Project Report On

Music Genre Classification Using Machine Learning & IBM Cloud

INTRODUCTION

1.1 Overview

Music has become the most favorable area nowadays especially in youth. Most of the people tend to listen music of certain genre such as classical, hip-hop or disco and want a user-friendly way to classify the music as per their preferences. Due to this “Music Genre Classification” came into picture. Music genre classification is a complex task in music information retrieval (MIR) due to selection and extraction of suitable features.

The Machine learning models have been shown to be capable of solving these kinds of real-life problems. Music genre classification can be implemented using various machine learning algorithms. In the proposed system, we are using a deep learning technique. I.e., K-Nearest Neighbours (KNN) for classifying the music in various genres. CNNs are used to solve image pattern recognition tasks. While analyzing music, acoustic feature extraction is the most crucial task. In proposed system, model is trained over GTZAN dataset.

1.2 Purpose

A music genre classifier is a software program that predicts the genre of a piece of music in audio format. These devices are used for tasks such as automatically tagging music for distributors such as Spotify and Billboard and determining appropriate background music for events.

Music Genre Classification or classification of music into different categories or genres is a concept that helps the masses to differentiate between 2 genres based on their composition or the beats they withhold. In recent times, music genre classification has become a very popular concept as more and more genres are emerging around the world.

From K-Pop to Jazz, music lovers rely on the technology of music genre classification and are able to listen to songs as per their preferences. While it takes only a click for a listener to switch from Jazz music to Rap, there is certainly much more beneath the surface that fuels our love for music.

2.LITERATURE SURVEY

2.1 Existing Problem

- ❑ Deep learning models:

    1.Convolutional Neural Networks (CNN).

    2.Recurrent Neural Networks (RNN).

    3.Boltzmann machine.

4.Autoencoders etc.

- ❑ Classification:

1.The K-Nearest Neighbours algorithm

2.Decision Tree

3.Support Vector Machines

4.Naive Bayes

- ❑ Regression:

1.Linear Regression

2.Lasso Regression

3.Ridge Regression

4.Support Vector Regression (SVR)

5.Ensemble Regression

- ❑ Clustering:

1.K means

2.K means++

3.K medoids

4.Agglomerative clustering

5.DBSCAN

Above are some of the existing approaches or methods to solve this problem of Music Genre Classification

The preferable method for solving Music Genre Classification problem is KNN-Algorithm which gives the best accuracy among all available methods or approaches.

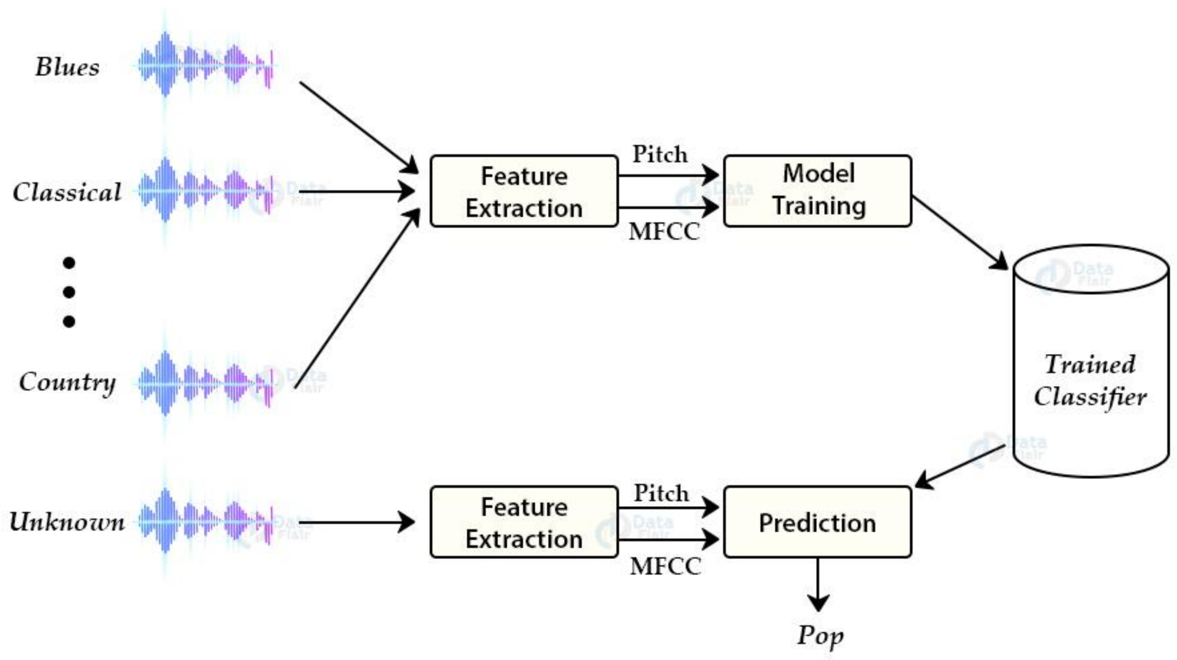
2.2 Proposed Solution

We have chosen **K-Nearest Neighbors** machine algorithm for classification of music genre and KNN is a machine learning algorithm used for regression, and classification. It is also known as the lazy learner algorithm. It simply uses a distance-based method find the K number of similar Neighbors to new data and the class in which the majority of Neighbours lies, it results in that class as an output.

From results we found that k NN classifier gave more exact results compared to support vector machine classifier. If the training data is bigger than number of features, k NN gives better outcomes than SVM. SVM can only identify limited set of patterns. KNN classifier is more powerful for the classification of music genre.

3. THEORETICAL ANALYSIS

3.1 Block Diagram:



3.2 Hardware/Software Designing

1.Software Requirements

1.Downloading of Anaconda Navigator

2.Downloading of python packages like

a. NumPy Package

b. Pandas

c. librosa

d. Tensor Flow

e. Matplotlib

f. scikit-learn

g. Flask

h. pyhton_speech_features

I. mfcc

j. from python_speech_features import mfcc

k. import sklearn.model_selection

l. from sklearn.model_selection import train_test_split

m. import scipy.io.wavfile as wav

n. import os

o. import pickle

p. import operator

These are some of the software requirements required to implement the music genre classification project using KNN algorithm.

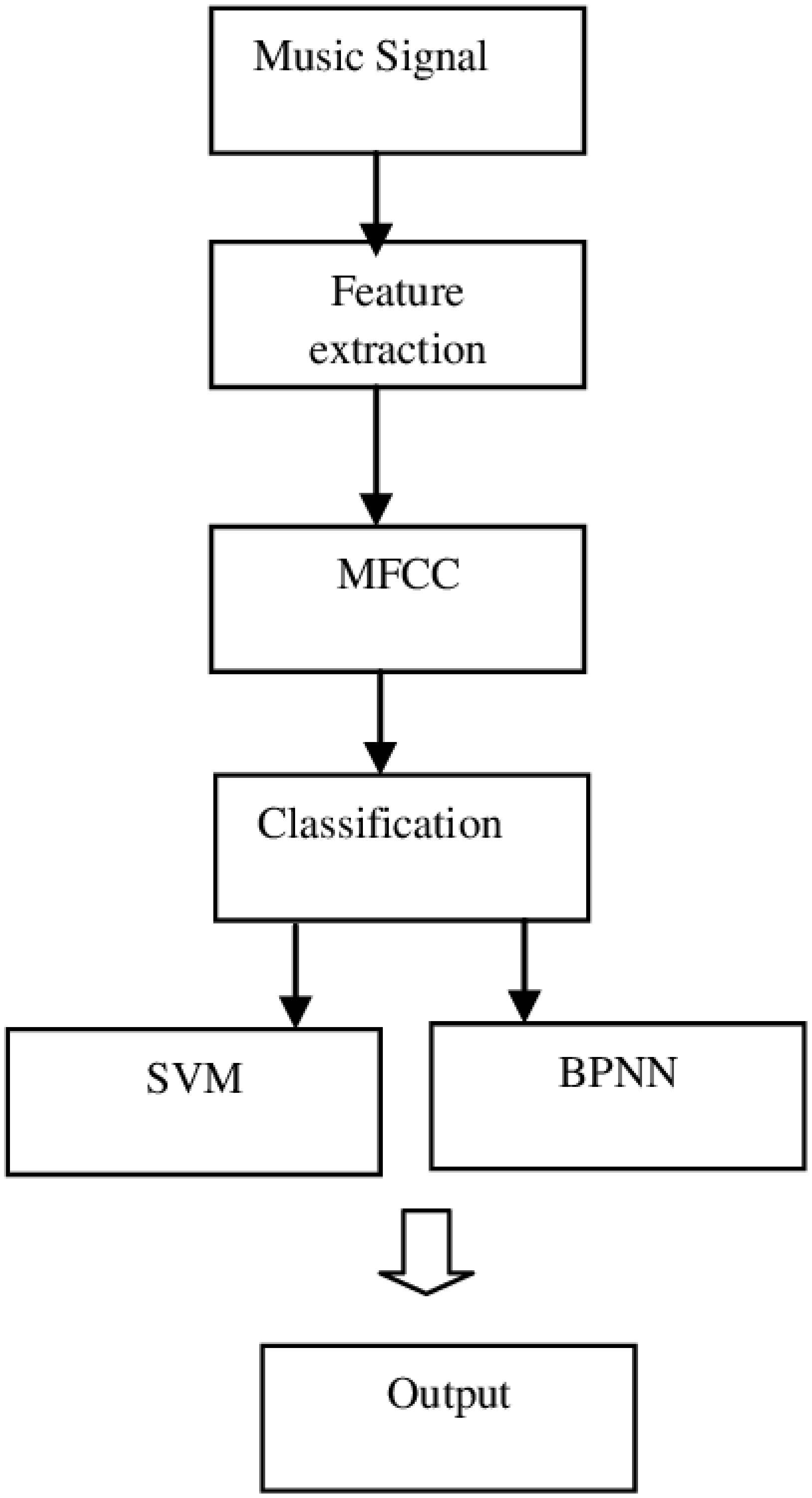
4.EXPERIMENTAL INVESTIGATION

The GTZAN genre collection dataset was collected. It consists of 1000 audio files each having 30 seconds duration. There are 10 classes (10 music genres) each containing 100 audio tracks. Each track is in .wav format. It contains audio files of the following 10 genres:

- Blues
- Classical
- Country
- Disco
- Hip-hop
- Jazz
- Metal
- Pop
- Reggae
- Rock

5.FLOWCHART

- ❖ **Flow chart of the music genre classification:**


Music Signal
Feature extraction
MFCC
Classification
SVM
BPNN
Output

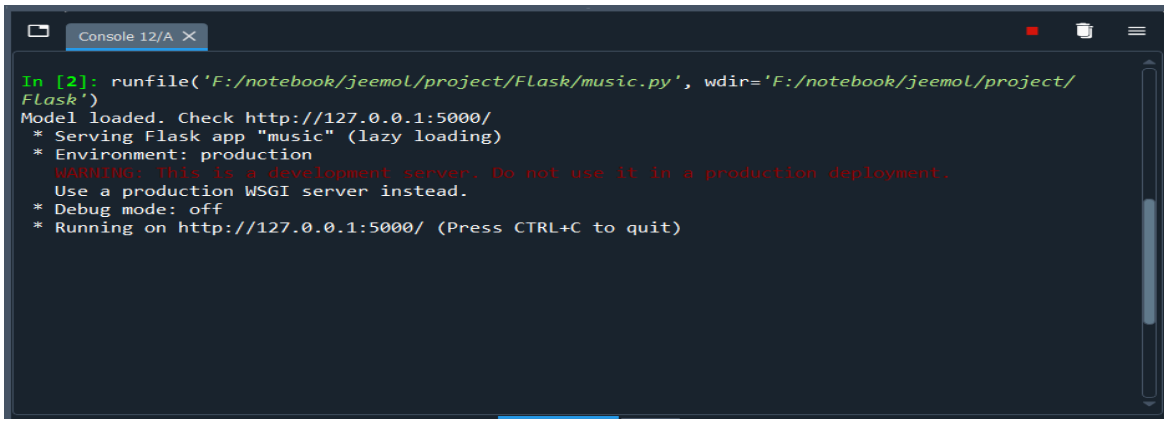
6.RESULT

Since GTZAN data set consists of ten different genres, accuracy was used as the main performance metric. Although it can be assumed as a subjective measure in view of a listener, the average percentage of music similarity is also used as a metric for quality of music recommendation.

The best performance in terms of accuracy is observed for the KNN model that uses as an input to predict the music genre with a test accuracy of 88.54**.**

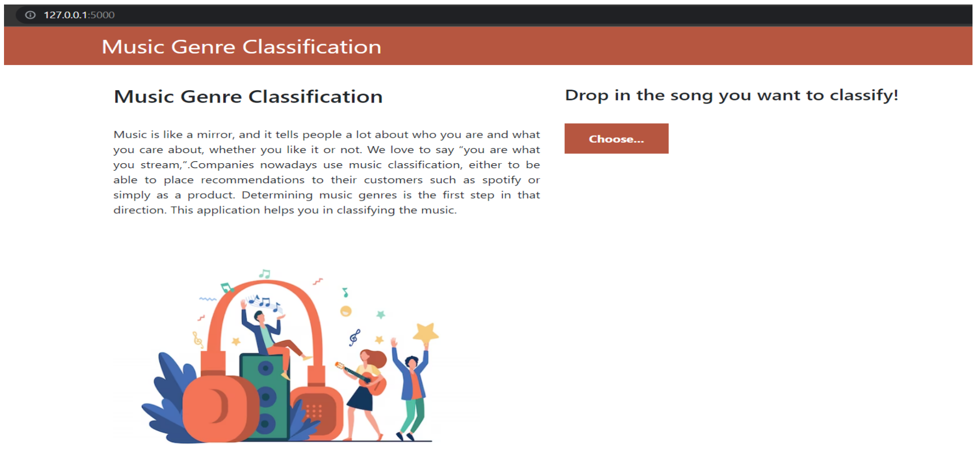
Although performance varied with each classification algorithm, similar trends were identified in the results of each. For example, across most experiments, Hip-Hop were classified the most accurately.

Fig:1 output of the GTZAN dataset:



```
In [2]: runfile('F:/notebook/jeemol/project/Flask/music.py', wdir='F:/notebook/jeemol/project/
Flask')
Model loaded. Check http://127.0.0.1:5000/
 * Serving Flask app "music" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fig 2: Web Application view :



127.0.0.1:5000

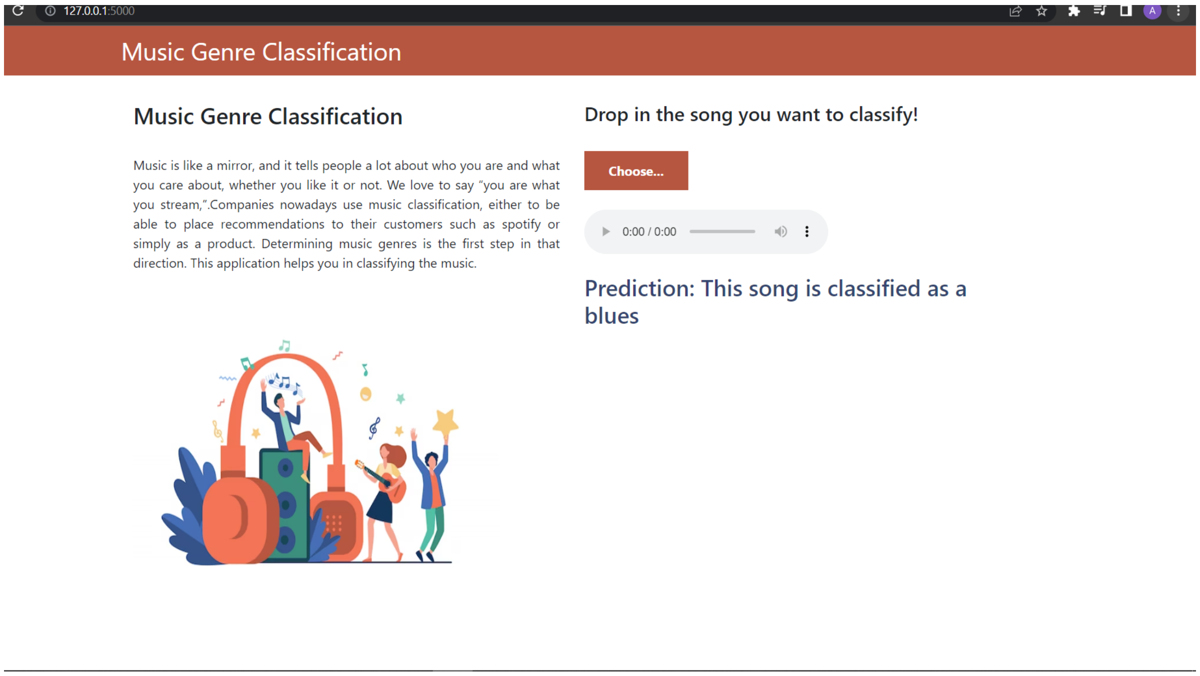
Music Genre Classification

Music Genre Classification

Music is like a mirror, and it tells people a lot about who you are and what you care about, whether you like it or not. We love to say "you are what you stream,".Companies nowadays use music classification, either to be able to place recommendations to their customers such as spotify or simply as a product. Determining music genres is the first step in that direction. This application helps you in classifying the music.

Drop in the song you want to classify!

Fig 3: Predicting the music genre type:



Music Genre Classification

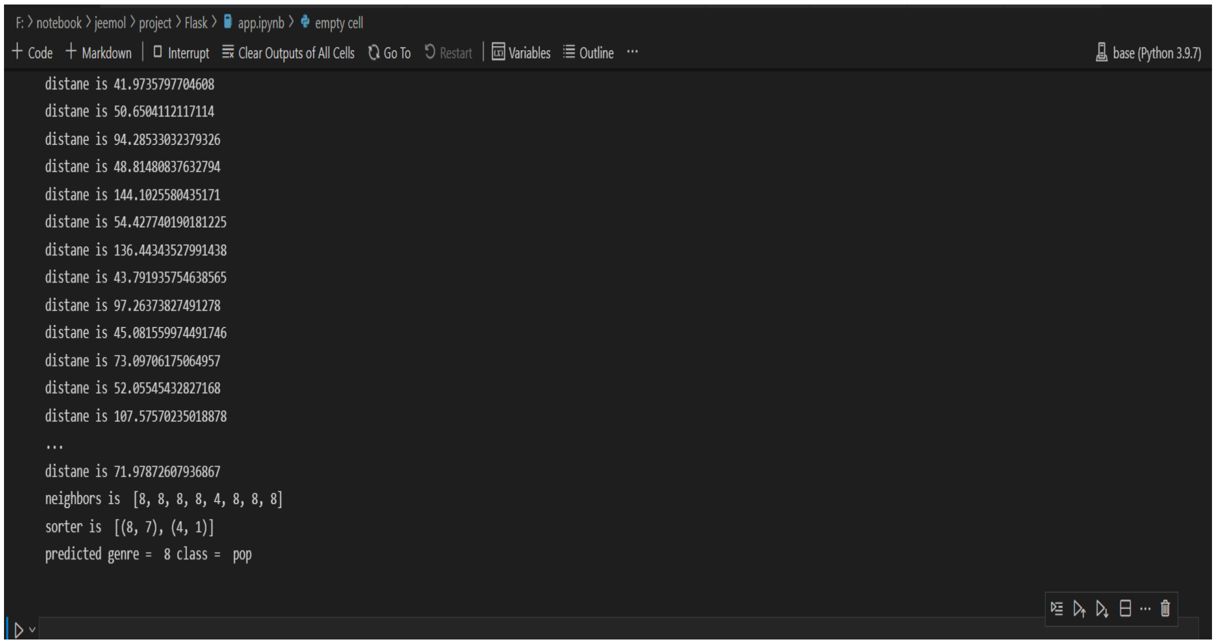
Music Genre Classification

Music is like a mirror, and it tells people a lot about who you are and what you care about, whether you like it or not. We love to say "you are what you stream,".Companies nowadays use music classification, either to be able to place recommendations to their customers such as spotify or simply as a product. Determining music genres is the first step in that direction. This application helps you in classifying the music.

Drop in the song you want to classify!

Prediction: This song is classified as a blues

Fig 4: Distances of music beats in pop genre :


F: › notebook › jeemol › project › Flask › app.ipynb › empty cell
+ Code + Markdown Interrupt Clear Outputs of All Cells Go To Restart Variables Outline ...
base (Python 3.9.7)

```
distane is 41.9735797704608
distane is 50.6504112117114
distane is 94.28533032379326
distane is 48.81480837632794
distane is 144.1025580435171
distane is 54.427740190181225
distane is 136.44343527991438
distane is 43.791935754638565
distane is 97.26373827491278
distane is 45.081559974491746
distane is 73.09706175064957
distane is 52.05545432827168
distane is 107.57570235018878
...
distane is 71.97872607936867
neighbors is  [8, 8, 8, 8, 4, 8, 8, 8]
sorter is  [(8, 7), (4, 1)]
predicted genre =  8 class =  pop
```

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Music genres are important because they are a fundamental means of understanding and discussing music as an art form.
- Classification results can be used to support sociological and psychological research into how humans construct the notion of musical similarity and form musical groupings, and how this compares to the objective truth produced by computerized classifiers.
- Effective if the training data is large

DISADVANTAGES

- This method is not able to be implemented in real time since we need to process the information of whole piece of music.
- Need to determine the value of parameter K.
- Distance based on learning is not clear which type of distance to use and which attribute to use to produce the best results.

❑

8.APPLICATIONS

Companies nowadays use music classification, either to be able to place recommendations to their customers (such as Spotify, Soundcloud) or simply as a product (for example, Shazam). Determining music genres is the first step in that direction. Machine Learning techniques have proved to be quite successful in extracting trends and patterns from a large data pool. The same principles are applied in Music Analysis also.

9.CONCLUSION

This project studies several methods of music genre classification. We study several audio feature extraction methods using digital signal processing methods, including MFCC (mel-frequency cepstral coefficients), etc. And we propose two main architectures for music genre classification. One is the combination with different level features with decision tree or random forest model, and the other one is the combination with a series of MFCC coefficients with KNN and K-Means clustering. We quantitatively compare the performance of classifying jazz, classical, metal and pop music using two architectures, and find that the random forest model gives a better result with 86.75% accuracy.

10. FUTURE SCOPE

➢ In the future, we hope to experiment with other types of deep learning methods, given they performed the best. Given that this is time series data, some sort of KNN model may work . We are also curious about generative aspects of this project, including some sort of genre conversion (in the same vein as generative adversarial networks which repaint photos in the style of Van Gogh, but for specifically for music). Additionally, we suspect that we may have opportunities for transfer learning, for example in classifying music by artist or by decade.

- In our project, although we do the dataset augmentation by split the 30 seconds music into mutiple clips with various split segmentation intervals. But we still see some over-fitting and inconsistency in the accuracy results. More test data will absolutely help us move further and understand more deeper of the algorithms we learnt in this course

11.BIBLIOGRAPHY

1. https://nevonprojects.com/music-genres-classification-using-knn-system/

2.https://www.kaggle.com/code/rxsraghavagrawal/music-genre-classification-using-knn-begineers/notebook

3. https://github.com/HetGalia/Music-Genre-Classification-using-KNN

APPENDIX:

Source code:

```
#!/usr/bin/env python

# coding: utf-8



# In[1]:





from __future__ import division, print_function

import os

import numpy as np

from flask import Flask, request, render_template
from werkzeug.utils import secure_filename

from python_speech_features import mfcc

import scipy.io.wavfile as wav

import pickle

import operator



# In[2]:




app = Flask(__name__)


dataset = []

def loadDataset(filename):

    with open("my1.dat" , 'rb') as f:

        while True:

            try:

                dataset.append(pickle.load(f))

            except EOFError:

                f.close()

                break


loadDataset("my1.dat")
```

```
# In[3]:



def distance(instance1 , instance2 , k ):
    distance =0
    mm1 = instance1[0]
    cm1 = instance1[1]
    mm2 = instance2[0]
    cm2 = instance2[1]

    #Method to calculate distance between two instances.
    distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))
    distance+=(np.dot(np.dot((mm2-mm1).transpose() , np.linalg.inv(cm2)) , mm2-mm1 ))
    distance+= np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))
    distance-= k
    print("distane is",distance)
    return distance



# In[4]:



def getNeighbors(trainingSet , instance , k):
    distances =[]

    for x in range (len(trainingSet)):

        dist = distance(trainingSet[x], instance, k )+ distance(instance, trainingSet[x], k)

        distances.append((trainingSet[x][2], dist))

    distances.sort(key=operator.itemgetter(1))

    neighbors = []

    for x in range(k):

        neighbors.append(distances[x][0])

    print("neighbors is ",neighbors)

    return neighbors




# In[5]:




def nearestClass(neighbors):

    classVote ={}

    for x in range(len(neighbors)):

        response = neighbors[x]

        if response in classVote:

            classVote[response]+=1

        else:

            classVote[response]=1

    sorter = sorted(classVote.items(), key = operator.itemgetter(1), reverse=True)
    print("sorter is ",sorter)

    return sorter[0][0]




# In[6]:




print('Model loaded. Check http://127.0.0.1:5000/')




# In[7]:






@app.route('/', methods=['GET'])

def index():

    # Main page

    return render_template('music.html')




# In[10]:




@app.route('/predict', methods=['GET', 'POST'])
```

```python
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['image']

        # Save the file to ./uploads
        basepath = "F:/notebook/jeemol/project/Flask"
        file_path = os.path.join(basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        print(file_path)
        i=1

        results = {1: 'blues', 2: 'classical', 3: 'country', 4: 'disco', 5: 'hiphop',
                   6: 'jazz', 7: 'metal', 8: 'pop', 9: 'reggae', 10: 'rock'}

        (rate,sig)=wav.read(file_path)
        print(rate,sig)
        mfcc_feat=mfcc(sig,rate,winlen=0.020,appendEnergy=False)
        covariance = np.cov(np.matrix.transpose(mfcc_feat))
        mean_matrix = mfcc_feat.mean(0)
        feature=(mean_matrix,covariance,0)
        pred=nearestClass(getNeighbors(dataset ,feature , 8))

        print("predicted genre = ",pred,"class = ",results[pred])
        return "This song is classified as a "+str(results[pred])



# In[ ]:




if __name__ == '__main__':

    app.run(threaded = False)



# In[ ]:
```

```
#Importing necessary Libraries

import numpy as np

import scipy.io.wavfile as wav

from python_speech_features import mfcc



import os
import pickle

import operator

import librosa # main package for working with Audio Data

import librosa.display


#Defining the necessary functions for creating a dataset for KNN matching.


#Define a function to get the distance between feature vectors and find neighbors:

def distance(instance1 , instance2 , k ):

    distance =0

    mm1 = instance1[0]

    cm1 = instance1[1]

    mm2 = instance2[0]

    cm2 = instance2[1]


     #Method to calculate distance between two instances.

    distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))

    distance+=(np.dot(np.dot((mm2-mm1).transpose() , np.linalg.inv(cm2)) , mm2-mm1 ))

    distance+= np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))

    distance-= k

    return distance


#This function returns a list of K nearest neighbours for any instance

#to be checked within a given dataset (dataset of features.)
```

```
def getNeighbors(trainingSet , instance , k):
   distances =[]
   for x in range (len(trainingSet)):
      dist = distance(trainingSet[x], instance, k )+ distance(instance, trainingSet[x], k)
      distances.append((trainingSet[x][2], dist))
   distances.sort(key=operator.itemgetter(1))
   neighbors = []
   for x in range(k):
      neighbors.append(distances[x][0])
   return neighbors



#Identify the nearest neighbors:
def nearestClass(neighbors):
   classVote = {}

   for x in range(len(neighbors)):
      response = neighbors[x]
      if response in classVote:
         classVote[response]+=1
      else:
         classVote[response]=1

   sorter = sorted(classVote.items(), key = operator.itemgetter(1), reverse=True)
```

```
    return sorter[0][0]


#Extract features from the data (audio files) and dump these features

#into a binary .dat file “my1.dat”:

directory = "F:/notebook/jeemol/project/"

f = open("my1.dat" ,'wb')

i = 0



#Dataset creation :  making a dat file were we get all the data about the audio files in a
".dat" file.

for folder in os.listdir(directory):

    #as we have 10 classes, we're starting the loop from 1 to 11

    #so that we run the loop for total of 10 times, with each folder change (genre change),  i
(label) changes.

    i += 1

    if i==11 :

        break

    for file in os.listdir(directory+folder):


        #To read an Wav audio File in Python

        (rate,sig) = wav.read(directory+folder+"/"+file)

        #MFCC is the feature we will use for our analysis,

        #because it provides data about the overall shape of the audio frequencies.

        mfcc_feat = mfcc(sig,rate ,winlen=0.020, appendEnergy = False)
        covariance = np.cov(np.matrix.transpose(mfcc_feat))

        mean_matrix = mfcc_feat.mean(0)

        #making a feature typle that contains the mean matrix from mfcc as well as
covariance,

        #and last variable in the feature tuple is the label (where numbers correspond to
particular genre)

        feature = (mean_matrix , covariance , i)


        #This the the created dataset, which takes the input from specified path

        #it then stores the feature as a .dat file which can be used later without having to train
all the files over it.

        pickle.dump(feature , f)

f.close()


#Loading the created dataset into a python readable object (list)

dataset = []

def loadDataset(filename):

    with open("F:/notebook/jeemol/project/Flask/my1.dat" , 'rb') as f:

        while True:

            try:

                dataset.append(pickle.load(f))

            except EOFError:

                f.close()

                break
```

```
loadDataset("F:/notebook/jeemol/project/Flask/my1.dat")



#we have to convert the dataset from a list to np array.
dataset = np.array(dataset)
#type(dataset) ##uncomment this line to check the type of (dataset),

#Train and test split on the dataset:
#as the dataset contains features for all the audio files,
#we have to split that manually into train and test data
from sklearn.model_selection import train_test_split
x_train ,x_test = train_test_split(dataset,test_size=0.15)

#Make prediction using KNN and get the accuracy on test data:
leng = len(x_test)
predictions = []
for x in range (leng):
    predictions.append(nearestClass(getNeighbors(x_train ,x_test[x] , 8)))

#Define a function for model evaluation:
def getAccuracy(testSet, predictions):
    #this is a variable to count total number of correct predictions.
    correct = 0
    for x in range (len(testSet)):
        if testSet[x][-1]==predictions[x]:

            correct+=1

    return 1.0*correct/len(testSet)




#Print accuracy using defined function

accuracy1 = getAccuracy(x_test , predictions)

print(accuracy1)
```