# MACHINE LEARNING

## PROJECT

## MUSIC GENRE  CLASSIFICATION

DEVI 106116022

SHOBANA C 106116086

GAYATRI 106116024

# MUSIC GENRE CLASSIFICATION

## OBJECTIVE

Classifying music according to the genre using machine learning technique has proved to be quite successful in extracting trends and patterns from large pool of data. This principle is applied in Music Analysis also.

## DATASET

We have used GITZAN dataset . The dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres namely, blues, classical, country, disco, hip-hop, jazz, reggae, rock, metal and pop. Each genre consists of 100 sound clips.

## PREPROCESSING THE DATA

Before training the classification model the audio clips need to be converted from .au format to .wav format to make it compatible with python's wave module for reading audio files. We used the open source Sox module for the conversion.

## AUDIO LIBRARIES

Main library used for audio acquisition

- **Librosa**

It is a Python module to analyze audio signals in general but geared more towards music. It includes the nuts and bolts to build a MIR(Music information retrieval) system. It has been very well documented along with a lot of examples and tutorials.

## FEATURE EXTRACTION

Every audio signal consists of many features. This step is to extract meaningful features from the audio files. To classify our audio clips, we will choose 5 features, i.e. Mel-Frequency Cepstral Coefficients, Spectral Centroid, Zero Crossing Rate, Chroma Frequencies, Spectral Roll-off. All the features are then appended into a .csv file so that classification algorithms can be used.

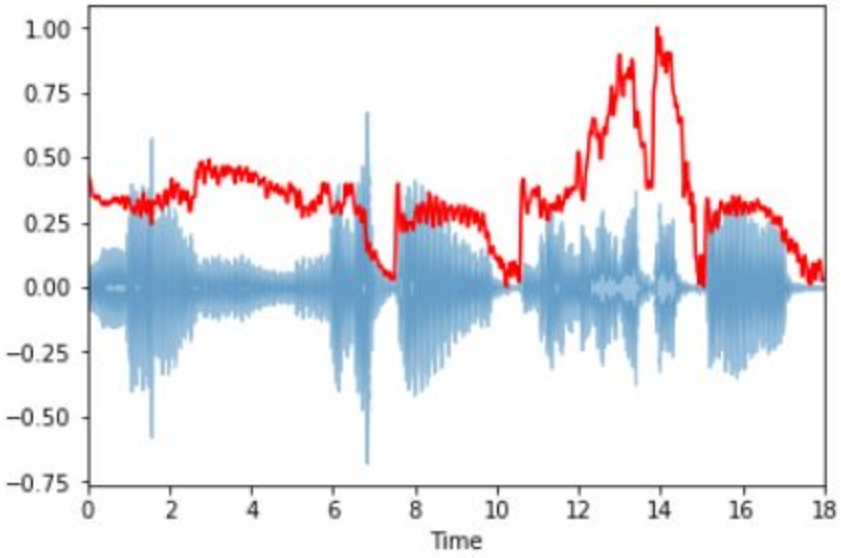
- **Zero Crossing Rate**

  The zero crossing rate is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to negative or back. This feature has been used heavily in both speech recognition and music information retrieval. It usually has higher values for highly percussive sounds like those in metal and rock.

- **Spectral Centroid**

  It indicates where the "centre of mass" for a sound is located and is calculated as the weighted mean of the frequencies present in the sound. Consider two songs, one from a blues genre and the other belonging to metal. Now as compared to the blues genre song which is the same throughout its length, the metal song has more frequencies towards the

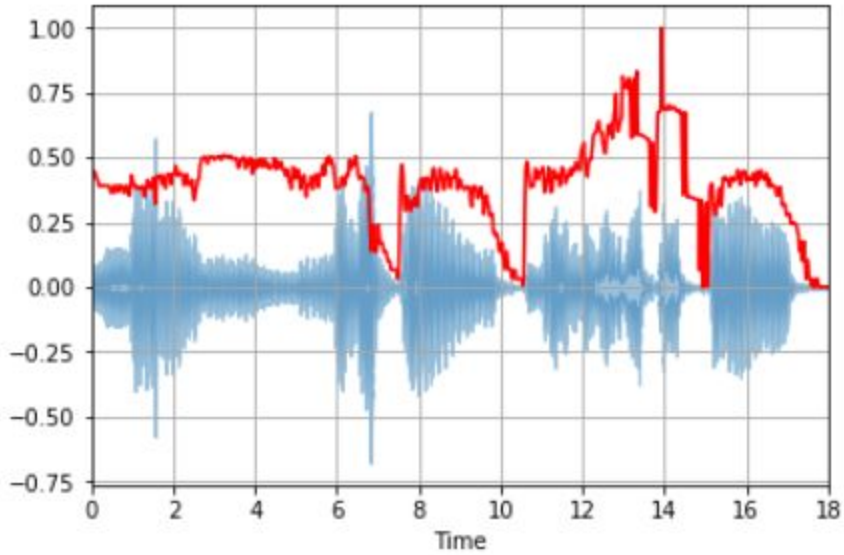
end. So spectral centroid for blues song will lie somewhere near the middle of its spectrum while that for a metal song would be towards its end.




There is a rise in the spectral centroid towards the end.

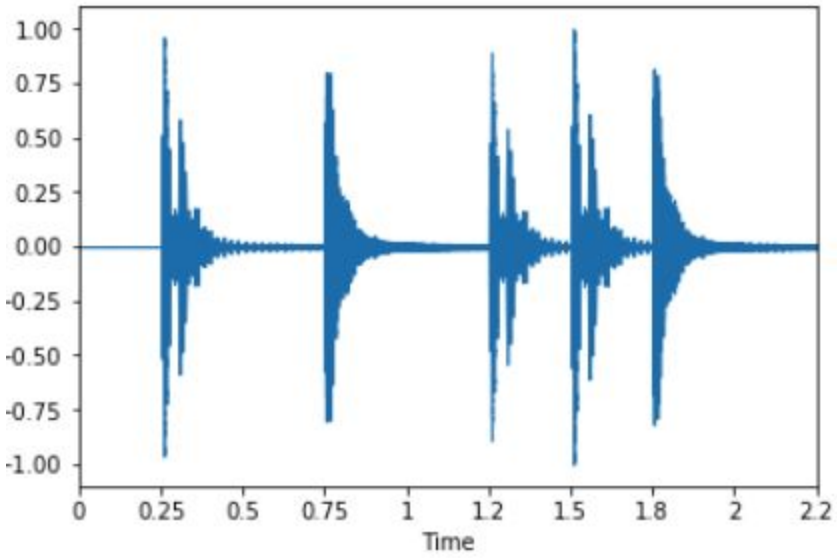
- **Spectral Rolloff**

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%, lies.



- **Mel-Frequency Cepstral Coefficients**

The Mel frequency cepstral coefficients (MFCCs) of a signal are a small set of features (usually about 10–20) which concisely describe the overall shape of a spectral envelope. It models the characteristics of the human voice.




Here mfcc computed 20 MFCC s over 97 frames.

- **Chroma Frequencies**

Chroma features are an interesting and powerful representation for music audio in which the entire spectrum is projected onto 12 bins representing the 12 distinct semitones (or chroma) of the musical octave.

```
In [9]: data_set=pandas.read_csv('data_set.csv',index_col=False)
        GENRES=['METAL', 'CLASSICAL', 'HIPHOP', 'BLUES', 'POP', 'REGGAE','COUNTRY','DISCO','JAZZ','ROCK']

        number_of_rows,number_of_cols = data_set.shape
        data_set[:10].style
```

Out[9]:

| | meanZCR | stdZCR | meanSpecCentroid | stdSpecCentroid | meanSpecContrast | stdSpecContrast | meanSpecBandwidth | stdSpecBandwidth | meanSpecRollof | s |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.505871 | -0.784343 | -0.381408 | -0.691147 | -0.107291 | 0.349902 | -0.18728 | -0.572265 | -0.236668 | |
| 1 | -0.789556 | -0.726759 | -0.559681 | -0.463433 | -0.214652 | 0.274741 | -0.139501 | -0.41469 | -0.333817 | |
| 2 | -0.590099 | -0.556617 | -0.500726 | -0.520669 | 0.110841 | 0.342202 | -0.352145 | -0.704373 | -0.413241 | |
| 3 | -0.943024 | -0.92013 | -0.872967 | -0.743929 | -0.0695439 | 0.0434305 | -0.635685 | -0.596014 | -0.772659 | |
| 4 | -0.20544 | -0.400993 | -0.236112 | -0.243814 | 0.190382 | 0.270769 | -0.310327 | -0.532099 | -0.198146 | |
| 5 | -0.731302 | -0.90132 | -0.711479 | -0.733247 | 0.0493073 | -0.157888 | -0.578599 | -0.542007 | -0.625424 | |
| 6 | -0.617483 | -0.667086 | -0.497414 | -0.341626 | 0.410133 | 0.0539862 | -0.523793 | -0.392305 | -0.39341 | |
| 7 | -0.716647 | -0.886928 | -0.552424 | -0.731196 | 0.0718099 | 0.063368 | -0.357423 | -0.687419 | -0.421731 | |
| 8 | -0.610462 | -0.688393 | -0.386585 | -0.516244 | 0.191406 | 0.557346 | -0.156668 | -0.516733 | -0.219502 | |
| 9 | -0.516476 | -0.676589 | -0.302418 | -0.461719 | -0.0168409 | 0.273545 | -0.13089 | -0.53325 | -0.121163 | |

## CLASSIFICATION

After the feature is extracted, we can use existing classification algorithms to classify the songs into different genres. We have used the following algorithms for classification:

- ***K-nn***

```
In [39]: results_knn=[]
         for i in range(1,11):
             knn=KNeighborsClassifier(n_neighbors=i)
             knn.fit(train_x,train_y)
             results_knn.append(knn.score(test_x,test_y))

         max_accuracy_knn=max(results_knn)
         best_k=1+results_knn.index(max(results_knn))
         print("Max Accuracy is {:.3f} on test dataset with {} neighbors.\n".format(max_accuracy_knn,best_k))

         plt.plot(numpy.arange(1,11),results_knn)
         plt.xlabel("n Neighbors")
         plt.ylabel("Accuracy")


         knn=KNeighborsClassifier(n_neighbors=best_k)
         knn.fit(train_x,train_y)
         print("Training Score: {:.3f}".format(knn.score(train_x,train_y)))
         print("Test score: {:.3f}".format(knn.score(test_x,test_y)))

         plot_cnf(knn,test_x,test_y,GENRES)
```

```
Max Accuracy is 0.573 on test dataset with 7 neighbors.

Training Score: 0.694
Test score: 0.573
---------------PERFORMANCE ANALYSIS FOR THE MODEL----------------

Real Test dataset labels:
['hiphop' 'rock' 'pop' 'blues' 'metal' 'disco' 'metal' 'rock' 'blues'
 'rock' 'rock' 'classical' 'reggae' 'blues' 'classical' 'blues' 'disco'
 'blues' 'reggae' 'pop' 'hiphop' 'country' 'pop' 'country' 'classical'
 'country' 'disco' 'disco' 'country' 'classical' 'pop' 'hiphop' 'jazz'
 'rock' 'pop' 'metal' 'rock' 'rock' 'jazz' 'reggae' 'country' 'pop'
 'hiphop' 'rock' 'blues' 'blues' 'jazz' 'reggae' 'metal' 'disco' 'hiphop'
 'reggae' 'blues' 'rock' 'disco' 'rock' 'metal' 'disco' 'rock' 'reggae'
 'country' 'country' 'hiphop' 'disco' 'hiphop' 'jazz' 'classical' 'reggae'
 'country' 'reggae' 'jazz' 'hiphop' 'country' 'disco' 'metal' 'metal'
```

- ***Random Forests***

```
In [36]:
         results_forest=[]
         for i in range(2,20):
             forest=RandomForestClassifier(random_state=42,n_estimators=i)
             forest.fit(train_x,train_y)
             results_forest.append(forest.score(test_x,test_y))

         max_accuracy_forest=max(results_forest)
         best_n_est=2+results_forest.index(max(results_forest))
         print("Max Accuracy is {:.3f} on test dataset with {} estimators.\n".format(max_accuracy_forest,best_n_est))

         plt.plot(numpy.arange(2,20),results_forest)
         plt.xlabel("n Estimators")
         plt.ylabel("Accuracy")


         forest=RandomForestClassifier(random_state=42,n_estimators=best_n_est)
         forest.fit(train_x,train_y)
         print("Training Score: {:.3f}".format(forest.score(train_x,train_y)))
         print("Test score: {:.3f}".format(forest.score(test_x,test_y)))

         plot_cnf(forest,test_x,test_y,GENRES)
```

```
Max Accuracy is 0.445 on test dataset with 19 estimators.

Training Score: 1.000
Test score: 0.445
---------------PERFORMANCE ANALYSIS FOR THE MODEL----------------

Real Test dataset labels:
['pop' 'disco' 'metal' 'jazz' 'metal' 'classical' 'hiphop' 'classical'
 'hiphop' 'hiphop' 'reggae' 'hiphop' 'pop' 'reggae' 'hiphop' 'reggae'
 'country' 'disco' 'blues' 'pop' 'classical' 'country' 'jazz' 'rock'
 'jazz' 'metal' 'jazz' 'classical' 'jazz' 'metal' 'metal' 'disco' 'disco'
 'pop' 'hiphop' 'classical' 'country' 'rock' 'country' 'jazz' 'hiphop'
 'metal' 'pop' 'rock' 'metal' 'rock' 'pop' 'reggae' 'metal' 'metal' 'pop'
 'classical' 'disco' 'classical' 'pop' 'country' 'hiphop' 'rock' 'reggae'
```

- **Ne*ural Network***

```
In [26]: neural=MLPClassifier(max_iter=400,random_state=2,hidden_layer_sizes=[40,40])
         neural.fit(train_x,train_y)
         print("Training Score: {:.3f}".format(neural.score(train_x,train_y)))
         print("Test score: {:.3f}".format(neural.score(test_x,test_y)))

         plot_cnf(neural,test_x,test_y,GENRES)
```

```
C:\Users\cshob\Anaconda3\lib\site-packages\sklearn\neural_network\multilayer_perceptron.py:562: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (400) reached and the optimization hasn't converged yet.
  % self.max_iter, ConvergenceWarning)
```

```
Training Score: 0.916
Test score: 0.643
---------------PERFORMANCE ANALYSIS FOR THE MODEL----------------

Real Test dataset labels:
['hiphop' 'rock' 'pop' 'blues' 'metal' 'disco' 'metal' 'rock' 'blues'
 'rock' 'rock' 'classical' 'reggae' 'blues' 'classical' 'blues' 'disco'
 'blues' 'reggae' 'pop' 'hiphop' 'country' 'pop' 'country' 'classical'
 'country' 'disco' 'disco' 'country' 'classical' 'pop' 'hiphop' 'jazz'
 'rock' 'pop' 'metal' 'rock' 'rock' 'jazz' 'reggae' 'country' 'pop'
 'hiphop' 'rock' 'blues' 'blues' 'jazz' 'reggae' 'metal' 'disco' 'hiphop'
 'reggae' 'blues' 'rock' 'disco' 'rock' 'metal' 'disco' 'rock' 'reggae'
 'country' 'country' 'hiphop' 'disco' 'hiphop' 'jazz' 'classical' 'reggae'
 'country' 'reggae' 'jazz' 'hiphop' 'country' 'disco' 'metal' 'metal'
 'pop' 'jazz' 'rock' 'disco' 'pop' 'metal' 'pop' 'hiphop' 'disco' 'disco'
 'rock' 'metal' 'country' 'disco' 'pop' 'metal' 'classical' 'rock'
 'reggae' 'blues' 'disco' 'hiphop' 'metal' 'disco' 'blues' 'hiphop'
 'metal' 'disco' 'hiphop' 'disco' 'country' 'blues' 'blues' 'country'
 'jazz' 'hiphop' 'country' 'blues' 'country' 'rock' 'country' 'blues'
 'classical' 'country' 'country' 'pop' 'rock' 'jazz' 'reggae' 'country'
 'pop' 'blues' 'blues' 'pop' 'country' 'hiphop' 'disco' 'hiphop' 'hiphop'
 'metal' 'jazz' 'pop' 'pop' 'reggae' 'rock' 'reggae' 'reggae' 'blues'
 'disco' 'metal' 'country' 'jazz' 'reggae' 'country' 'country' 'classical'
 'rock' 'rock' 'metal' 'reggae' 'disco' 'reggae' 'pop' 'reggae' 'rock'
 'pop' 'pop' 'country' 'hiphop' 'jazz' 'pop' 'jazz' 'metal' 'classical'
 'metal' 'country' 'blues' 'reggae' 'jazz' 'country' 'classical' 'disco'
```

- **SVM**

```
In [32]: svm=SVC(C=100,gamma=0.08)
         svm.fit(train_x,train_y)
         print("Training Score: {:.3f}".format(svm.score(train_x,train_y)))
         print("Test score: {:.3f}".format(svm.score(test_x,test_y)))

         plot_cnf(svm,test_x,test_y,GENRES)
```

```
Training Score: 0.991
Test score: 0.665
---------------PERFORMANCE ANALYSIS FOR THE MODEL----------------

Real Test dataset labels:
['hiphop' 'metal' 'classical' 'classical' 'reggae' 'blues' 'blues' 'blues'
 'classical' 'disco' 'classical' 'reggae' 'jazz' 'disco' 'country'
 'classical' 'rock' 'classical' 'hiphop' 'hiphop' 'disco' 'classical'
 'disco' 'disco' 'rock' 'classical' 'jazz' 'disco' 'jazz' 'rock' 'country'
 'blues' 'pop' 'jazz' 'country' 'rock' 'metal' 'blues' 'pop' 'metal'
 'metal' 'rock' 'reggae' 'blues' 'rock' 'jazz' 'hiphop' 'rock' 'hiphop'
 'jazz' 'disco' 'metal' 'country' 'disco' 'hiphop' 'metal' 'rock'
 'classical' 'rock' 'pop' 'jazz' 'reggae' 'reggae' 'pop' 'jazz' 'metal'
 'jazz' 'rock' 'rock' 'country' 'metal' 'jazz' 'disco' 'pop' 'hiphop'
 'classical' 'jazz' 'reggae' 'hiphop' 'country' 'hiphop' 'jazz' 'disco'
 'reggae' 'country' 'metal' 'pop' 'metal' 'jazz' 'pop' 'metal' 'blues'
 'hiphop' 'reggae' 'reggae' 'metal' 'pop' 'hiphop' 'blues' 'blues'
 'hiphop' 'reggae' 'jazz' 'disco' 'classical' 'hiphop' 'rock' 'disco'
 'classical' 'country' 'disco' 'disco' 'metal' 'pop' 'reggae' 'country'
 'hiphop' 'country' 'blues' 'blues' 'classical' 'country' 'hiphop' 'pop'
 'blues' 'blues' 'jazz' 'metal' 'metal' 'reggae' 'reggae' 'pop' 'rock'
 'classical' 'blues' 'metal' 'rock' 'blues' 'jazz' 'reggae' 'disco' 'pop'
 'blues' 'disco' 'disco' 'classical' 'country' 'metal' 'pop' 'blues'
 'jazz' 'blues' 'jazz' 'reggae' 'classical' 'disco' 'pop' 'reggae' 'pop'
 'rock' 'country' 'rock' 'hiphop' 'blues' 'metal' 'reggae' 'hiphop'
 'country' 'jazz' 'blues' 'classical' 'country' 'disco' 'hiphop' 'reggae'
 'disco' 'country' 'rock' 'country' 'reggae' 'rock' 'pop' 'classical'
 'classical' 'classical' 'pop' 'hiphop' 'rock' 'pop' 'rock' 'jazz' 'pop'
```