Mini Project Report

***“AUDIO PROCESSING AND MACHINE LEARNING”***

A Project report submitted to :-

Dr. Richa mishra

**Submitted By:**

ABHISHEK PANDEY

Roll Number :- 1958401

Enrollment Number :- 16AU/490

*In partial fulfillment of the requirements*

*for the award of the degree*

*of*

**Bachelor of Technology**



**JK INSTITUTE OF APPLIED PHYSICS & TECHNOLOGY, UNIVERSITY OF ALLAHABAD, PRAYAGRAJ**

**Certificate**

This is to certify that **Abhishek Pandey**  of B. Tech 6th Semester, Computer Science & Engineering, having enrollment no. 16AU/490 bearing roll no. 1958401 has successfully completed his project on “***Audio Processing and Machine Learning***” for the session 2018-2019 as stated within the syllabus. This project is a bonafide piece of work carried out with the consultation of his guide Dr. Richa Mishra.

**DATE: 15 July, 2019 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**PLACE: Prayagraj Dr. Richa Mishra**

**Contents**

**1.Acknowledgement** ...................................................................................................................................

**2.Abstract** ....................................................................................................................................................

**3.About** .......................................................................................................................................................

**4.Feasibility Study** .......................................................................................................................................

**5.Process Flow……………**...............................................................................................................................

**6. Output……………………………………………………………………………………………………………………………………………**

**7.Conclusion……..** .........................................................................................................................................

**8.References……..** .........................................................................................................................................

**9.Appendix……..** .........................................................................................................................................

**ACKNOWLEDGEMENT**

I have taken eﬀorts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I would like to take the opportunity to express my humble gratitude to Dr. Richa MIshra under whom I executed this project. Her constant guidance and willingness to share her vast knowledge made me understand this project and its manifestations in great depths and helped me to complete the assigned tasks.

I would like to thank all faculty members and staﬀ of the Department of Computer Science and Engineering, J.K Institute of Applied Physics & Technology for their generous help in various ways for the completion of this thesis.

Finally, yet importantly, I would like to express my heartfelt thanks to my beloved parents for their blessings, my friends and classmates for their help and wishes for the successful completion of this project.

***Abhishek Pandey***

**Abstract**

This is a audio processing project using machine learning.

Our system uses musical audio files as input and uses the same to detect the musical note and their onset time . Thereafter we use some machine language algorithms to extract features from the file to detect the instrument playing it.

We have used some python libraries for audio processing and for the instrument detection we have using Support Vector machines(SVM) as our classification algorithm.

For training our SVM we have extracted the features from some predetermined audio files whose values have been stored in a csv file to form our classifier.

*ABOUT*

1.1 Introduction :-

Our system uses musical audio files as input and uses the same to detect the musical note and their onset time . Thereafter we use some machine language algorithms to extract features from the file to detect the instrument playing it.

We have used some python libraries for audio processing and for the instrument detection we have using Support Vector machines(SVM) as our classification algorithm.

1.2 Purpose :-

Our project can segregate all the different instruments which might be playing in an audio file . These results can be used for further applications in various fields.

1.4 System Requirement :-

* Operating System :- Windows/ Linux
* Memory :- 50-60 GB
* RAM :- 4GB
* Internet Connection :-Not required
* Python version:- 2.7

*Feasibility study :-*

2.1 Economic Feasibility:-

Our system has very low economic input and hence can be used by even a common man . The softwares used are open sourced and hence free of cost.

2.2 Technical Feasibility:-

The tools used are open sourced . Python 2.7 can be downloaded free from the internet along with the training audio set which is available free via various websites.

2.3 Behavioural Feasibility:-

The system’s working is quite easy to use and learn due to its simple but attractive interface. User requires no special training for operating the system.

**Process Flow**

In order to identify the notes that are present in the file, we will implement the following steps in the same sequence:

1. Reading the sound file

2. Detecting silence in the file

3. Detecting the location of notes using data obtained from (2)

4. Calculating the frequency of each detected note by using DFT

5. Matching the calculated frequency to the standard frequencies of notes to identify the

note that is being played.

**Modules Used**

We will use the following three Python modules in our implementation:

1. **wave:** To read audio file

2. **struct:** To decode audio file

3. **numpy:** For all numerical computations. Eg. Fourier Transform

**Detecting Silence**

For the detection of silence, one of the approaches that can be implemented involves using a

window of some fixed size. Let us assume that our window is 0.05 seconds long. In terms of the number of samples, this window will have the length equal to 0.05 \*ii . Considering,

ii =44100, this window will of length 2205 .

We can slide this window over the input signal and for each position of window, record the sum of squared elements of input falling within the window. This will roughly be equal to the mean square of amplitude of the signal under the window. If this value falls below a particular threshold, then we can classify that input within that window as silence.

Implementing this using a for loop will work, but it may be inefficient for large audio files.

Interested reader might also try to implement this using convolve function from numpy.

**Detecting Location of Notes**

Once the silence has been detected, it is easy to infer the location of notes by keeping track of start and end index of the input falling within the window. Basically everything that is not silence can be considered as a note.

**Calculating DFT and Identifying Notes**

Once we have the location of each note, we can use Equation 7 to calculate the frequency of the note. To do so, we find iiii by calculating the DFT of the portion of signal at the identified location of the note. The DFT of a signal can be calculated in Python using fft function from numpy.fft module.

numpy.fft.fft(signal) - This function computes the one dimensional Discrete Time Fourier

Transform of the specified signal and returns a complex numpy ndarray.

**Instrument Segregation from Monophonic tones**

In order to identify the instruments present in the monophonic audio, we extract the

different features of that audio given below and then train the SVM classifier.

**Temporal Features -**

 Zero Crossing Rate(ZCR)[mean,var]

 Root Mean Square(RMS)[mean,var]

 Timbral Temporal Descriptors:

 Log Attack Time

 Temporal Centroid

**Spectral Features-**

 Spectral Centroid[mean,var]

 Spectral Spread[mean, var]

 Spectral Flux[mean,var]

 Spectral Flatness[mean,var]

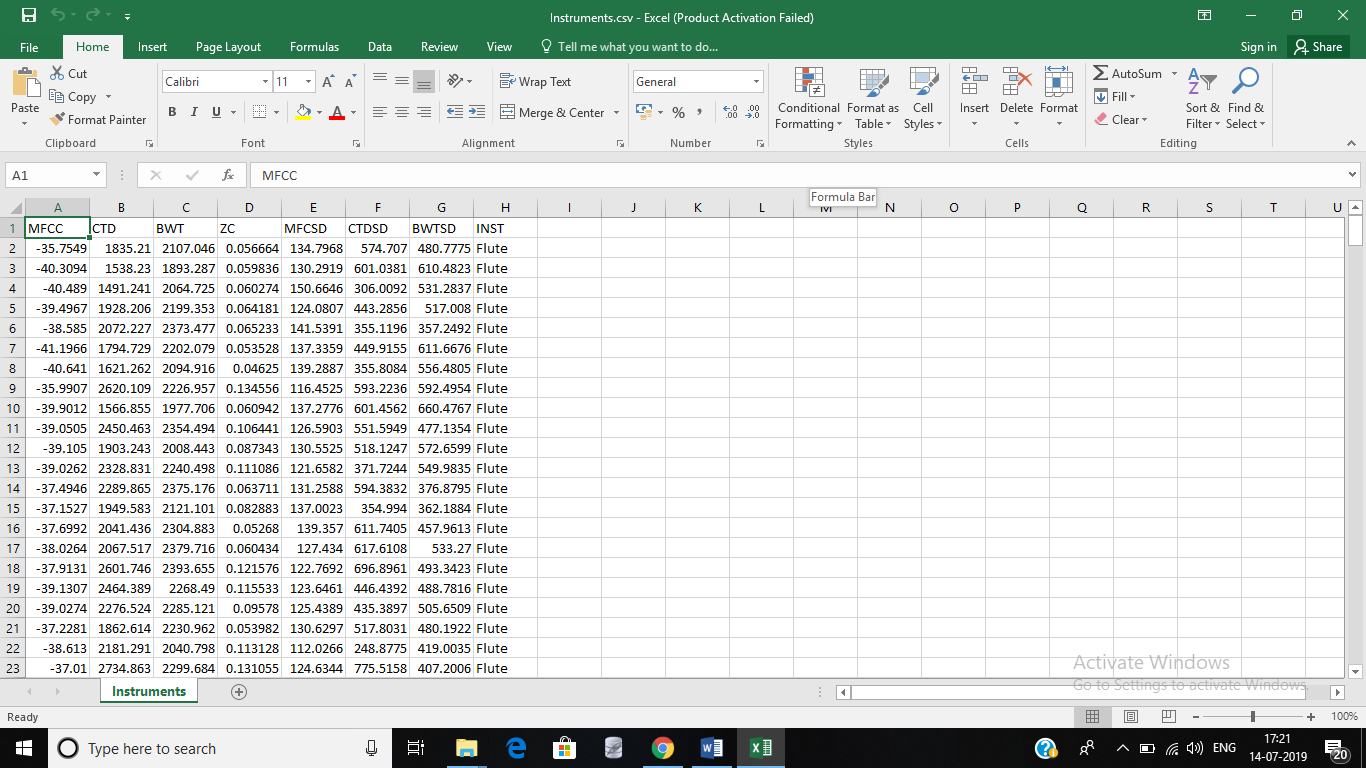
 Spectral Irregularity[mean,var]

 Timbral Spectral Descriptors:

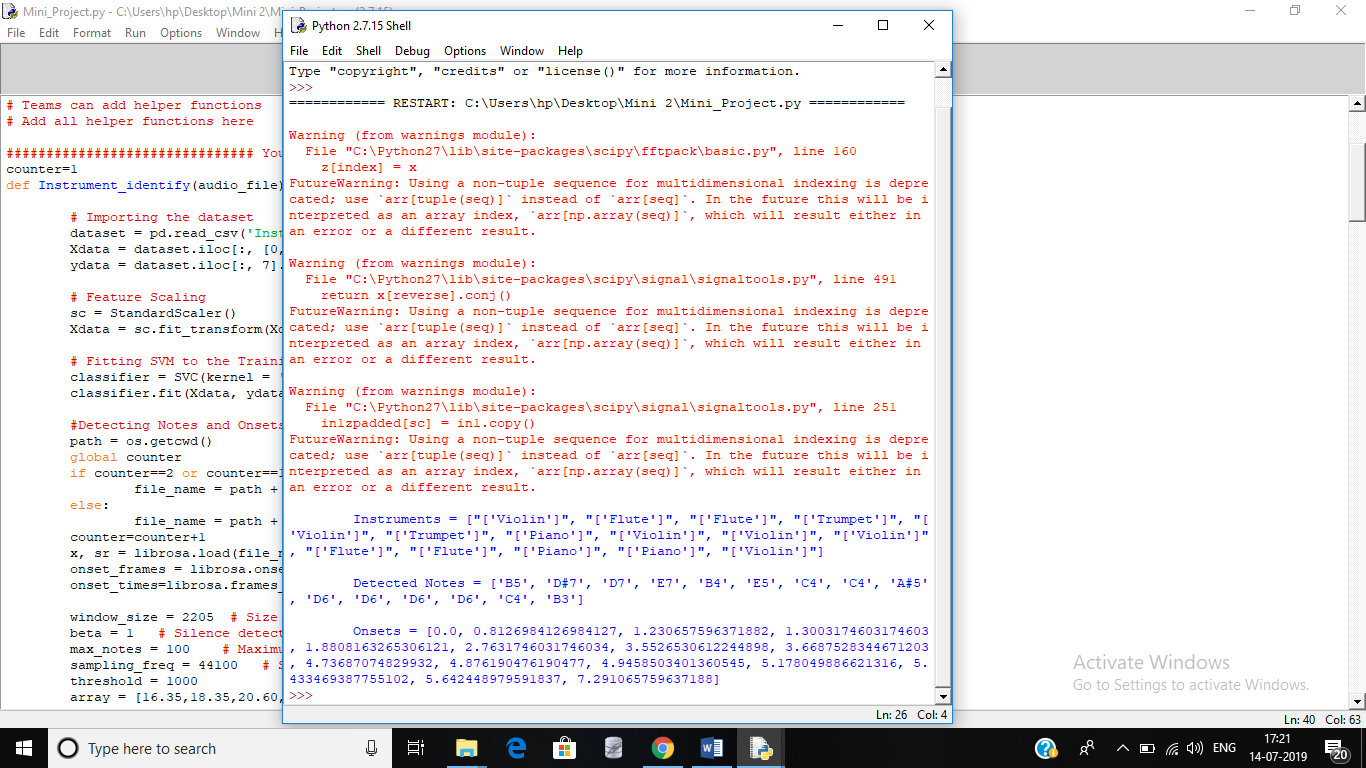
**MFCC**

*Output*

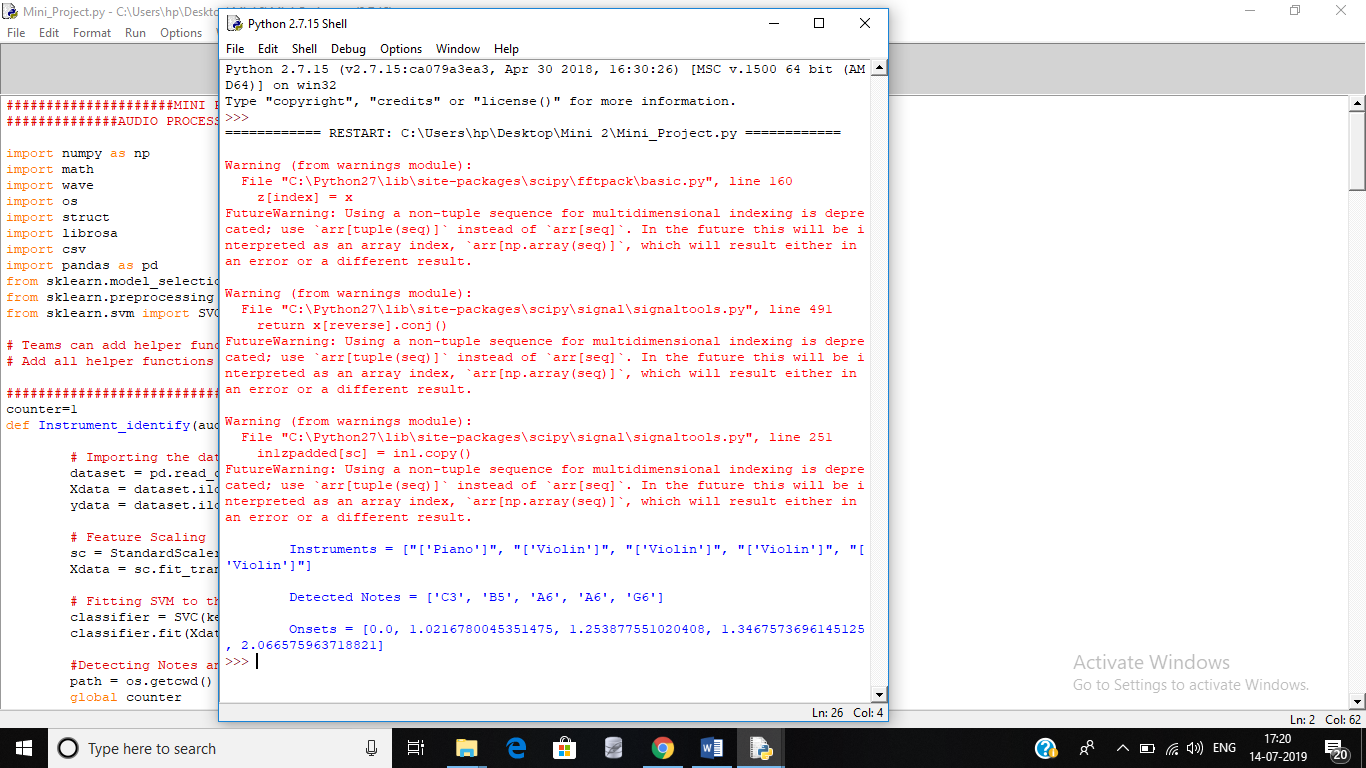
***1:-CSV File Used (For Training of SVM)***

**

***Output of audio\_1.wav***

**

***Output of audio\_2.wav***

**

*Conclusion :-*

*This audio processing project can come in very handy in daily use especially for musicians who are inept at playing various instruments.*

*Musicians can use it to study the audio and the various sounds present in the audio. This system will help them to segregate the instruments making those sounds in the audio.*

*Moreover our system has minimum economic and technical requirements and can hence be used by even the commonest of users .*

References :-

* https://www.coursera.org/lecture/audio-signal-processing/introduction-to-sonic

visualizer-6fRkw

* <https://www.coursera.org/learn/machine-learning/home/week/7>
* <https://www.coursera.org/lecture/audio-signal-processing/f0-detection-tx63Q>

Appendix :-

#####################MINI PROJECT######################

##############AUDIO PROCESSING AND MACHINE LEARNING###########

import numpy as np

import math

import wave

import os

import struct

import librosa

import csv

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVC

# Teams can add helper functions

# Add all helper functions here

counter=1

def Instrument\_identify(audio\_file):

# Importing the dataset

dataset = pd.read\_csv('Instruments.csv')

Xdata = dataset.iloc[:, [0,1,2,3,4,5,6]].values

ydata = dataset.iloc[:, 7].values

# Feature Scaling

sc = StandardScaler()

Xdata = sc.fit\_transform(Xdata)

# Fitting SVM to the Training set

classifier = SVC(kernel = 'linear', random\_state = 0)

classifier.fit(Xdata, ydata)

#Detecting Notes and Onsets

path = os.getcwd()

global counter

if counter==2 or counter==1:

file\_name = path + "\Task\_2\_Audio\_files\Audio\_2.wav"

else:

file\_name = path + "\Task\_2\_Audio\_files\Audio\_"+str(counter)+".wav"

counter=counter+1

x, sr = librosa.load(file\_name)

onset\_frames = librosa.onset.onset\_detect(x, sr=sr, wait=1, pre\_avg=1, post\_avg=1, pre\_max=1, post\_max=1)

onset\_times=librosa.frames\_to\_time(onset\_frames, sr=sr)

window\_size = 2205 # Size of window to be used for detecting silence

beta = 1 # Silence detection parameter

max\_notes = 100 # Maximum number of notes in file, for efficiency

sampling\_freq = 44100 # Sampling frequency of audio signal

threshold = 1000

array = [16.35,18.35,20.60,21.83,24.50,27.50,30.87,

32.70,36.71,41.20,43.65,49.00,55.00,61.74,

65.41,73.42,82.41,87.31,98.00,110.00,123.47,

130.91,146.83,164.81,174.61,196.00,220.00,246.94,

261.63,293.66,329.63,349.23,392.00,440.00,493.88,

523.25,587.33,659.25,698.46,783.99,880.00,987.77,

1046.50, 1174.66, 1318.51, 1396.91, 1567.98, 1760.00, 1975.53,

2093.00, 2349.32, 2637.02, 2793.83, 3135.96, 3520.00, 3951.07,

4186.01, 4698.63, 5274.04, 5587.65, 6271.93, 7040.00, 7902.13,

17.32,19.45,23.12,25.96,29.14,

34.65,38.89,46.25,51.91,58.27,

69.30,77.78,92.50,103.83,116.54,

138.59,155.56,185.00,207.65,233.08,

277.18,311.13,369.99,415.30,466.16,

554.37,622.25,739.99,830.61,932.33,

1108.73,1244.51,1479.98,1661.22,1864.66,

2217.46,2389.02,2959.96,3322.44,3729.31,

4434.92,4978.03,5919.91,6644.88,7458.62]

notes = ['C0', 'D0', 'E0', 'F0', 'G0', 'A0', 'B0',

'C1', 'D1', 'E1', 'F1', 'G1', 'A1', 'B1',

'C2', 'D2', 'E2', 'F2', 'G2', 'A2', 'B2',

'C3', 'D3', 'E3', 'F3', 'G3', 'A3', 'B3',

'C4', 'D4', 'E4', 'F4', 'G4', 'A4', 'B4',

'C5', 'D5', 'E5', 'F5', 'G5', 'A5', 'B5',

'C6', 'D6', 'E6', 'F6', 'G6', 'A6', 'B6',

'C7', 'D7', 'E7', 'F7', 'G7', 'A7', 'B7',

'C8', 'D8', 'E8', 'F8', 'G8', 'A8', 'B8',

'C#0','D#0','F#0','G#0','A#0',

'C#1','D#1','F#1','G#1','A#1',

'C#2','D#2','F#2','G#2','A#2',

'C#3','D#3','F#3','G#3','A#3',

'C#4','D#4','F#4','G#4','A#4',

'C#5','D#5','F#5','G#5','A#5',

'C#6','D#6','F#6','G#6','A#6',

'C#7','D#7','F#7','G#7','A#7',

'C#8','D#8','F#8','G#8','A#8']

Identified\_Notes = []

Onsets=[0.0]

Instruments = []

###################### Read Audio File ###################

path = os.getcwd()

sound\_file = wave.open(file\_name, 'r')

file\_length = sound\_file.getnframes()

sound = np.zeros(file\_length)

mean\_square = []

sound\_square = np.zeros(file\_length)

for i in range(file\_length):

data = sound\_file.readframes(1)

data = struct.unpack("<h", data)

sound[i] = int(data[0])

sound = np.divide(sound, float(2\*\*15)) # Normalize data in range -1 to 1

######################### DETECTING SCILENCE ################

sound\_square = np.square(sound)

frequency = []

dft = []

i = 0

j = 0

k = 0

onset=[0.0]

t=0.00

s = 0.0

while(i<len(sound\_square)-window\_size):

j = 0

while(j<window\_size):

s = s + sound\_square[i+j]

j = j + 1

i=i+window\_size

l=len(sound\_square)/window\_size

S=s/l

#print(S)

i=0

# traversing sound\_square array with a fixed window\_size

while(i<len(sound\_square)-window\_size):

s = 0.0

j = 0

while(j<window\_size):

s = s + sound\_square[i+j]

j = j + 1

# detecting the silence waves

if s<=1000:

if i-k>window\_size\*4:

dft = np.array(dft) # applying fourier transform function

dft = np.fft.fft(sound[k:i])

dft=np.argsort(dft)

if(dft[0]>dft[-1] and dft[1]>dft[-1]):

i\_max = dft[-1]

elif(dft[1]>dft[0] and dft[-1]>dft[0]):

i\_max = dft[0]

else :

i\_max = dft[1]

# calculating frequency

frequency.append((i\_max\*sampling\_freq)/(i-k))

dft = []

k = i+1

onset.append(t)

i = i + window\_size

t=t+0.05

for i in frequency :

idx = (np.abs(array-i)).argmin()

Identified\_Notes.append(notes[idx])

Detected\_Notes=[Identified\_Notes[0]]

for i in onset\_times:

for j in range(len(onset)-1):

if i>=onset[j] and i<=onset[j+1]:

Onsets.append(i)

Detected\_Notes.append(Identified\_Notes[j])

#Extracting Audio Features to Predict the Instruments Except the last one

Len = len(Onsets)

for i in range(0,Len-1):

d = Onsets[i+1] - Onsets[i]

y, sr = librosa.load(file\_name,offset=Onsets[i],duration=d)

hop\_length = 512

y\_harmonic, y\_percussive = librosa.effects.hpss(y)

tempo, beat\_frames = librosa.beat.beat\_track(y=y\_percussive,

sr=sr)

mfcc = librosa.feature.mfcc(y=y, sr=sr, hop\_length=hop\_length, n\_mfcc=13)

centroid=librosa.feature.spectral\_centroid(y=y, sr=sr)

bandwidth=librosa.feature.spectral\_bandwidth(y=y, sr=sr)

zcr=librosa.feature.zero\_crossing\_rate(y=y, frame\_length=2048, hop\_length=512, center=True)

mfc=np.mean(mfcc)

ctd=np.mean(centroid)

bwt=np.mean(bandwidth)

zc=np.mean(zcr)

mfcsd=np.std(mfcc)

ctdsd=np.std(centroid)

bwtsd=np.std(bandwidth)

zcsd=np.std(zc)

X\_test = sc.transform([[mfc,ctd,bwt,zc,mfcsd,ctdsd,bwtsd]])

# Predicting the Test set results

y\_pred = classifier.predict(X\_test)

y\_pred = str(y\_pred)

Instruments.append(y\_pred)

#Predicting the last Instrument

y, sr = librosa.load(file\_name,offset=Onsets[i])

hop\_length = 512

y\_harmonic, y\_percussive = librosa.effects.hpss(y)

tempo, beat\_frames = librosa.beat.beat\_track(y=y\_percussive,

sr=sr)

mfcc = librosa.feature.mfcc(y=y, sr=sr, hop\_length=hop\_length, n\_mfcc=13)

centroid=librosa.feature.spectral\_centroid(y=y, sr=sr)

bandwidth=librosa.feature.spectral\_bandwidth(y=y, sr=sr)

zcr=librosa.feature.zero\_crossing\_rate(y=y, frame\_length=2048, hop\_length=512, center=True)

mfc=np.mean(mfcc)

ctd=np.mean(centroid)

bwt=np.mean(bandwidth)

zc=np.mean(zcr)

mfcsd=np.std(mfcc)

ctdsd=np.std(centroid)

bwtsd=np.std(bandwidth)

zcsd=np.std(zc)

X\_test = sc.transform([[mfc,ctd,bwt,zc,mfcsd,ctdsd,bwtsd]])

y\_pred = classifier.predict(X\_test)

y\_pred = str(y\_pred)

Instruments.append(y\_pred)

Onsets = [0.00, 0.99, 1.32, 2.04]

return Instruments, Detected\_Notes, Onsets

############################# Main Function #######################

if \_\_name\_\_ == "\_\_main\_\_":

path = os.getcwd()

file\_name = path + "/Task\_2\_Audio\_files/Audio\_1.wav"

audio\_file = wave.open(file\_name)

Instruments, Detected\_Notes, Onsets = Instrument\_identify(audio\_file)

print("\n\tInstruments = " + str(Instruments))

print("\n\tDetected Notes = " + str(Detected\_Notes))

print("\n\tOnsets = " + str(Onsets))

# code for checking output for all audio files