Internship Project

Report on

Music Genre Classification Using IBM Machine Learning Service

Submitted by:

Swathi C S	4BD18CS111
Swathi P	4BD18CS112
Yashaswini S N	4BD18CS123
Sahana D V	4BD19CS409

Department Of Computer Science and Engineering

BAPUJI INSTITUTE OF ENGINEERING AND TECHNOLOGY

DAVANGERE

Academic year: 2021-2022

CONTENTS

MUSIC GENRE CLASSIFICATION USING IBM MACHINE LEARNING SERVICE

Project report

CHAPTER 1

INTRODUCTION

1.1 Overview

- 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, although in practice these terms are sometimes used interchangeably
- Automatic musical genre classification can assist humans or even replace them in this process and would be of a very valuable addition to music information retrieval systems.
- Dividing music into genres is arbitrary, but there are perceptual criteria that are related to instrumentation, structure of the rhythm and texture of the music that can play a role in characterizing particular genre.

1.2 Purpose

- The goal of this project is to build a proof-of-concept music genre classifier using a deep learning approach that can correctly predict the genre and confidence level of Western music from four candidate genres (classical, jazz, rap, rock etc).
- In this music genre classification project, we have developed a classifier on audio files to predict its genre. We work through this project on GTZAN music genre classification dataset. This tutorial explains how to extract important features from audio files. In this deep learning project we have implemented a K nearest neighbor using a count of K.

CHAPTER 2

LITERATURE AND SURVEY

2.1 Exiting problem

Classifying the music without human interaction has been a fascinating problem for lots of people working from different branches like signal processing, machine learning, and music theory. There is a vast amount of research work related to audio and music classification.

A work by Tzanetakis and Cook in (2002), where researchers performed music genre classification using the timbral-related features, texture features, and pitch-related features based on the multi-pitch detection algorithm. Some of the features used in this work include MFCCs, roll-off, and spectral contrast. Their system achieved an overall accuracy.

2.2 Proposed solution

- In our proposed solution, we have compared the performance of several machine learning and deep learning algorithms that we have used for the task of music genre classification.

CHAPTER 3

THEORITICAL ANALYSIS

3.1 Block Diagram

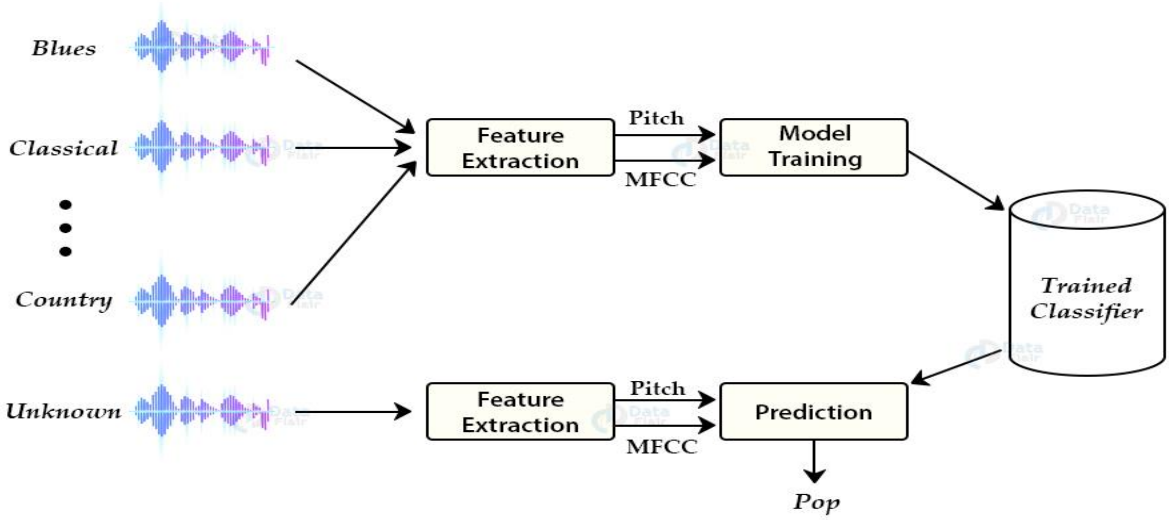

Figure 3.1: Block diagram

3.2 Hardware / Software designing

To do this project we use Anaconda Navigator, Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like Jupyter Lab, Jupyter Notebook, Qt Console, Spyder, Glueviz, Orange, R studio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder.

CHAPTER 4

EXPERIMENTAL INVESTIGATIONS

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

CHAPTER 5

FLOWCHART

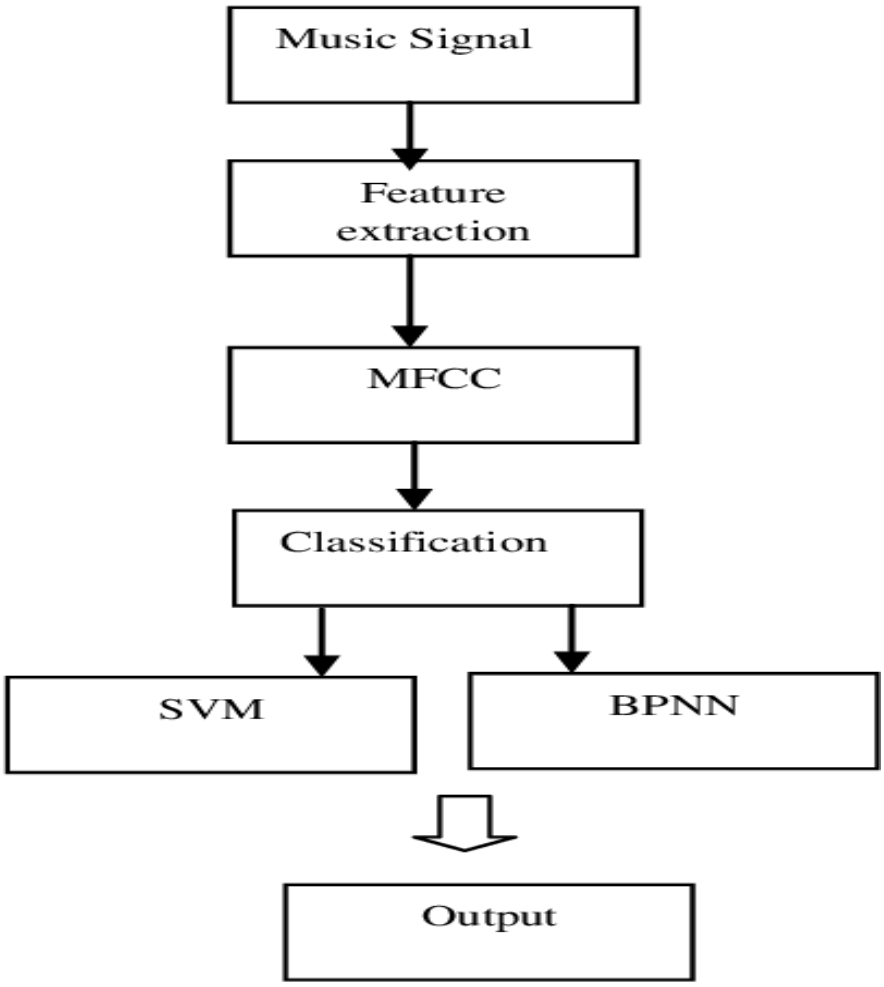

Figure 5.1: Workflow

CHAPTER 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.

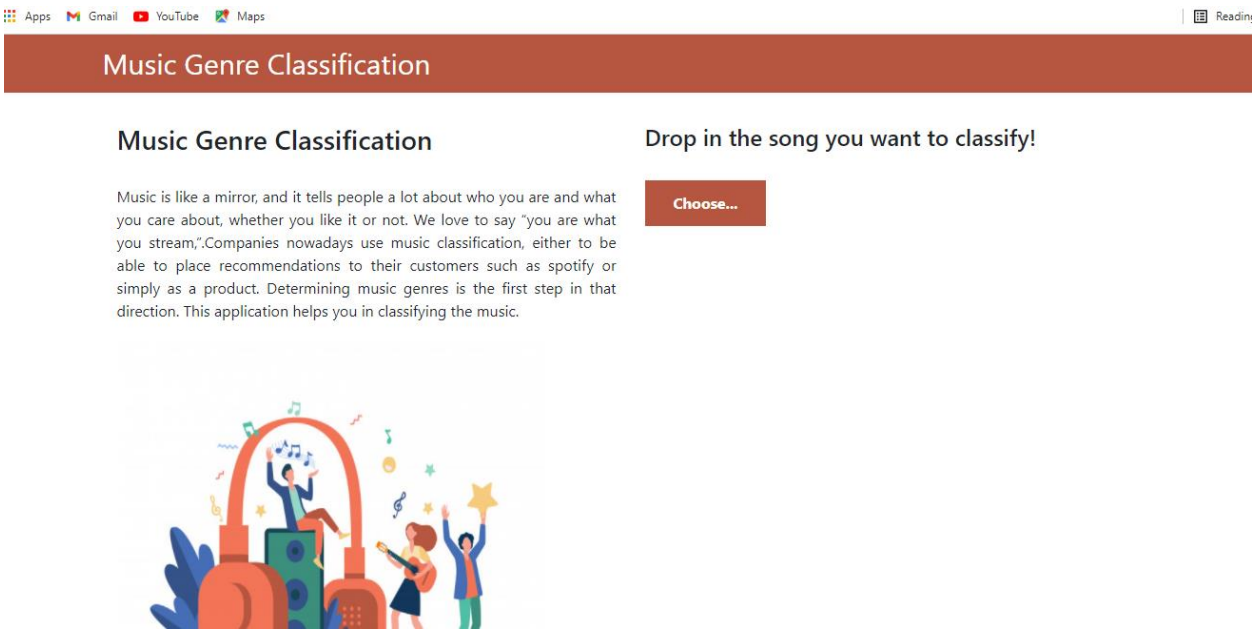

Figure 6.1: Home Page

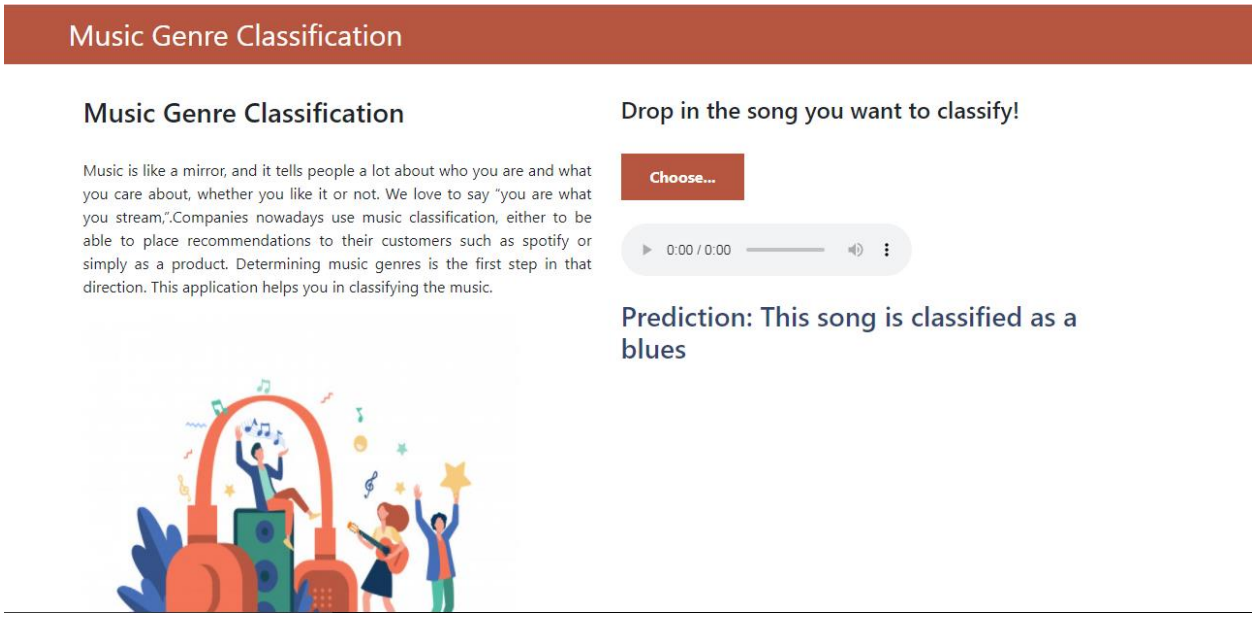

Figure 6.2: Prediction of chosen song

CHAPTER 7

ADVANTAGES & DISADVANTAGES

7.1 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

7.2 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.

CHAPTER 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.

CHAPTER 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.

CHAPTER 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 multiple 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.

CHAPTER 11

BIBILOGRAPHY

- **http://marsyas.info/downloads/datasets.html**

- **https://smartinternz.com/Student/dashboard**

Appendix

a. Source Code

In [1]:
```
import numpy as np
import scipy.io.wavfile as wav
from python_speech_features import mfcc

import os
import pickle
import operator
```

In [2]:
```
def distance(instance1 , instance2 , k ):
    distance =0
    mm1 = instance1[0]
    cm1 = instance1[1]
    mm2 = instance2[0]
    cm2 = instance2[1]
    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
```

In [3]:
```
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
```

In [4]:
```
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]
```

In [5]:
```
directory ="D:/TSB projects/Music genre detection/Music genres/"
f = open("my.dat" ,'wb')
i = 0
```

In [6]:
```
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()
```

In [7]:
```
dataset = []
def loadDataset(project):
    with open("D:\TSB projects\Music genre detection\my.dat" , 'rb') as f:
        while True:
            try:
                dataset.append(pickle.load(f))
            except EOFError:
                f.close()
                break

loadDataset("D:\TSB projects\Music genre detection\my.dat" )
```

In [8]:
```
dataset = np.array(dataset)
```

In [9]:
```
from sklearn.model_selection import train_test_split
x_train ,x_test = train_test_split(dataset,test_size=0.15)
```

In [10]:
```
leng = len(x_test)
predictions = []
for x in range (leng):
    predictions.append(nearestClass(getNeighbors(x_train ,x_test[x] , 5)))
```

In [10]:
```
leng = len(x_test)
predictions = []
for x in range (leng):
    predictions.append(nearestClass(getNeighbors(x_train ,x_test[x] , 5)))
```

In [11]:
```
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)
```

In [12]:
```
accuracy1 = getAccuracy(x_test , predictions)
print(accuracy1)
```

```
0.8066666666666666
```