classification models).

The classes to which audio files belong are Blues, Hip-hop, classical, Pop, Disco, Country, Metal, Jazz, Reggae, and Rock. One can easily find the dataset over Kaggle and can download it from GTZAN Dataset

III. Data Visualization and Data Analysis

For Data Visualization, we have excessively used various functionalities provided by **librosa** package. The functionalities are :

- **Waveform** :- To visualize the amplitude envelope of the waveform, we use **librosa.display.waveshow** function.

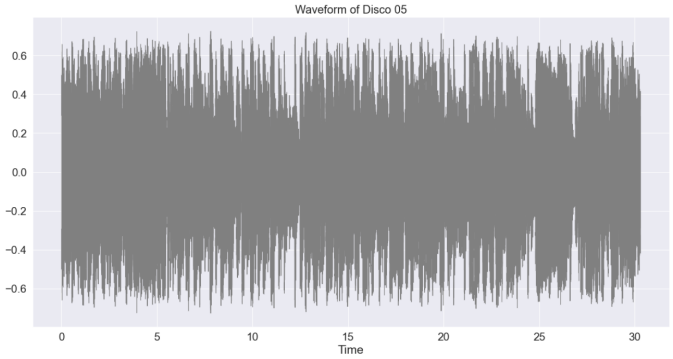

Fig. 1. Waveform of Disco 05

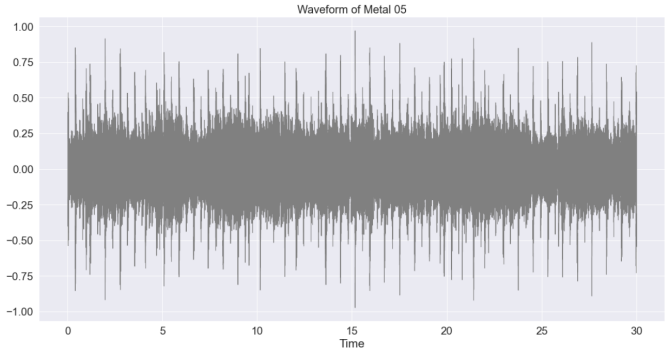

Fig. 2. Waveform of Metal 05

- **Waveform in frequency domain** :- To visualize in frequency domain we fourier transform the amplitude vs time data into Magnitude vs frequency. We use **librosa.stft** function.

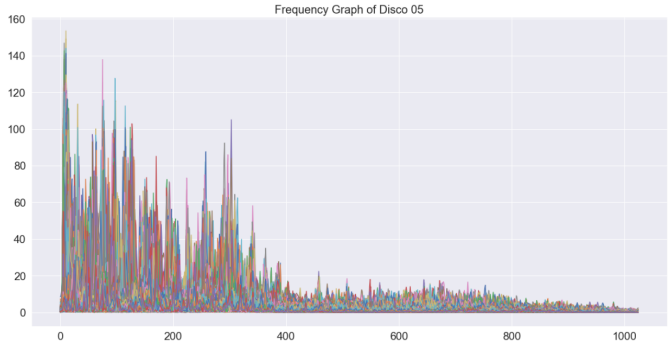

Fig. 3. Frequency Graph of Disco 05

- **Spectrogram** :- It is visual representation of spectrum of frequencies of sound at different points of time. Spectrograms are sometimes called sonographs, voiceprints, or voicegrams. We use **librosa.display.specshow** function.

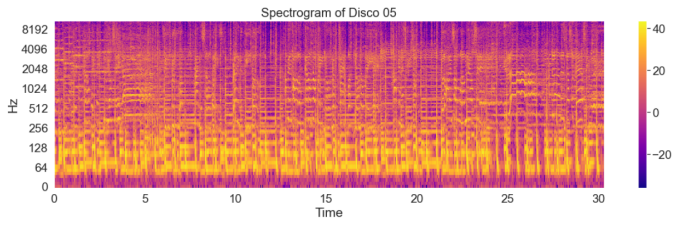

Fig. 4. Spectorgram of Disco 05

- **Spectral Centroids** :- It indicates where the " center of mass" for a sound is located and is calculated as the weighted mean of the frequencies present in the sound. We use **librosa.feature.spectral centroid** function.

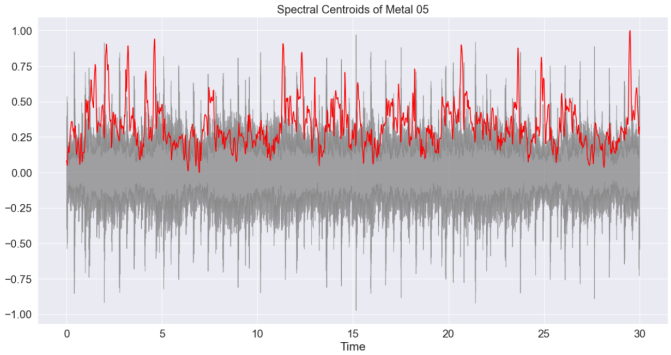

Fig. 5. Spectral Centroids of Metal 05

- **Spectral Rolloff** :- It is a measure of the shape of the signal. It represents the frequency below which a specified percentage of the total spectral energy, e.g., 85 percent lies. We use **librosa.feature.spectral rolloff** function.

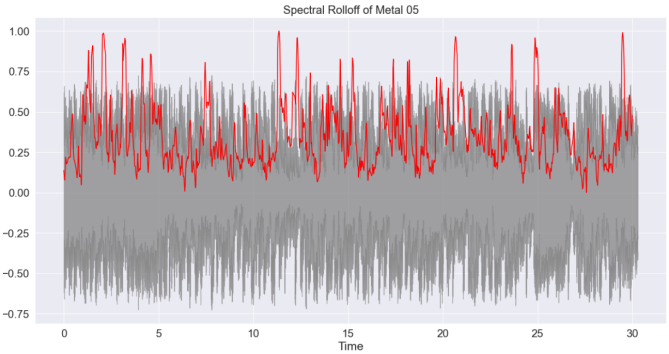

Fig. 6. Spectral Rolloff of Metal 05

- **Chromagram** :- Chromagram is a plot where spectrum of frequencies are classified into 12 chroma of an octave. We use **librosa.feature.chroma stft** function.

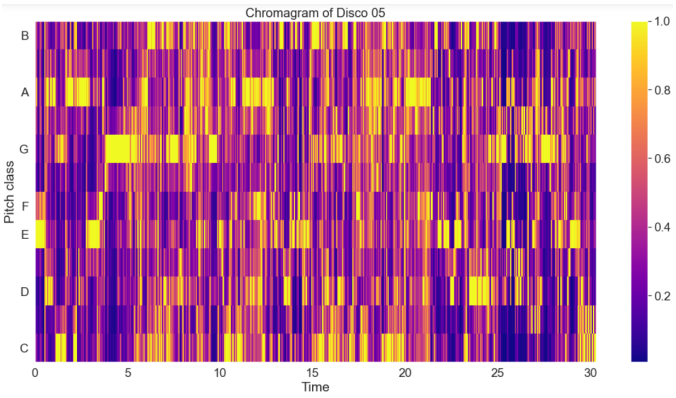

Fig. 7. Chromagram of Disco 05

Now lets perform EDA on the file containing 1000 entries, where each entry has features of 30 second long audio files of different genres. We first clean the data and check for any null values.

- **Zero Crossing** :- It counts the number of times the waveform crosses the value 0. We visualize this using violin plots where each violin represents the number of zero crossings for a particular genre of music.

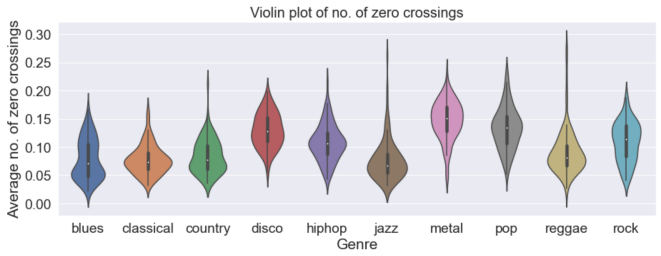

Fig. 8. Zero Crossings of different genres

- **Beats Per Minute** :- It counts the number of beats that occur in one minute of audio. We visualize this using violin plots where each violin represents the number of Beats Per Minute(BPM) for a particular genre of music.

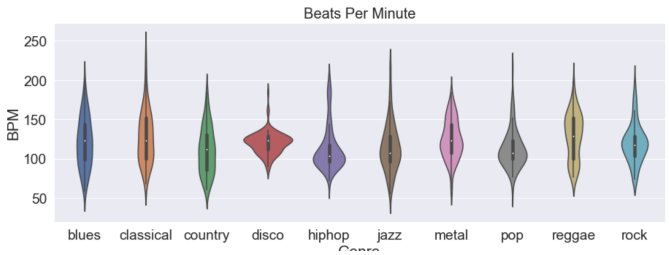

Fig. 9. Beats Per Minute of different genres

- **Correlation Heatmap** :- Using correlation heatmaps we can visually determine the relation between multiple variables. Here we are selecting those variable which represents mean value of some parameter.

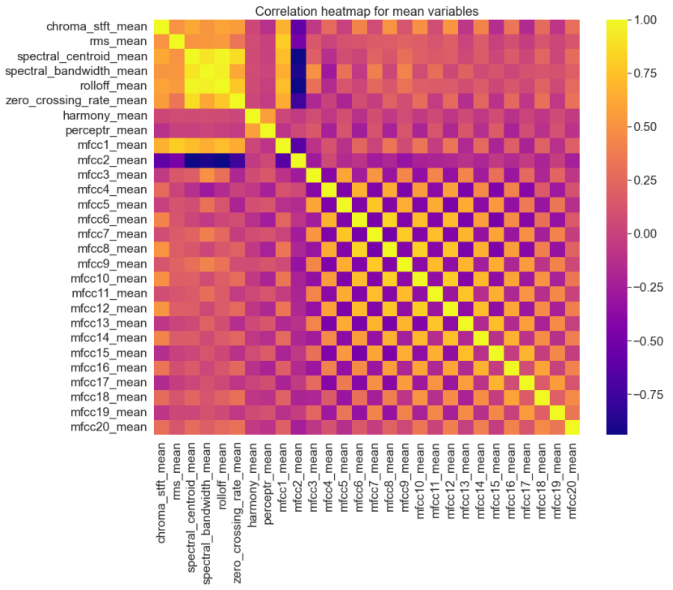

Fig. 10. Correlation Heatmap

IV. Machine Learning Models

Firstly, to use data, we scaled it to normalised data using the **MinMaxScaler** feature from the **sklearn.preprocessing** module.

A. Splitting the dataset

Ideally, the dataset is split into train, validation and test datasets as described below:

- **Training dataset** :- It is the set of data that is used to train and make the model learn the hidden features/patterns in the data.
- **Validation dataset** :- The validation set is a set of data, separate from the training set, that is used to validate our model performance during training.
- **Test dataset** :- The test set is a separate set of data used to test the model after completing the training.

However, sometimes, because of bias in dat split, the model may perform very well on the train and validaion set but fail to perform satisfactorily on the test set and this might give a wrong estimate of the model's accuracy. To overcome this, we have used **K-fold cross-validation** to get the accuracy of different models.

B. Choosing different machine learning models

We will try the following classification models and choose the one which gives the best accuracy using K-fold cross-validation:

- **Logistic Regression** :- Logistic Regression is used to determine if an input belongs to a certain group or not.
- **Decision Tree** :- Decision trees are also classifiers that are used to determine what category an input falls into by traversing the leaves and nodes of a tree.
- **Random Forest** :- Random forest is a collection of many decision trees from random subsets of the data, resulting in a combination of trees that may be more accurate in prediction than a single decision tree.
- **Support Vector Classifier** :- SVM, or Support Vector Machines create coordinates for each object in an n-dimensional space and uses a hyperplane to group objects by common features.
- **K Nearest Neighbours** :- The k Nearest Neighbors technique involves grouping the closest objects in a dataset and finding the most frequent or average characteristics among the objects.
- **Naive Bayes** :- Naive Bayes is an algorithm that assumes independence among variables and uses probability to classify objects based on features.

C. K-Fold Cross Validation

We estimated the accuracy of all these models using 5-fold cross-validation. Also, since the data is organized in order of music genres, if we simply split the data into 5 fold, each fold will have all the data from only two genres! If we use such folds, we will always get almost zero accuracy as the training set would not have any instances similar to those in the test set in each iteration. As a result, the accuracy of all models would be zero. This can be easily seen from the calculated accuracy using **shuffle=False** while implementing Kfold:

```
TRAIN: [1998 1999 2000 ... 9987 9988 9989] TEST: [   0    1    2 ... 1995 1996 1997]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [1998 1999 2000 ... 3993 3994 3995]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [3996 3997 3998 ... 5991 5992 5993]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [5994 5995 5996 ... 7989 7990 7991]
TRAIN: [   0    1    2 ... 7989 7990 7991] TEST: [7992 7993 7994 ... 9987 9988 9989]
[0. 0. 0. 0. 0.]
LogisticRegression()Accuracy: 0.0
TRAIN: [1998 1999 2000 ... 9987 9988 9989] TEST: [   0    1    2 ... 1995 1996 1997]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [1998 1999 2000 ... 3993 3994 3995]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [3996 3997 3998 ... 5991 5992 5993]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [5994 5995 5996 ... 7989 7990 7991]
TRAIN: [   0    1    2 ... 7989 7990 7991] TEST: [7992 7993 7994 ... 9987 9988 9989]
[0.    0.    0.001 0.    0.   ]
DecisionTreeClassifier()Accuracy: 0.0002
TRAIN: [1998 1999 2000 ... 9987 9988 9989] TEST: [   0    1    2 ... 1995 1996 1997]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [1998 1999 2000 ... 3993 3994 3995]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [3996 3997 3998 ... 5991 5992 5993]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [5994 5995 5996 ... 7989 7990 7991]
TRAIN: [   0    1    2 ... 7989 7990 7991] TEST: [7992 7993 7994 ... 9987 9988 9989]
[0.     0.     0.001  0.0005 0.    ]
RandomForestClassifier()Accuracy: 0.0003000000000000003
TRAIN: [1998 1999 2000 ... 9987 9988 9989] TEST: [   0    1    2 ... 1995 1996 1997]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [1998 1999 2000 ... 3993 3994 3995]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [3996 3997 3998 ... 5991 5992 5993]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [5994 5995 5996 ... 7989 7990 7991]
TRAIN: [   0    1    2 ... 7989 7990 7991] TEST: [7992 7993 7994 ... 9987 9988 9989]
[0. 0. 0. 0. 0.]
SVC()Accuracy: 0.0
TRAIN: [1998 1999 2000 ... 9987 9988 9989] TEST: [   0    1    2 ... 1995 1996 1997]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [1998 1999 2000 ... 3993 3994 3995]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [3996 3997 3998 ... 5991 5992 5993]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [5994 5995 5996 ... 7989 7990 7991]
TRAIN: [   0    1    2 ... 7989 7990 7991] TEST: [7992 7993 7994 ... 9987 9988 9989]
[0. 0. 0. 0. 0.]
KNeighborsClassifier()Accuracy: 0.0
TRAIN: [1998 1999 2000 ... 9987 9988 9989] TEST: [   0    1    2 ... 1995 1996 1997]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [1998 1999 2000 ... 3993 3994 3995]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [3996 3997 3998 ... 5991 5992 5993]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [5994 5995 5996 ... 7989 7990 7991]
TRAIN: [   0    1    2 ... 7989 7990 7991] TEST: [7992 7993 7994 ... 9987 9988 9989]
[0.     0.     0.0015 0.     0.    ]
GaussianNB()Accuracy: 0.0003000000000000003
```

Fig. 11. Kfold validation using shuffle=False

Therefore, we used shuffle=True and random_state=1 to get a better estimate of the accuracy of the different models. K-fold cross validation ultimately helps preventing overfitting on training dataset caused due to bias during train-test split.

D. Model Training and Accuracy

Now we see how different models fit the data and then we will be able to find the model with best accuracy using K-Fold Cross Validation. Given below is a snippet of the code used to implement this:

```
LR = LogisticRegression()
DT = DecisionTreeClassifier()
RF = RandomForestClassifier()
SVM = SVC()
KNN = KNeighborsClassifier()
NB = GaussianNB()

models = [LR,DT,RF,SVM,KNN,NB]

for model in models:
    accuracy = np.zeros(5)
    i = 0
    kf = KFold(n_splits=5, shuffle=True, random_state=1)

    for train_index, test_index in kf.split(features):
        print("TRAIN:", train_index, "TEST:", test_index)
        features_train, features_test = features.iloc[train_index], features.iloc[test_index]
        labels_train, labels_test = labels[train_index], labels[test_index]
        model.fit(features_train, labels_train)
        labels_pred = model.predict(features_test)
        #print('Accuracy:' round(accuracy_score(labels_test, labels_pred), 5))
        accuracy[i] = round(accuracy_score(labels_test, labels_pred), 5)
        i = i + 1
    print(accuracy)
    print(str(model) + 'Accuracy: ' + str(accuracy.mean()))
```

Fig. 12. Implementing K-Fold Cross Validation

```
TRAIN: [   0    1    2 ... 9986 9987 9989] TEST: [   6    7   21 ... 9971 9972 9988]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [   4   16   17 ... 9983 9984 9986]
TRAIN: [   0    1    2 ... 9986 9988 9989] TEST: [   3    5    8 ... 9978 9980 9987]
TRAIN: [   0    2    3 ... 9986 9987 9988] TEST: [   1    9   10 ... 9977 9982 9989]
TRAIN: [   1    3    4 ... 9987 9988 9989] TEST: [   0    2   15 ... 9976 9979 9985]
[0.20571 0.22773 0.21421 0.23423 0.23674]
LogisticRegression()Accuracy: 0.22372399999999998
TRAIN: [   0    1    2 ... 9986 9987 9989] TEST: [   6    7   21 ... 9971 9972 9988]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [   4   16   17 ... 9983 9984 9986]
TRAIN: [   0    1    2 ... 9986 9988 9989] TEST: [   3    5    8 ... 9978 9980 9987]
TRAIN: [   0    2    3 ... 9986 9987 9988] TEST: [   1    9   10 ... 9977 9982 9989]
TRAIN: [   1    3    4 ... 9987 9988 9989] TEST: [   0    2   15 ... 9976 9979 9985]
[0.67868 0.65465 0.66266 0.66016 0.66567]
DecisionTreeClassifier()Accuracy: 0.664364
TRAIN: [   0    1    2 ... 9986 9987 9989] TEST: [   6    7   21 ... 9971 9972 9988]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [   4   16   17 ... 9983 9984 9986]
TRAIN: [   0    1    2 ... 9986 9988 9989] TEST: [   3    5    8 ... 9978 9980 9987]
TRAIN: [   0    2    3 ... 9986 9987 9988] TEST: [   1    9   10 ... 9977 9982 9989]
TRAIN: [   1    3    4 ... 9987 9988 9989] TEST: [   0    2   15 ... 9976 9979 9985]
[0.87788 0.87187 0.85586 0.86837 0.87688]
RandomForestClassifier()Accuracy: 0.870172
TRAIN: [   0    1    2 ... 9986 9987 9989] TEST: [   6    7   21 ... 9971 9972 9988]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [   4   16   17 ... 9983 9984 9986]
TRAIN: [   0    1    2 ... 9986 9988 9989] TEST: [   3    5    8 ... 9978 9980 9987]
TRAIN: [   0    2    3 ... 9986 9987 9988] TEST: [   1    9   10 ... 9977 9982 9989]
TRAIN: [   1    3    4 ... 9987 9988 9989] TEST: [   0    2   15 ... 9976 9979 9985]
[0.28078 0.28679 0.28629 0.27728 0.29479]
SVC()Accuracy: 0.28518599999999994
TRAIN: [   0    1    2 ... 9986 9987 9989] TEST: [   6    7   21 ... 9971 9972 9988]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [   4   16   17 ... 9983 9984 9986]
TRAIN: [   0    1    2 ... 9986 9988 9989] TEST: [   3    5    8 ... 9978 9980 9987]
TRAIN: [   0    2    3 ... 9986 9987 9988] TEST: [   1    9   10 ... 9977 9982 9989]
TRAIN: [   1    3    4 ... 9987 9988 9989] TEST: [   0    2   15 ... 9976 9979 9985]
[0.26777 0.25676 0.27878 0.26927 0.27277]
KNeighborsClassifier()Accuracy: 0.26907
TRAIN: [   0    1    2 ... 9986 9987 9989] TEST: [   6    7   21 ... 9971 9972 9988]
TRAIN: [   0    1    2 ... 9987 9988 9989] TEST: [   4   16   17 ... 9983 9984 9986]
TRAIN: [   0    1    2 ... 9986 9988 9989] TEST: [   3    5    8 ... 9978 9980 9987]
TRAIN: [   0    2    3 ... 9986 9987 9988] TEST: [   1    9   10 ... 9977 9982 9989]
TRAIN: [   1    3    4 ... 9987 9988 9989] TEST: [   0    2   15 ... 9976 9979 9985]
[0.42693 0.41141 0.41491 0.42292 0.42142]
GaussianNB()Accuracy: 0.41951799999999995
```

Fig. 13. Accuracy of different classification models

Clearly, **Random Forest Classifier** gives the highest accuracy.

V. MUSIC GENRE PREDICTION - RESULTS

Let us go ahead with Random Forest Classifier. We first split the data into training and test sets. Here, we've used 85% data for training the model and 15% of the data will be predicted based on it.

We then train the model on the training data and predict the labels for the test data features. The results are shown below:

```
In [29]: from sklearn.model_selection import train_test_split

         features_train, features_test, labels_train, labels_test = train_test_split(features,labels, train_size=0.85)

         RF.fit(features_train, labels_train)
Out[29]: RandomForestClassifier()

In [30]: labels_pred = RF.predict(features_test)

         print('Accuracy', ':', round(accuracy_score(labels_test, labels_pred), 5))
         Accuracy : 0.86191
```

Fig. 14. Prediction using Random Forest Classifier

The confusion matrix is shown below:

```
In [32]: from sklearn.metrics import confusion_matrix
         confusion_matrix(labels_test, labels_pred)

Out[32]: array([[136,   0,   5,   2,   0,   4,   1,   0,   2,   2],
                [  0, 160,   1,   0,   0,   3,   0,   0,   1,   0],
                [  5,   0, 144,   3,   0,   5,   0,   2,   6,   2],
                [  0,   2,  10, 127,   5,   1,   3,   3,   6,   0],
                [  2,   0,   1,   2, 112,   1,   2,   5,   6,   2],
                [  2,   7,   4,   1,   0, 123,   0,   1,   0,   2],
                [  3,   0,   0,   1,   0,   0, 130,   0,   0,   8],
                [  0,   1,   5,   3,   5,   0,   0, 117,   4,   2],
                [  2,   0,   6,   3,   3,   2,   1,   3, 139,   0],
                [  3,   1,  10,   7,   0,   6,   8,   1,   7, 104]], dtype=int64)
```

As we can see, most of the labels are correctly predicted by the model!

Fig. 15. Confusion matrix

Let's see whether our model gives the expected output for a particular input:

```
In [40]: input_features = np.array(features_test.iloc[46])
         print('actual label=' + labels_test.iloc[46])

         actual label=classical

In [41]: input_features = [input_features]
         predicted_output = RF.predict(input_features)
         print("X=%s, Predicted=%s" % (input_features[0], predicted_output[0]))

         X=[ 3.49688768e-01  8.80972371e-02  2.08271537e-02  1.45204349e-05
           1.64856213e+03  2.76393083e+04  1.72334188e+03  8.35330736e+03
           3.24439115e+03  6.09925886e+04  9.14137620e-02  5.59693060e-04
          -7.25451682e-04  4.01431724e-04 -6.21000072e-04  6.92661160e-06
           7.83025568e+01 -3.14067200e+02  7.51990356e+01  1.31005371e+02
           6.65715027e+01 -4.20387650e+01  4.28114319e+01  5.26039963e+01
           2.34803181e+01 -6.91924286e+00  4.43618088e+01  2.05195465e+01
           3.36384735e+01 -1.69745388e+01  2.10100098e+01  1.97107601e+01
           4.67669945e+01 -8.64368629e+00  2.87010288e+01  2.28990231e+01
           6.64836502e+01 -3.61770582e+00  2.83178558e+01  1.46355886e+01
           1.26431732e+02 -2.49454093e+00  1.31258469e+02  9.22462845e+00
           7.72005997e+01  8.44758224e+00  3.58542252e+01  1.27634726e+01
           4.72729187e+01  3.13751221e+00  2.55577450e+01  6.78646469e+00
           4.42495804e+01  4.06098604e+00  8.84661179e+01 -6.54077291e-01
           7.35635986e+01], Predicted=classical
```

Fig. 16. Predicted output

We can see that the predicted output is correct!

REFERENCES

[1] librosa.org
[2] how to split data into three sets train validation and test
[3] top machine learning algorithms for classification
[4] music genre classification
[5] wikipedia.org
[6] scikit-learn.org