SPEAKER RECOGNITION USING MFCC

by

Senthil Vel K	19BEC1122
S R Shivaritha	19BEC1130
Surya Y	19BEC1176

A project report submitted to

Dr. Annis Fathima A

SCHOOL OF ELECTRONICS AND COMMUNICATION ENGINEERING

in partial fulfillment of the requirements for the course of

ECE2006 – DIGITAL SIGNAL PROCESSING

in

B.Tech. ELECTRONICS AND COMMUNICATION ENGINEERING

VIT UNIVERSITY, CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

AUGUST 2021

BONAFIDE CERTIFICATE

Certified that this project report entitled "**SPEAKER RECOGNITION USING MFCC"** is a bonafide work of **SENTHIL VEL K (19BEC1122), S R SHIVARITHA (19BEC1130) and SURYA Y (19BEC1176)** who carried out the project work under my supervision and guidance.

Dr. Annis Fathima A

Associate Professor

School of Electronics Engineering

(SENSE), VIT University, Chennai

Chennai – 600127.

ABSTRACT

In this project, a voice recognition algorithm using python is implemented for speaker recognition. Speaker Recognition is the problem of identifying a speaker from a recording of their speech. Voice controlled devices rely heavily on speaker recognition.

The main goal is to identify the speaker from a given set of data samples. We use MFCC to identify the key features and train the data with Vector quantization and predict the output. The system is implemented using python and multiple datasets are used to test the reliability and accuracy of the prediction.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Annis Fathima A,** Associate Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Siva Subramanian A**, Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support.

We express our thanks to our Program chair **Dr. Vetrivelan P** for their support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

TABLE OF CONTENTS

1. INTRODUCTION

Speech is a basic common biometric property. The application of speech signals is endless in the field of speech signal processing. The problem of identifying a speaker from a recorded set of audio files is called Speaker Recognition. In security systems speaker recognition concepts are used to increase safety and reliability. Many modern devices and applications rely on speaker recognition for its use.

The main principle in speaker recognition is extracting features from each speech, thereby analyzing the characteristics of each speaker. The difference in resonance of vocal tracts in each speaker is found through MFCC. The Mel Frequency Cepstral Coefficients of each speaker is extracted and Vector Quantization using LBG algorithm is applied to train the given data. The data set consists of 8 different speakers pronouncing the word "zero" and 7 different speakers pronouncing the word "hello" in a monotonous voice. The variation of Speaker exists in speech signals because of different resonances of the vocal tract. MFCC is the technique to exploit the differences of the speech signal.

In this project python is used for its scientific computing abilities that use packages like numpy, scipy and matplotlib.

1.1 OBJECTIVES AND GOALS

- To design a Speaker Recognition System using Python
- Identify speakers with maximum accuracy.

1.2 BENEFITS

A lot of technologies use speaker recognized applications for ease of use and security purposes. The scope of speaker recognition is endless in security systems. Speaker recognition can be used to identify suspected people in crime scenes as well. Speaker recognition can be used for unlocking secret vaults or office doors. It can also be used to detect the presence of students and employees. This system can come in great use to identify the speaker for people with dementia

1.3 FEATURES

- Speakers are recognized with high accuracy.
- Monotonic audio files are identified.

2. DESIGN

2.1 BLOCK DIAGRAM

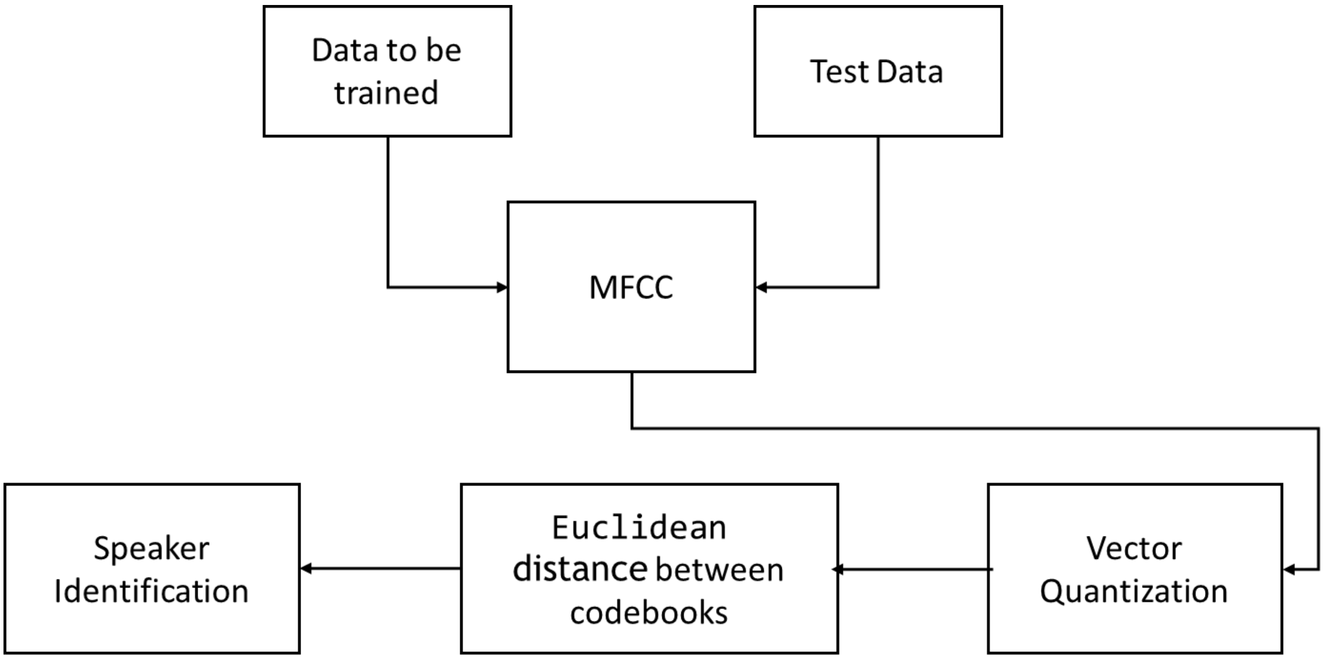


figure1. Block Diagram .

3. SOFTWARE IMPLEMENTATION

3.1 PROCEDURE

The process of speaker recognition consists of the training phase and the recognition phase. In the training phase, the features of a speaker's speech signal are stored as reference features. The feature vectors of speech are used to create a speakerís model.

- The audio files are sampled at a rate 12500hz with a bit rate of 200kbits/s.
- The user enters the input to select the data set.
- The accuracy and speaker to be identified can be selected in the menu option displayed.
- The speaker file to be identified is selected. The data set begins to train.
- The MFCC characteristics are extracted for the training data set by,
- Codebooks are formed using the LBG algorithm for the extracted MFCC characteristics.
- Codebooks for each speaker are plotted.
- The speaker is identified by computing the nearest distance from each codebook for the selected audio file.
- The speaker is recognized with their respective audio file. The user can choose to continue the process or stop.

3.2 PYTHON CODE

<u>Function 1 (Feature Extraction - Mel-Frequency Cepstral Coefficients)</u>

In this function the characteristic extraction using MFCC takes place. The detailed schematic is given below,

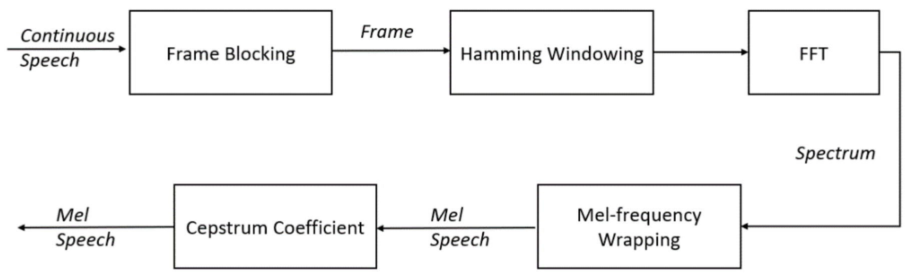


figure 2. Block Diagram of MFCC.

- The audio files are divided into frames and are multiplied with a hamming window.

Fast Fourier Transform (FFT) is applied to 512 samples on each frame. Thus, the periodogram is calculated. The power spectrum is obtained through,

$$P(k) = \frac{1}{N}|S(k)|^2$$

- The frequency range is set from 1hz to 16000hz. This range is divided into “n” Mel filter banks. The value of n is 22.
- Mel-spaced filterbank is computed and frequency is warped on Mel scale.
- The energies of the filter banks are calculated by multiplying the filter bank with the power spectrum.
- Logarithms of each filter bank are taken and Discrete Cosine Transform (DCT) is applied to correlate the value of Mel spectrum.

```python
from __future__ import division
from scipy.signal import hamming
from scipy.fftpack import fft, fftshift, dct
import numpy as np
import matplotlib.pyplot as plt

def hertz_to_mel(freq):
    return 1125 * np.log(1 + freq / 700)

def mel_to_hertz(m):
    return 700 * (np.exp(m / 1125) - 1)

# calculate mel frequency filter bank
def mel_filterbank(nfft, nfiltbank, fs):
    # set limits of mel scale from 300Hz to 8000Hz
    lower_mel = hertz_to_mel(1)
    upper_mel = hertz_to_mel(16000)
    mel = np.linspace(lower_mel, upper_mel, nfiltbank + 2)
    hertz = [mel_to_hertz(m) for m in mel]
    fbins = [int(hz * (nfft / 2 + 1) / fs) for hz in hertz]
    fbank = np.empty((int(nfft / 2 + 1), nfiltbank))
    for i in range(1, nfiltbank + 1):
        for k in range(int(nfft / 2 + 1)):
            if k < fbins[i - 1]:
                fbank[k, i - 1] = 0
            elif k >= fbins[i - 1] and k < fbins[i]:
                fbank[k, i - 1] = (k - fbins[i - 1]) / (fbins[i] - fbins[i - 1])
            elif k >= fbins[i] and k <= fbins[i + 1]:
                fbank[k, i - 1] = (fbins[i + 1] - k) / (fbins[i + 1] - fbins[i])
            else:
                fbank[k, i - 1] = 0
    return fbank
def mfcc(s, fs, nfiltbank,h_z):
    # divide into segments of 25 ms with overlap of 10ms
```

```python
nSamples = np.int32(0.025 * fs)
overlap = np.int32(0.01 * fs)
nFrames = np.int32(np.ceil(len(s) / (nSamples - overlap)))

# zero padding to make signal length long enough to have nFrames
padding = ((nSamples - overlap) * nFrames) - len(s)

if padding > 0:
    signal = np.append(s, np.zeros(padding))

else:
    signal = s
segment = np.empty((nSamples, nFrames))
start = 0
for i in range(nFrames):
    segment[:, i] = signal[start:start + nSamples]
    start = (nSamples - overlap) * i

# compute periodogram
nfft = 512
periodogram = np.empty((nFrames, int(nfft / 2 + 1)))
for i in range(nFrames):
    x = segment[:, i] * hamming(nSamples)
    spectrum = fftshift(fft(x, nfft))
    periodogram[i, :] = abs(spectrum[int(nfft / 2 - 1):]) / nSamples

# calculating mfccs
fbank = mel_filterbank(nfft, nfiltbank, fs)
# nfiltbank MFCCs for each frame
mel_coeff = np.empty((nfiltbank, nFrames))
for i in range(nfiltbank):
    for k in range(nFrames):
        mel_coeff[i, k] = np.sum(periodogram[k, :] * fbank[:, i])
# print(mel_coeff)
if (h_z=="h"):
    mel_coeff = np.log10(mel_coeff)
    mel_coeff = dct(mel_coeff)
    # exclude 0th order coefficient (much larger than others)
    mel_coeff[0, :] = np.zeros(nFrames)
else:
    mel_coeff = dct(mel_coeff)
# exclude 0th order coefficient (much larger than others)
    mel_coeff[0, :] = np.zeros(nFrames)

return mel_coeff
```

Function 2 (Feature Matching – LBG algorithm)

In this function we use the LBG algorithm for clustering a set of training vectors into a set of codebook vectors.

- The first codebook is chosen randomly. This is set as the centroid from which other vectors are clustered to the nearest distance.
- For the rest of the codebook, Nearest neighbor search is done to train each vector and the codebook is found accordingly and assigned to it.
- The codeword in each cell is updated using the centroid assigned to that cell in the training vector.
- On repeating the process for each vector, a codebook size of M is designed.

The distance of the speaker's features from all the trained codebooks is calculated to identify a speaker. The speaker is matched with the codebook that has minimum distortion (distance).

```
# speaker specific Vector Quantization codebook using LBG algorithm
from __future__ import division
import numpy as np

# calculate Euclidean distance between two matrices
def EUDistance(d, c):
    # np.shape(d)[0] = np.shape(c)[0]
    n = np.shape(d)[1]
    p = np.shape(c)[1]
    distance = np.empty((n, p))

    if n < p:
        for i in range(n):
            copies = np.transpose(np.tile(d[:, i], (p, 1)))
            distance[i, :] = np.sum((copies - c) ** 2, 0)
    else:
        for i in range(p):
            copies = np.transpose(np.tile(c[:, i], (n, 1)))
            distance[:, i] = np.transpose(np.sum((d - copies) ** 2, 0))

    distance = np.sqrt(distance)
    return distance

def lbg(features, M):
    eps = 0.01
    codebook = np.mean(features, 1)
    distortion = 1
    nCentroid = 1
    while nCentroid < M:

        # double the size of codebook
        new_codebook = np.empty((len(codebook), nCentroid * 2))
        if nCentroid == 1:

            new_codebook[:, 0] = codebook * (1 + eps)
            new_codebook[:, 1] = codebook * (1 - eps)
        else:
            for i in range(nCentroid):
                new_codebook[:, 2 * i] = codebook[:, i] * (1 + eps)
                new_codebook[:, 2 * i + 1] = codebook[:, i] * (1 - eps)
        codebook = new_codebook
        nCentroid = np.shape(codebook)[1]
        D = EUDistance(features, codebook)

        while np.abs(distortion) > eps:
            # nearest neighbour search

            prev_distance = np.mean(D)
            nearest_codebook = np.argmin(D, axis=1)
            # cluster vectors and find new centroid
            for i in range(nCentroid):
                # add along 3rd dimension
                codebook[:, i] = np.mean(features[:, np.where(nearest_codebook ==
i)], 2).T
                # replace all NaN values with 0
            codebook = np.nan_to_num(codebook)
            D = EUDistance(features, codebook)
            distortion = (prev_distance - np.mean(D)) / prev_distance
    return codebook
```

Function 3 (Feature Training)

The algorithm to detect has been implemented. Now the data needs to be trained and codebooks are derived for each speaker using VQ. The number of speakers is nSpeaker = 7. Speech recordings of 7 speakers uttering the word 'hello' have been taken for training and testing. Each codebook should have 16 codewords, hence nCentroid = 16.

```
from __future__ import division
import numpy as np
from scipy.io.wavfile import read
from LBG import lbg
from mel_coefficients import mfcc
import matplotlib.pyplot as plt
fname = str()
def trainingh(nfiltbank):
    nSpeaker = 7
    nCentroid = 16
    codebooks_mfcc = np.empty((nSpeaker, nfiltbank, nCentroid))
    directoryh = "hello/train_hello"
     names = {1: "surya", 2: "yuvan", 3: "khamalesh", 4: "lekha", 5: "lalitha", 6: 
"senthil", 7: "shivaritha"}
    h_z="h"
    for i in range(nSpeaker):
        fname = '/s' + str(i + 1) + '.wav'
        print('Now speaker ', str(i + 1), 'features are being trained')
        (fs, s) = read(directoryh + fname)
        mel_coeff = mfcc(s, fs, nfiltbank,h_z)
        codebooks_mfcc[i, :, :] = lbg(mel_coeff, nCentroid)
        plt.figure(i)
         plt.title('Codebook for speaker, '+str(i+1)+',' + names[i + 1] + ' with ' 
+ str(nCentroid) + ' centroids')
        for j in range(nCentroid):
            plt.stem(codebooks_mfcc[i, :, j])
            plt.ylabel('MFCC')
            plt.xlabel('Number of features')
    plt.show()
    print('Training complete')
    # plotting 5th and 6th dimension MFCC features on a 2D plane
    codebooks = np.empty((2, nfiltbank, nCentroid))
    mel_coeff = np.empty((2, nfiltbank, 68))
    for i in range(2):
        fname = '/s' + str(i + 1) + '.wav'
        (fs, s) = read(directoryh + fname)
        mel_coeff[i, :, :] = mfcc(s, fs, nfiltbank,h_z)[:, 0:68]
        codebooks[i, :, :] = lbg(mel_coeff[i, :, :], nCentroid)
    return (codebooks_mfcc)
```

Function 4 (Feature Training)

Speech files are stored and the .wav filenames are passed as parameters to the training() function. The number of speakers is nSpeaker = 8. Speech recordings of 8 speakers uttering the word ‘zero’ have been taken for training and testing. Each codebook should have 16 codewords, hence nCentroid = 16.

```
from __future__ import division
import numpy as np
from scipy.io.wavfile import read
from LBG import lbg
from mel_coefficients import mfcc
import matplotlib.pyplot as plt

fname = str()
def trainingz(nfiltbank):
    nSpeaker = 8
    nCentroid = 16
    codebooks_mfcc = np.empty((nSpeaker, nfiltbank, nCentroid))
    directoryz = "zero/train"
    h_z="z"
     names = {1: "gaurav", 2: "khamalesh", 3: "yuvan", 4: "lekha", 5: "lalitha",
6: "shivaritha", 7: "surya",
             8: "Senthil"}
    for i in range(nSpeaker):
        fname = '/s' + str(i + 1) + '.wav'
        print('Now speaker ', str(i + 1), 'features are being trained')
        (fs, s) = read(directoryz + fname)
        mel_coeff = mfcc(s, fs, nfiltbank,h_z)
        codebooks_mfcc[i, :, :] = lbg(mel_coeff, nCentroid)
        plt.figure(i)
         plt.title('Codebook for speaker, '+str(i+1)+',' + names[i+1] + ' with ' +
str(nCentroid) + ' centroids')
        for j in range(nCentroid):
            # plt.subplot(211)
            plt.stem(codebooks_mfcc[i, :, j])
            plt.ylabel('MFCC')
            plt.xlabel('Number of features')
    plt.show()
    print('Training complete')
    # plotting 5th and 6th dimension MFCC features on a 2D plane
    codebooks = np.empty((2, nfiltbank, nCentroid))
    mel_coeff = np.empty((2, nfiltbank, 68))
    for i in range(2):
        fname = '/s' + str(i + 1) + '.wav'
        (fs, s) = read(directoryz + fname)
        mel_coeff[i, :, :] = mfcc(s, fs, nfiltbank,h_z)[:, 0:68]
        codebooks[i, :, :] = lbg(mel_coeff[i, :, :], nCentroid)
    return (codebooks_mfcc)
```

Function 5 (Testing)

We test our speaker recognition algorithm. This contains the code for identifying the speaker by comparing their feature vector to the codebooks of all trained speakers and computing the minimum distance between them.

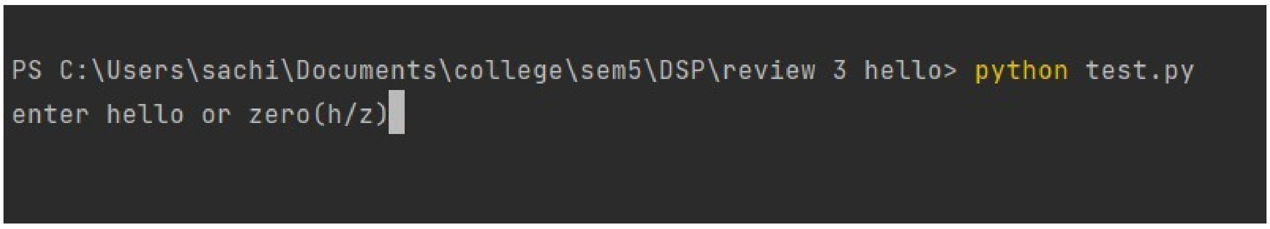
```
from __future__ import division
import numpy as np
from scipy.io.wavfile import read
from LBG import EUDistance
from mel_coefficients import mfcc
from train import trainingh
from trainz import trainingz

nfiltbank = 22

directoryz = 'zero/test'
directoryh='hello/test_hello'
directory1 = 'zero/speaker'
directory2='hello/speaker'

def minDistance(features, codebooks):
    speaker = 0
    distmin = np.inf
    for k in range(np.shape(codebooks)[0]):
        D = EUDistance(features, codebooks[k, :, :])
        dist = np.sum(np.min(D, axis=1)) / (np.shape(D)[0])
        if dist < distmin:
            distmin = dist
            speaker = k
    return speaker

def h1():
    nSpeaker = 7
    nCorrect_MFCC = 0
    for i in range(nSpeaker):
        fname = '/s' + str(i+1) + '.wav'
        print('Now speaker ', str(i + 1), 'features are being tested')
        (fs, s) = read(directoryh + fname)
        mel_coefs = mfcc(s, fs, nfiltbank,h_z)
        sp_mfcc = minDistance(mel_coefs, codebooks_mfcc)
        print('Speaker(hello)', (i + 1), ' in test matches with speaker ',
(sp_mfcc + 1), 'in train for training with MFCC')
        if i == sp_mfcc:
            nCorrect_MFCC += 1
    percentageCorrect_MFCC = (nCorrect_MFCC / nSpeaker) * 100
    print('Accuracy of result for training with MFCC is ',
percentageCorrect_MFCC, '%')
```

```
def h2():
    nSpeaker = 7
    nCorrect_MFCC = 0
    fname = input("enter wav file: ")
    for i in range(nSpeaker):
        print('Now speaker ', str(i + 1), 'features are being tested')
        (fs, s) = read(directory2 + fname)
        mel_coefs = mfcc(s, fs, nfiltbank,h_z)
        sp_mfcc = minDistance(mel_coefs, codebooks_mfcc)

        if i == sp_mfcc:
            nCorrect_MFCC += 1
    names = {1: "surya", 2: "yuvan", 3: "khamalesh", 4: "lekha", 5: "lalitha",
6:"senthil", 7: "shivaritha" }
    print('Identified Speaker:', names[sp_mfcc + 1])

def z1():
    nSpeaker = 8
    nCorrect_MFCC = 0
    for i in range(nSpeaker):
        fname = '/s' + str(i+1) + '.wav'
        print('Now speaker ', str(i + 1), 'features are being tested')
        (fs, s) = read(directoryz + fname)
        mel_coefs = mfcc(s, fs, nfiltbank,h_z)
        sp_mfcc = minDistance(mel_coefs, codebooks_mfcc)
        print('Speaker(zero)', (i + 1), ' in test matches with speaker ',
(sp_mfcc + 1), 'in train for training with MFCC')

        if i == sp_mfcc:
            nCorrect_MFCC += 1
    percentageCorrect_MFCC = (nCorrect_MFCC / nSpeaker) * 100
    print('Accuracy of result for training with MFCC is ',
percentageCorrect_MFCC, '%')

def z2():
    nSpeaker = 7
    nCorrect_MFCC = 0
    fname = input("enter wav file: ")
    for i in range(nSpeaker):
        print('Now speaker ', str(i + 1), 'features are being tested')
        (fs, s) = read(directory1 + fname)
        mel_coefs = mfcc(s, fs, nfiltbank,h_z)
        sp_mfcc = minDistance(mel_coefs, codebooks_mfcc)

        if i == sp_mfcc:
            nCorrect_MFCC += 1
    names = {1: "gaurav", 2: "khamalesh", 3: "yuvan", 4: "lekha", 5: "lalitha",
6:"shivaritha", 7: "surya", 8: "Senthil" }
    print('Identified Speaker:', names[sp_mfcc + 1])
cont=False
while cont==False:
    h_z = input("enter hello or zero(h/z)")
    if h_z == "h":
        (codebooks_mfcc) = trainingh(nfiltbank)
    else:
        (codebooks_mfcc) = trainingz(nfiltbank)
    c=False
    while c==False:
        inp = input("enter choice:\n1.show percentage(p)\n2.identify
speaker(s)\n")
        if inp=="1":
            if h_z=="h":
                h1()
            else:
                z1()
        elif inp=="2":
            if h_z=="h":
                h2()
            else:
                z2()
        y1=input("do u wanna change word? ")
        if y1 == "y":
            c = True
    y=input("do u want to continue?(y/n): ")
    if y=="n":
        cont=True
        break
```

4. SIMULATION OUTPUTS:

4.1 TRAINING THE DATASET FOR WORD "HELLO"

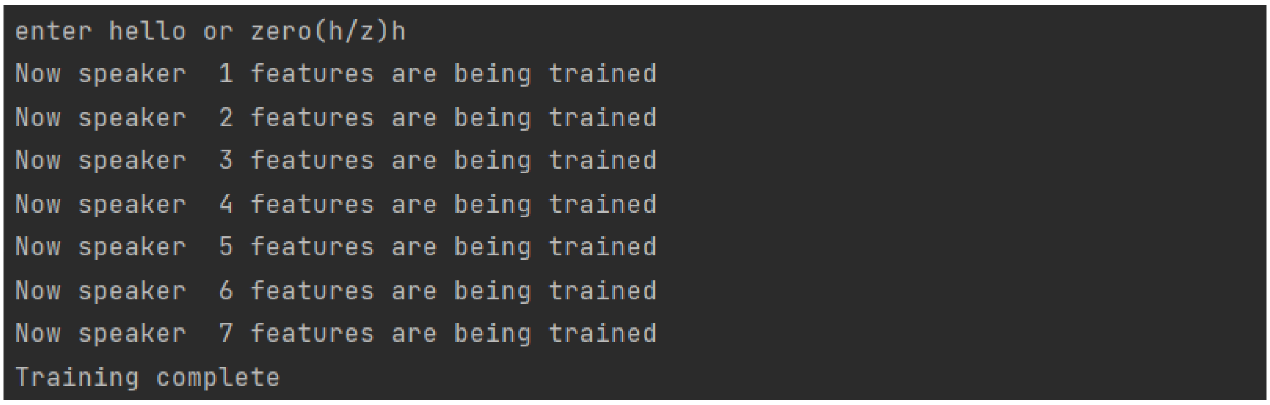
In the python console, the test file of the code is executed. This test file contains a function which calls other python files in the project.



```
PS C:\Users\sachi\Documents\college\sem5\DSP\review 3 hello> python test.py
enter hello or zero(h/z)
```

figure 3. Test file is executed.

Users can enter whether they want to use the word "hello" or "zero". Depending on the user input(“hello”), the dataset present in the train folder gets accessed and is trained.



```
enter hello or zero(h/z)h
Now speaker  1 features are being trained
Now speaker  2 features are being trained
Now speaker  3 features are being trained
Now speaker  4 features are being trained
Now speaker  5 features are being trained
Now speaker  6 features are being trained
Now speaker  7 features are being trained
Training complete
```

figure 4. Features are trained.

4.2 MFCC VS NUMBER OF FEATURES (CODEBOOK)

The codebooks for the trained dataset of hello are displayed.

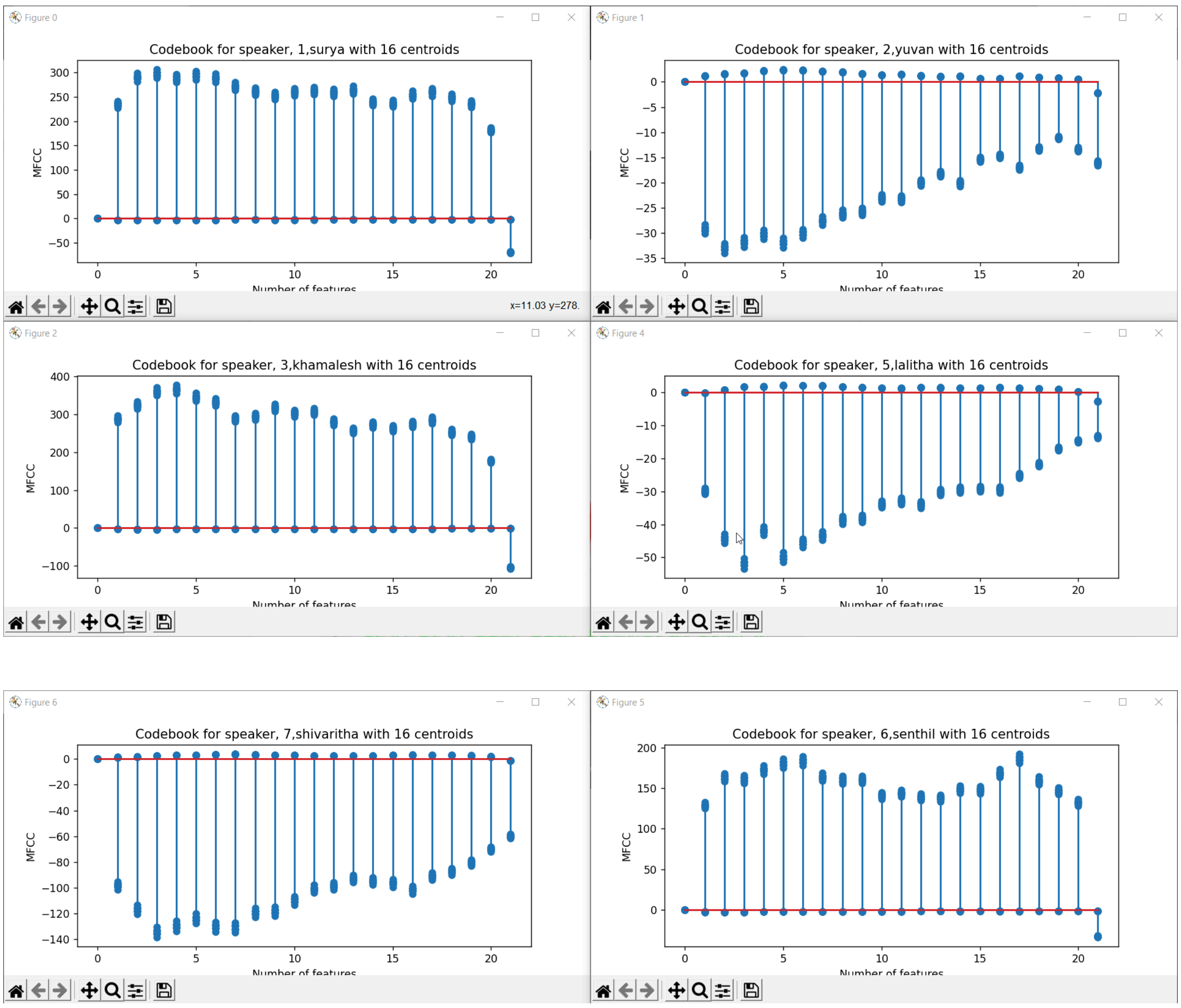


figure 5. Codebooks for each speaker.

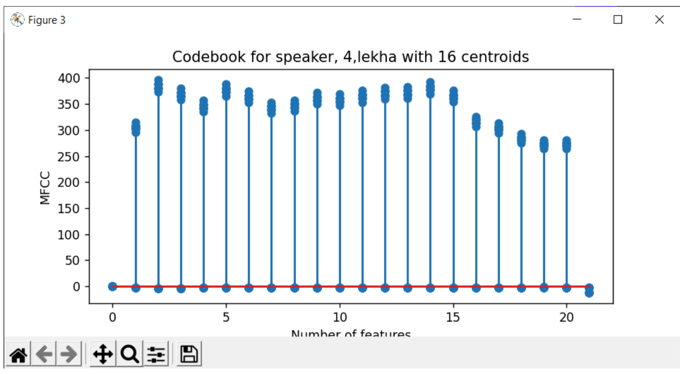
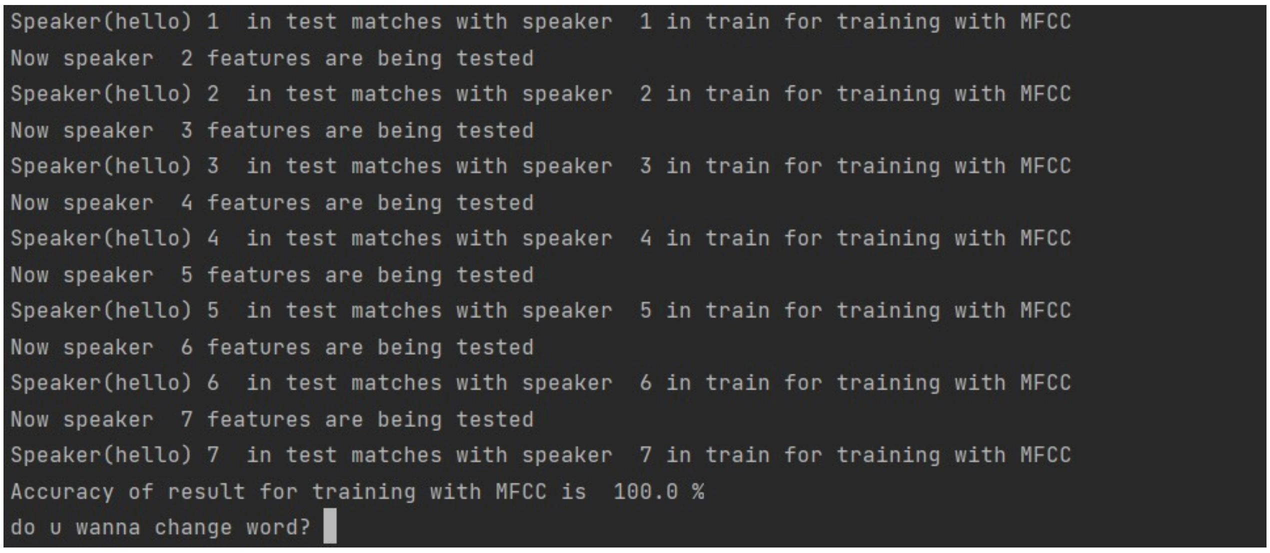


figure 6. Codebooks for each speaker.

4.3 ACCURACY OF SPEAKER RECOGNITION

Users can either choose to display the accuracy or to recognise the speaker using an audio file.
The test data is matched with train data.

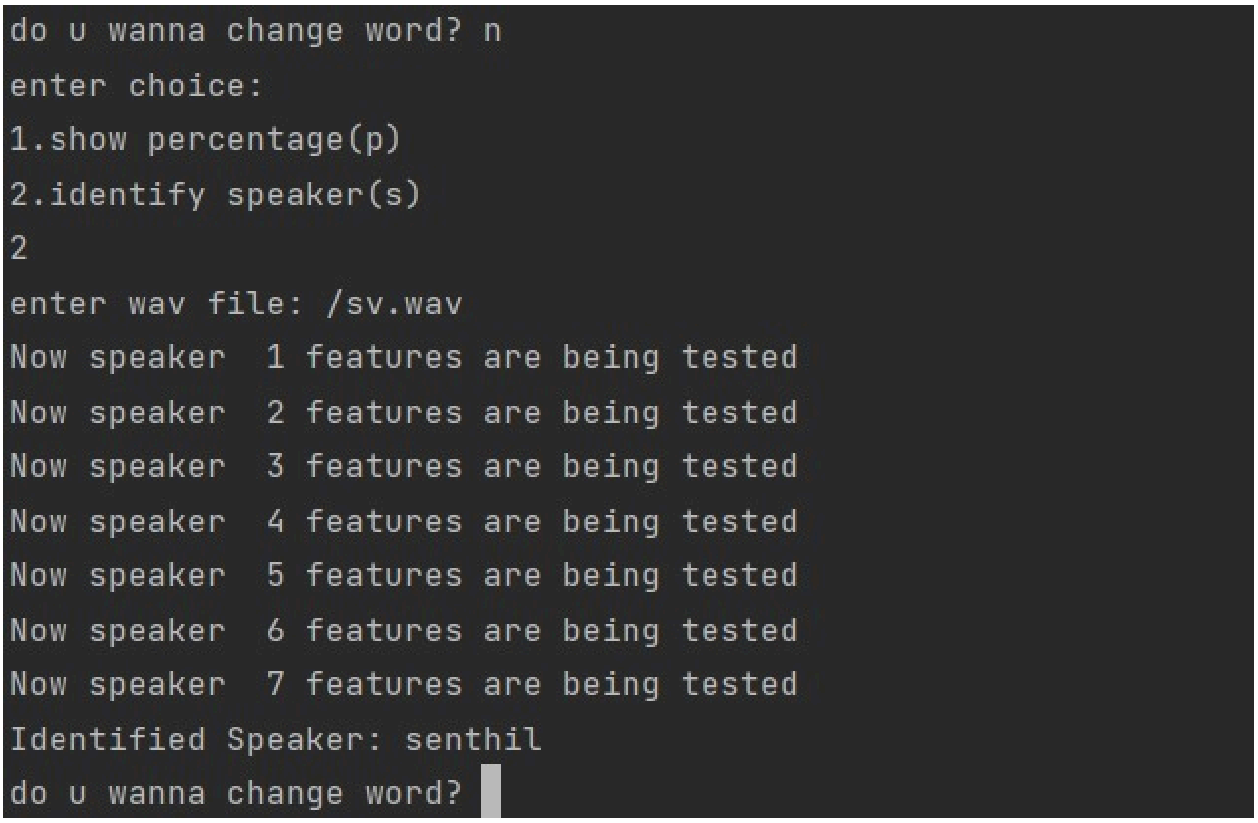


```
Speaker(hello) 1  in test matches with speaker  1 in train for training with MFCC
Now speaker  2 features are being tested
Speaker(hello) 2  in test matches with speaker  2 in train for training with MFCC
Now speaker  3 features are being tested
Speaker(hello) 3  in test matches with speaker  3 in train for training with MFCC
Now speaker  4 features are being tested
Speaker(hello) 4  in test matches with speaker  4 in train for training with MFCC
Now speaker  5 features are being tested
Speaker(hello) 5  in test matches with speaker  5 in train for training with MFCC
Now speaker  6 features are being tested
Speaker(hello) 6  in test matches with speaker  6 in train for training with MFCC
Now speaker  7 features are being tested
Speaker(hello) 7  in test matches with speaker  7 in train for training with MFCC
Accuracy of result for training with MFCC is  100.0 %
do u wanna change word?
```

figure 7. Each audio file is tested.

4.4 SPEAKER RECOGNITION

Users can either change the word or continue.
Now the audio file is given as input and speaker recognition is

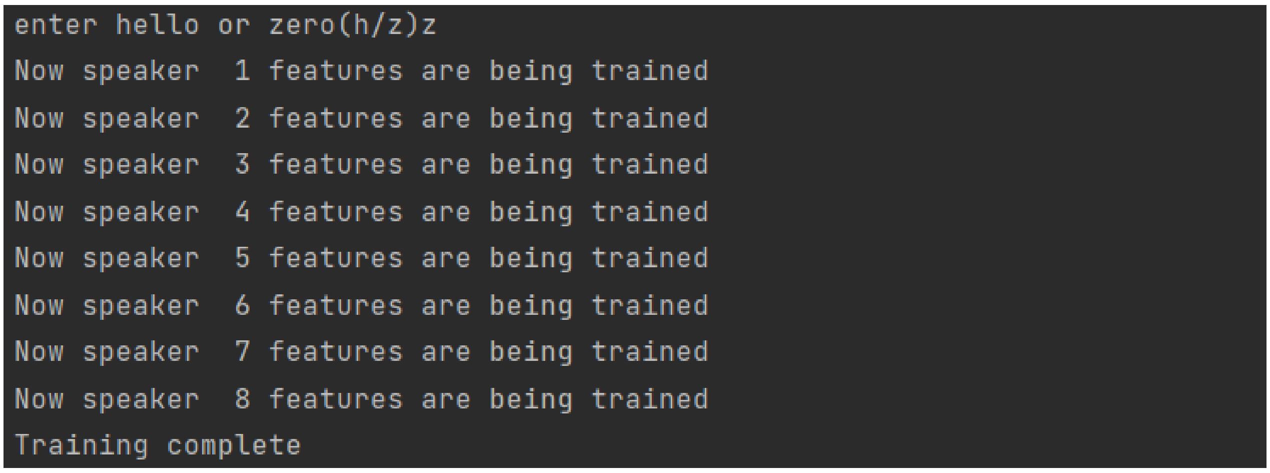


```
do u wanna change word? n
enter choice:
1.show percentage(p)
2.identify speaker(s)
2
enter wav file: /sv.wav
Now speaker  1 features are being tested
Now speaker  2 features are being tested
Now speaker  3 features are being tested
Now speaker  4 features are being tested
Now speaker  5 features are being tested
Now speaker  6 features are being tested
Now speaker  7 features are being tested
Identified Speaker: senthil
do u wanna change word?
```

figure 8. The unknown audio file is entered and the speaker is identified.

4.5 TRAINING DATASET FOR WORD “ZERO”

Users can enter whether they want to use the word “hello” or “zero”. Depending on the user input(“zero”), the dataset present in the train folder gets accessed and is trained.



```
enter hello or zero(h/z)z
Now speaker  1 features are being trained
Now speaker  2 features are being trained
Now speaker  3 features are being trained
Now speaker  4 features are being trained
Now speaker  5 features are being trained
Now speaker  6 features are being trained
Now speaker  7 features are being trained
Now speaker  8 features are being trained
Training complete
```

figure 9. Speakers are trained .

4.6 MFCC VS NUMBER OF FEATURES (CODEBOOK)

The codebooks for the trained dataset of zero are displayed.

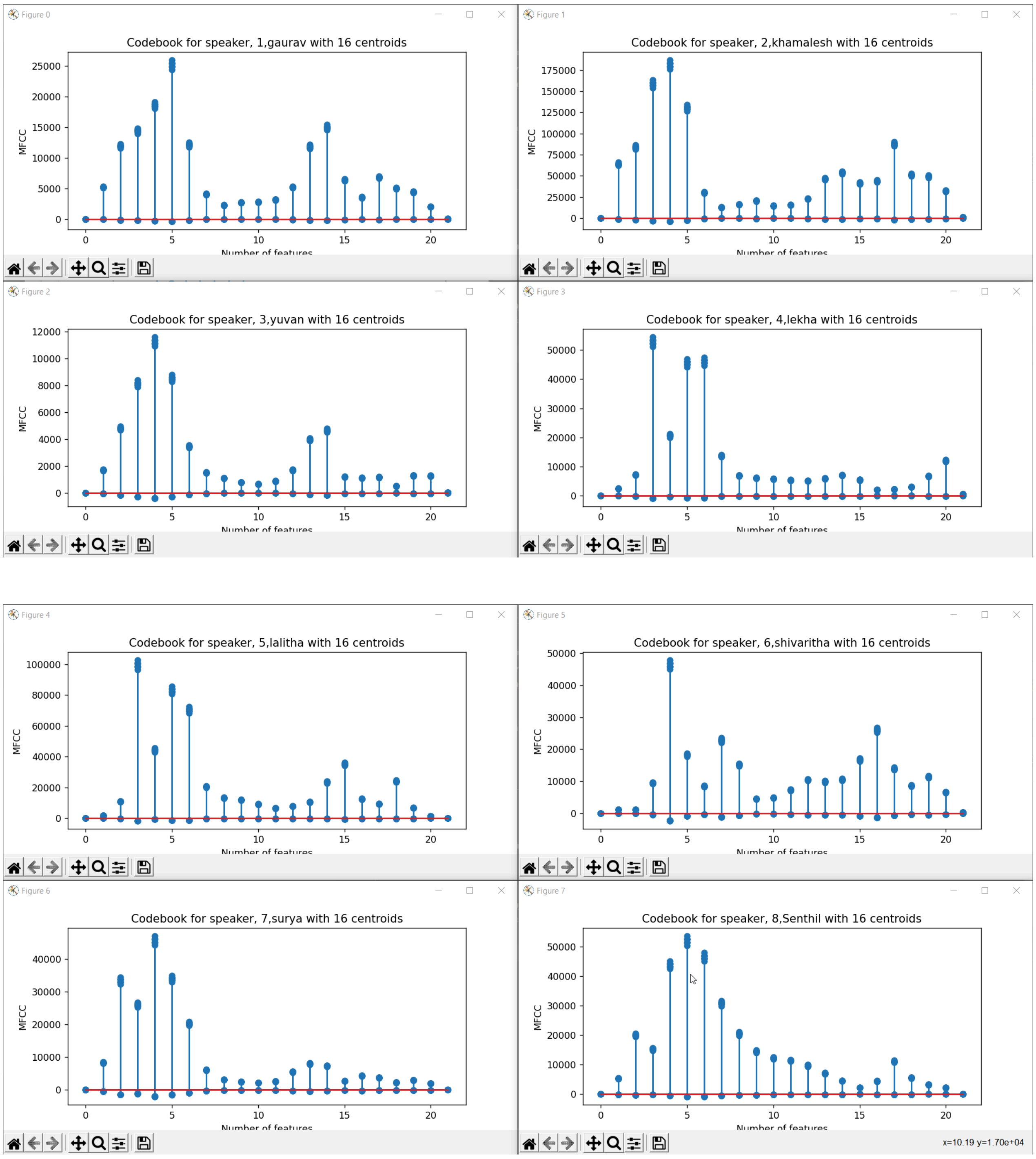
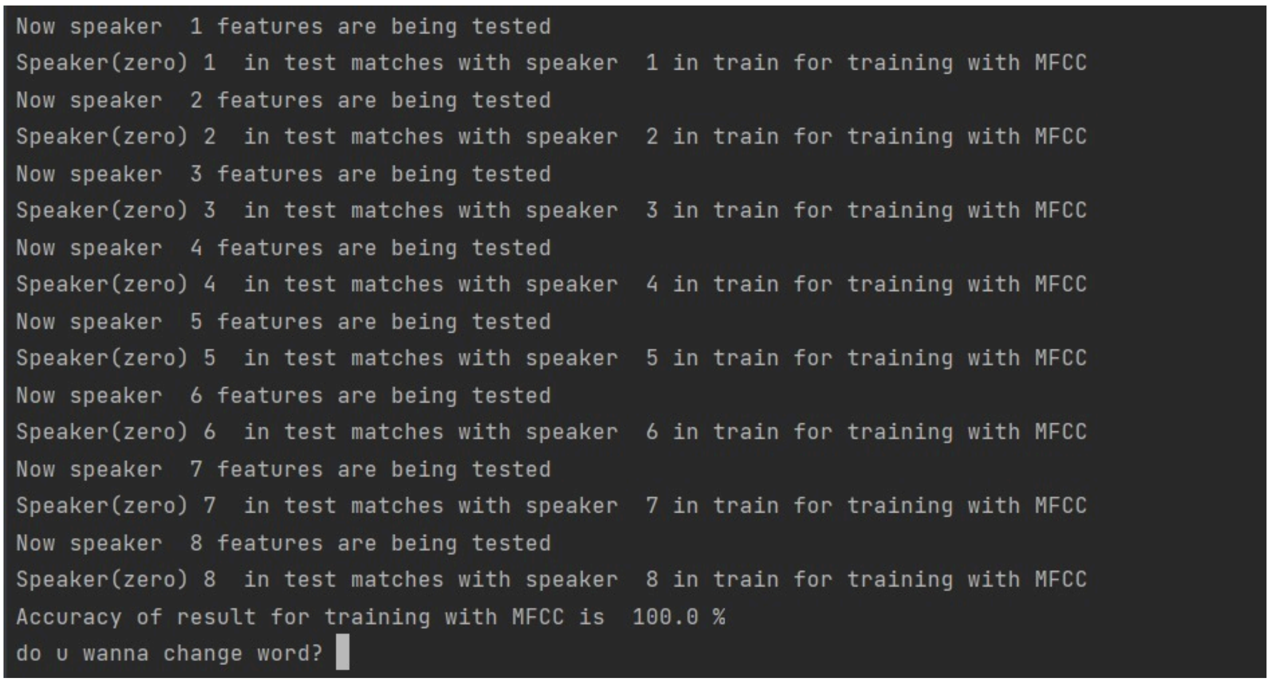


figure 10. Codebook for each trained speech.

4.7 ACCURACY OF SPEAKER RECOGNITION

Users can either choose to display the accuracy or to recognise the speaker using an audio file.
The test data is matched with train data.

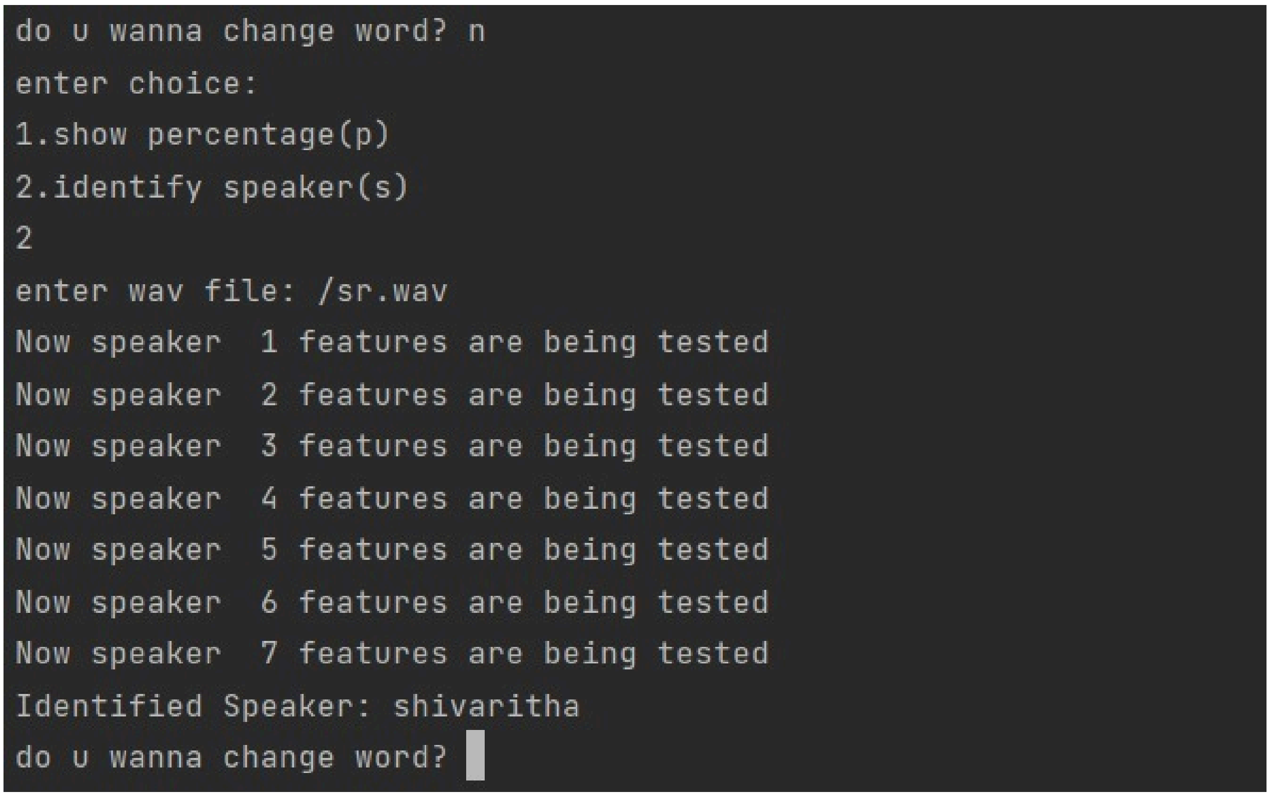


```
Now speaker  1 features are being tested
Speaker(zero) 1  in test matches with speaker  1 in train for training with MFCC
Now speaker  2 features are being tested
Speaker(zero) 2  in test matches with speaker  2 in train for training with MFCC
Now speaker  3 features are being tested
Speaker(zero) 3  in test matches with speaker  3 in train for training with MFCC
Now speaker  4 features are being tested
Speaker(zero) 4  in test matches with speaker  4 in train for training with MFCC
Now speaker  5 features are being tested
Speaker(zero) 5  in test matches with speaker  5 in train for training with MFCC
Now speaker  6 features are being tested
Speaker(zero) 6  in test matches with speaker  6 in train for training with MFCC
Now speaker  7 features are being tested
Speaker(zero) 7  in test matches with speaker  7 in train for training with MFCC
Now speaker  8 features are being tested
Speaker(zero) 8  in test matches with speaker  8 in train for training with MFCC
Accuracy of result for training with MFCC is  100.0 %
do u wanna change word?
```

figure 11. Test speech is matched with their respective trained speakers.

4.8 SPEAKER RECOGNITION

Users can either change the word or continue.
Now the audio file is given as input and the speaker is identified.



```
do u wanna change word? n
enter choice:
1.show percentage(p)
2.identify speaker(s)
2
enter wav file: /sr.wav
Now speaker  1 features are being tested
Now speaker  2 features are being tested
Now speaker  3 features are being tested
Now speaker  4 features are being tested
Now speaker  5 features are being tested
Now speaker  6 features are being tested
Now speaker  7 features are being tested
Identified Speaker: shivaritha
do u wanna change word?
```

figure 12. Speaker is identified.

5. CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

The speaker recognition system was successfully designed and appears to recognize each speaker with high accuracy. This application comes very handy for many voice recognition technologies that use voice as the medium of input. Speaker recognition can play a key role in authentication, surveillance and forensic. From this system it is observed that the audio file of each speaker is provided as a training data set. These files are sampled at the same rate. The system consists of 2 modules: (i) Feature extraction; (ii) Feature Matching. In feature extraction, the mfcc characteristics of each speaker is extracted and plotted as codebooks using LBG, a vector quantization algorithm. In feature matching, the characteristics for the unknown speaker are extracted and compared with the known speakers through the created codebooks. And hence, the speaker is identified. Speaker recognition has a wide scope in the field of security systems. Speaker recognition

5.2 FUTURE WORK

In the future we hope to improve the following:

1. Accuracy of prediction is increased.
2. Identifying even in presence of noise.
3. System to comprehend a larger data set with different audio file properties like sampling rate, bit rate.

We can also implement miscellaneous additions such as testing live audio files and comparing with the data set to find the speaker.

6.REFERENCES

1. http://svr-www.eng.cam.ac.uk/comp.speech/Section6/Q6.6.html
2. https://www.researchgate.net/publication/316768145_Voice_Recognition_Based_on_Vector_Quantization_Using_LBG
3. http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/
4. https://vdocument.in/design-of-an-automatic-speaker-recognition-system-using-mfcc-.html