25/04/2020

Speech Recognition and Voice Detection

A Project Report

DEPARTMENT OF COMPUTER ENGINERING
FACULTY OF TECHNOLOGY
DHARMSINH DESAI UNIVERSITY, NADIAD

A Project Report

On

Speech Recognition and Voice Detection

B. Tech (CE) Semester-VI

In fulfilment of all requirements for the subject of
System Design Practice (SDP)

Bachelor of Technology
In
Computer Engineering

Submitted by

Parth A. Patel (CE - 093)
(17CEUBS099)

Yug D. Rajani (CE - 109)
(17ITUOS059)

Darsh N. Shah (CE - 119)
(17CEUOS091)

Under the Guidance of
Prof. Hariom A. Pandya

DEPARTMENT OF COMPUTER ENGINEERING
FACULTY OF TECHNOLOGY,
DHARMSINH DESAI UNIVERSITY
COLLEGE ROAD, NADIAD- 387001

DHARMSINH DESAI UNIVERSITY

NADIAD-387001, GUJARAT

CERTIFICATE

This is to certify that the project carried out in the subject of <u>System Design Practice</u> titled "Speech Recognition and Voice Detection" and recorded in this report is the bona fide work of

Parth A. Patel
ROLL NO: CE - 093
ID: 17CEUBS099

Yug D. Rajani
ROLL NO: CE-109
ID: 17ITUOS059

Darsh N. Shah
ROLL NO: CE-119
ID: 17CEUOS091

Department of Computer Engineering, semester VI. They were involved in Project Development during the period December-2019 to April-2020.

Prof. H A. Pandya
Project Guide,
Department of Computer Engineering
Faculty of Technology
Dharmsinh Desai University, Nadiad

Dr. C. K. Bhensdadia
Head,
Department of Computer Engineering
Faculty of Technology
Dharmsinh Desai University, Nadiad

Table of Contents

Acknowledgement

We have worked really hard to bring this project to completion, however it would not have been possible without the kind support and help of many individuals. We would like to extend sincere thanks to all of them.

We take this opportunity to express our profound gratitude and deep regards to my guide **Prof. Hariom A. Pandya** (Assistant Professor, CE Dept., DDIT) for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

We also take this opportunity to express a deep sense of gratitude to **Dr. C. K. Bhensdadia** (Head of Department, CE Dept., DDIT) for their cordial support, valuable information and guidance which helped me in completing this task through various stages.

Abstract

The era of digital technology opens the windows to frequent use of **Web Applications** for small tasks. The use of pen and paper is decreasing and the Web Applications are emerging. It's the time to change from conventional ways to websites, which has become the part of our daily routine. Furthermore, **Machine Learning**, which is an application of artificial intelligence, is an emerging branch in the subject of Computer Science. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly. We hereby introduce **Speech Recognition and Voice Detection,** a Web Application which helps recognise the voice of a speaker using the knowledge gained by it during the previous interactions with the user. Directly or indirectly, you are always in contact with audio. Your brain is continuously processing and understanding audio data and giving you information about the environment. Even when you think you are in a quiet environment, you tend to catch much more subtle sounds, like the rustling of leaves or the splatter of rain. This is the extent of your connection with audio. The application uses various features of the voice of a human to be trained and to learn the patterns in his dialect. In real world, this application and its concept can be utilised in order to solve various security issues and for investigation purpose by the various departments of the world.

CHAPTER 1
INTRODUCTION

Project Details in Brief:

The application “Speech Recognition and Voice Detection” is a research-based project which helps users identify the speaker from a known list of speakers. A user can use the application, either for detecting the voice from the known speakers or by adding a new person’s voice to be used for detection in future. Conclusions will be derived from various tests conducted using the different set of features of the voice used for storing and detecting the voice by the application.

Technology Used:

- Front End
  - HTML
  - CSS
  - Bootstrap
  - Javascript

- Back End
  - TensorFlow (Keras)
  - Python (Django Framework)

- Diagram Tool
  - UMLet

Basic Terminology

- **Layer:** A layer is a container that usually receives weighted input, transforms it with a set of mostly non-linear functions and then passes these values as output to the next layer. A layer is usually uniform, that is it only contains one type of activation function, pooling, convolution etc.

- **Neural Network:** A neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

- **Neurons:** A "neuron" in an artificial neural network is a mathematical approximation of a biological neuron. It takes a vector of inputs, performs a transformation on them, and outputs a single scalar value. It can be thought of as a filter. Typically we use nonlinear filters in neural networks.

- **Epoch:** An epoch is a term used in machine learning and indicates the number of passes through the entire training dataset the machine learning algorithm has completed.

- **Dropout:** Simply put, dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. By "ignoring", it means these units are not considered during a particular forward or backward pass.

- **Optimizer:** Optimizers shape and mould your model into its most accurate possible form by futzing with the weights.

- **Accuracy:** Accuracy is the number of correctly predicted data points out of all the data points. More formally, it is defined as the number of true positives and true negatives divided by the number of true positives, true negatives, false positives, and false negatives.

CHAPTER 2 SOFTWARE REQUIREMENTS SPECIFICATION

2.1 Scope

The scope of the system is very large as it is applicable to any security measures trying to detect voice and embracing automation.

2.2 Major Functional Requirements

R1: Add file to the server

Description: User can add a file to the server to predict the speaker.

Input: User selects desired Audio file and then clicks on “Add File” button.

Output: " File Successfully Added” message.

R2: Delete file from the server

Description: User is allowed to delete file from the server when not needed.

Input: User selection of the file to be deleted and then clicks on “Delete” button.

Output: "File Successfully Deleted" message.

R3: Play Audio File

Description: User can play the audio file uploaded on server.

Input: User selection.

Output: Audio output of the selected file.

R4: Make prediction

Description: User can predict the name of the speaker by giving the desired input file.

Input: User selection.

Output: Name of the speaker predicted by the model.

2.3 Non-Functional Requirements

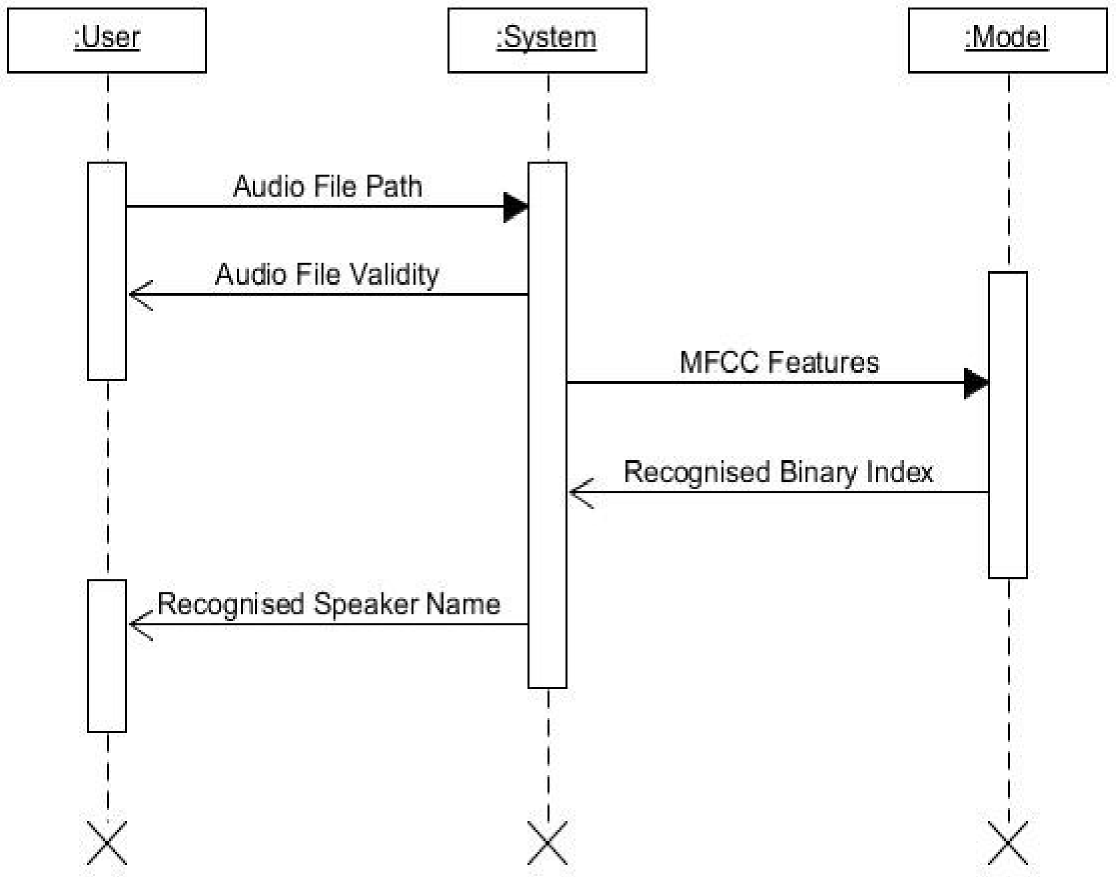
1. **Performance**
   The system must be fast and accurate as the main aim is automation and efficiency, it must provide accurate results in less time and using less energy.

2. **Safety**
   As all the audio data of the people would be stored in the system, safety is one of the main concerns of the system and hence only authorised users should be allowed to access the dataset of the system.

3. **Reliability**
   The data of any audio file(s) should not be leaked and must be in safe hands so the system must be reliable.

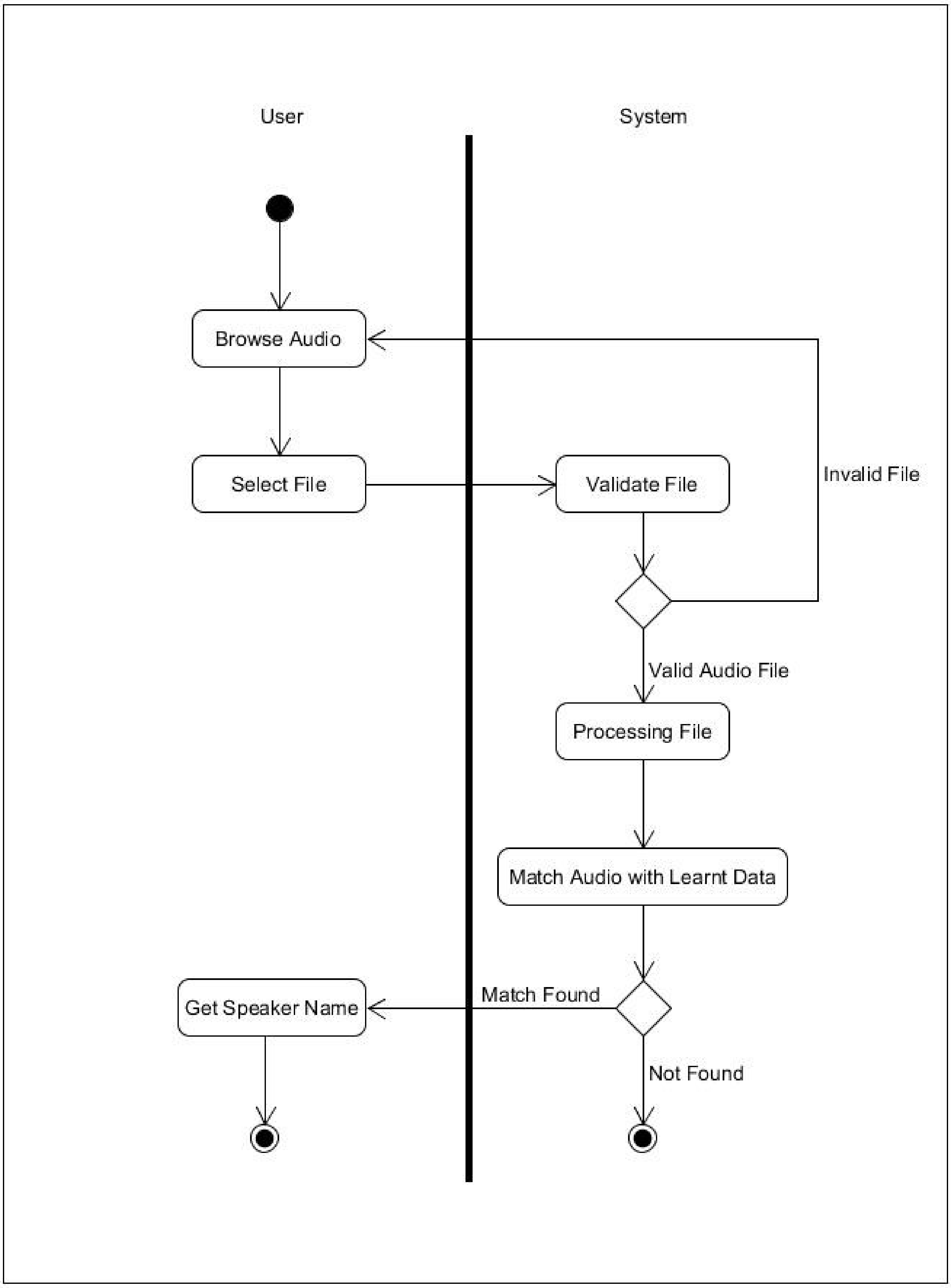
CHAPTER 3 DESIGN

3.1 Sequence Diagram



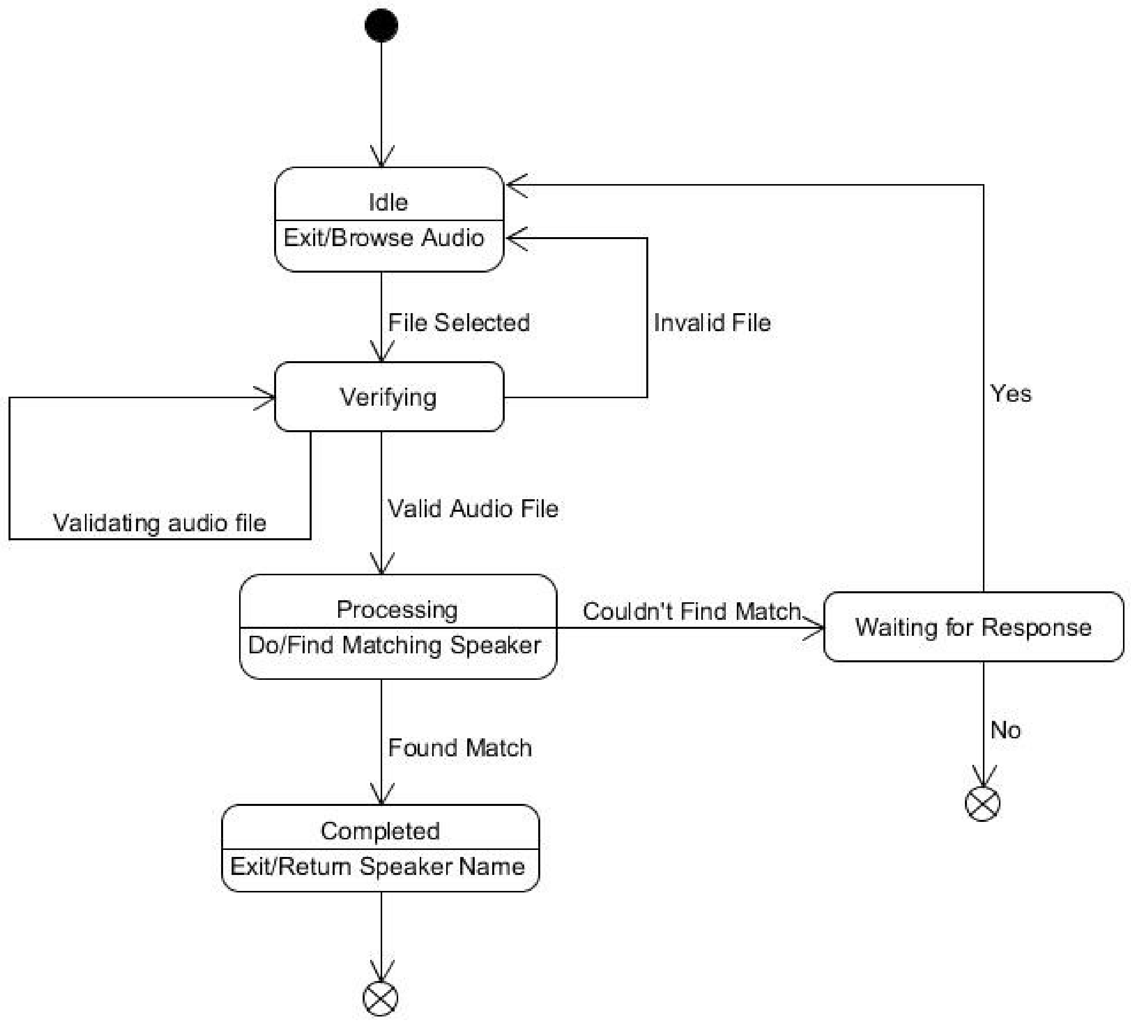
Sequence Diagram for Recognising Input Audio File Speaker

3.2 Activity Diagram



Activity Diagram for Recognising Input Audio File Speaker

3.3 State Diagram



State Diagram for Recognising Input Audio File Speaker

CHAPTER 4 IMPLEMENTATION DETAILS

4.1 Dataset Details

Link: http://www.openslr.org/resources/12/dev-clean.tar.gz

Number of Speakers: 70

Number of Recordings: 5653

Number of Files used for Training: 3636 (65 %)

Number of Files used for Validation: 575 (10 %)

Number of Files used for Testing: 1442 (25 %)

Accuracy observed with different versions of the model:

Version	Layers	Neurons	Epoch	Dropout	Optimizer	Accuracy
1	3	256, 256, 40	10	0.5 (3 layers)	adam	7 – 10 %
2	4	40, 25, 10, 40	1000	-	RMSProp	90 – 100 %
3	4	40, 25, 10, 40	1000	0.5 (3 layers)	adam	30 – 40 %
4	4	40, 25, 10, 40	1000	0.5 (2 layers)	adam	60 – 70 %
5	4	40, 25, 10, 40	1000	-	adam	90 – 100 %

Version 5 of the model is being currently used.

4.2 Various Functions Used

- **Loss Function:** The loss function is the guide to the terrain, telling the optimizer when it's moving in the right or wrong direction.

- **'adam':** Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. Its name is derived from adaptive moment estimation, and the reason it's called that is because Adam uses estimations of first and second moments of gradient to adapt the learning rate for each weight of the neural network.

- **'RMSProp':** The RMSprop optimizer is similar to the gradient descent algorithm with momentum. The RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster.

- **Moment:** Nth moment of a random variable is defined as the expected value of that variable to the power of n. More formally:

  $m_n = E[X^n]$; m — moment, X – random variable

- **Categorical Cross-entropy:** Categorical cross-entropy is a loss function that is used for single label categorization. This is when only one category is applicable for each data point. In other words, an example can belong to one class only.

- **matplotlib:** Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

4.3 Mel Frequency Cepstral Coefficients (MFCCs)

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. In MIR, it is often used to describe timbre.

In Sound Processing, the **Mel-Frequency Cepstrum** (**MFC**) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-Frequency Cepstral Coefficients (**MFCCs**) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum").

The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound, for example, in audio compression.

MFCCs are commonly derived as follows:

1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

There can be variations on this process, for example: differences in the shape or spacing of the windows used to map the scale, or addition of dynamics features such as "delta" and "delta-delta" (first- and second-order frame-to-frame difference) coefficients.

MFCCs are commonly used as features in speech recognition systems, such as the systems which can automatically recognize numbers spoken into a telephone.

MFCCs are also increasingly finding uses in music information retrieval applications such as genre classification, audio similarity measures, etc.

We extract 40-dimensional features from speech frames. There are 20 MFCC features and 20 derivatives of MFCC features. The derivatives of MFCCs provides the information of dynamics of MFCCs over the time.

It turns out that calculating the delta-MFCC and appending them to the original MFCC features (20-dimenaional) increases the performance in lot of speech analytics applications. To calculate delta features from MFCCs, the following equation is applied:

$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+n} - c_{t-n})}{2\sum_{n=1}^{N} n^2}$$

; Where ‘N’ is number of deltas summed over. Typically taken as 2.

4.4 The Central Idea

As with all unstructured data formats, audio data has a couple of pre-processing steps which have to be followed before it is presented for analysis.

The first step is to actually load the data into a machine understandable format. For this, we simply take values after every specific time steps. For example; in a 2 second audio file, we extract values at half a second. This is called **sampling of audio data,** and the rate at which it is sampled is called the **sampling rate.**

Another way of representing audio data is by converting it into a different domain of data representation, namely the frequency domain. When we sample an audio data, we require much more data points to represent the whole data and also, the sampling rate should be as high as possible.

On the other hand, if we represent audio data in **frequency domain**, much less computational space is required. To get an intuition, take a look at the image below:

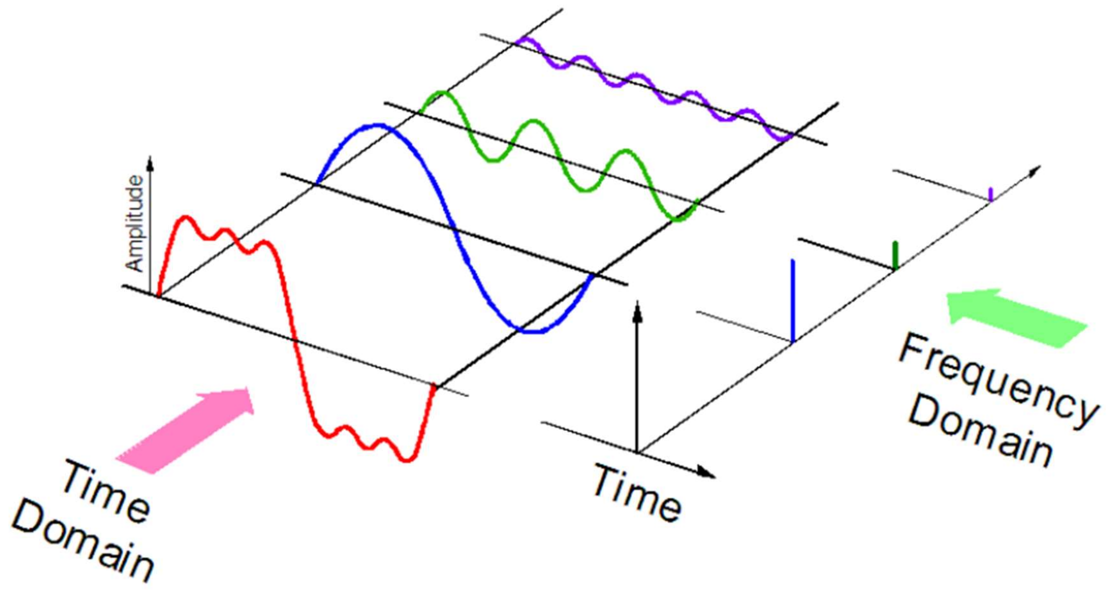


Fig. Representation of Audio in Frequency Domain

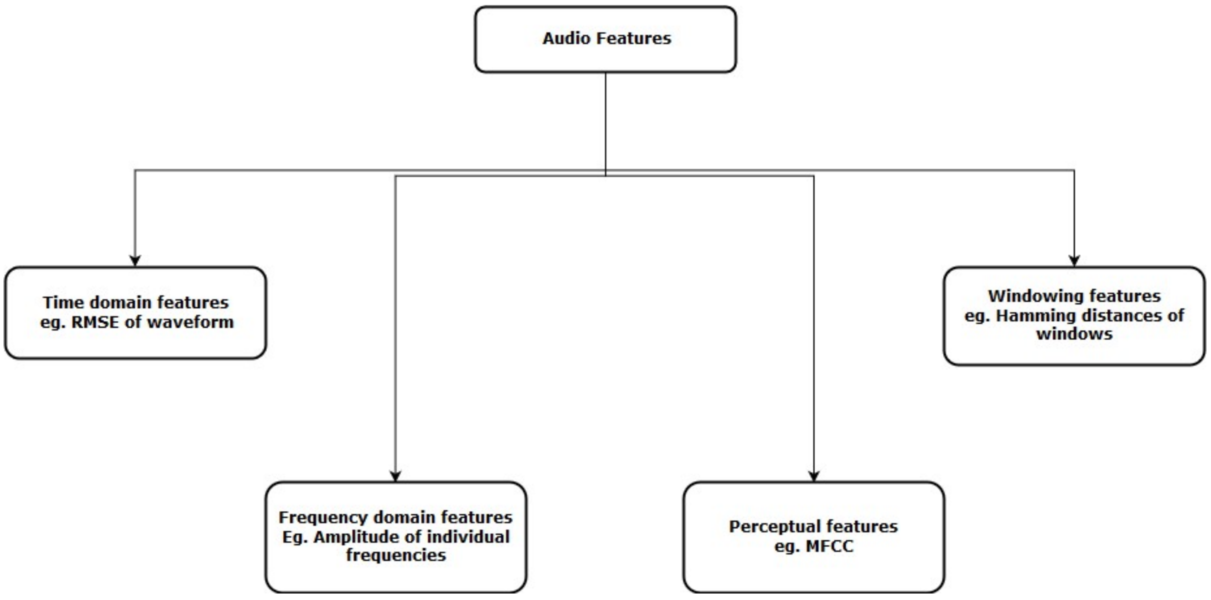

Audio Features
Time domain features eg. RMSE of waveform
Frequency domain features Eg. Amplitude of individual frequencies
Perceptual features eg. MFCC
Windowing features eg. Hamming distances of windows

Fig. Classification of Audio Features

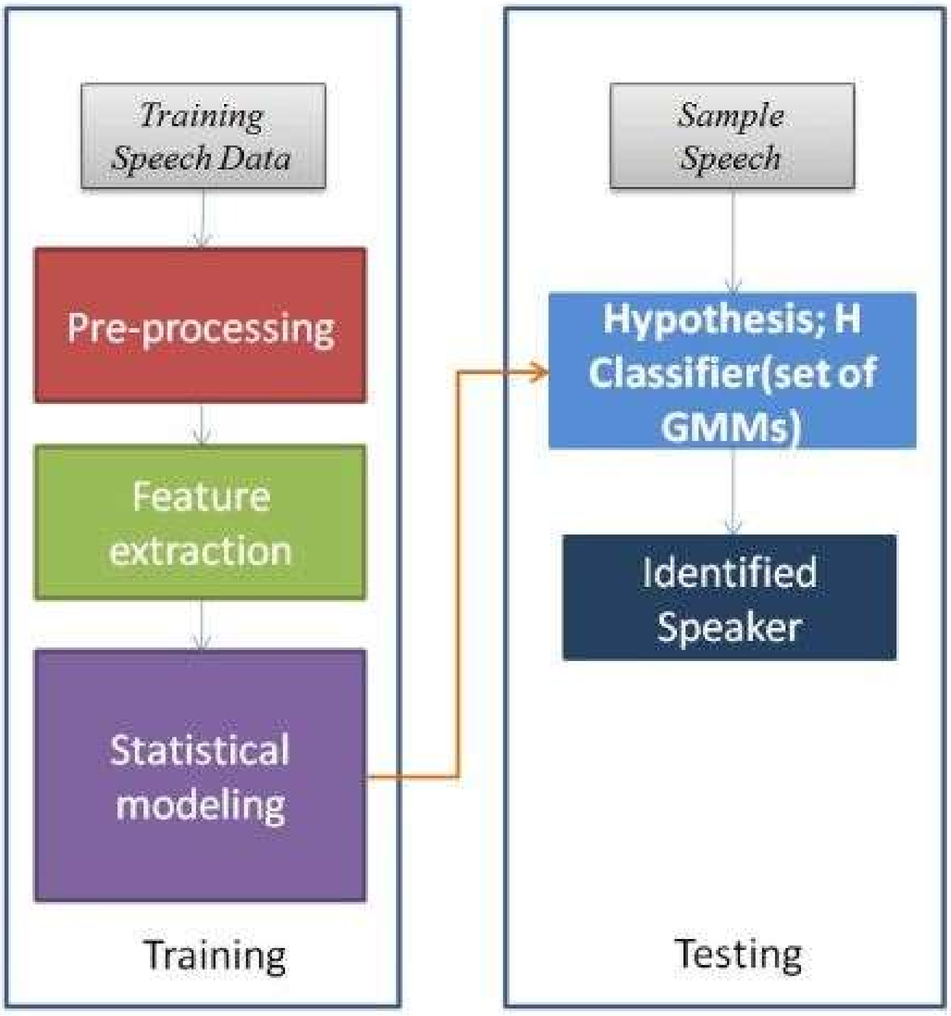
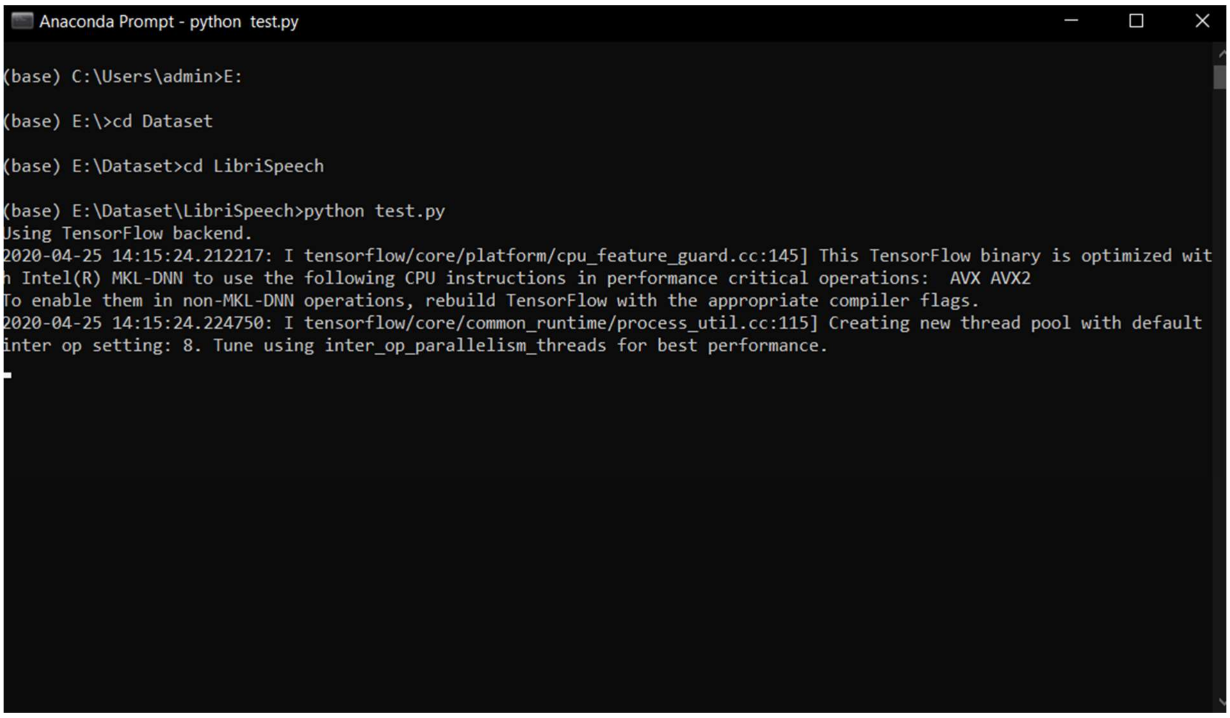


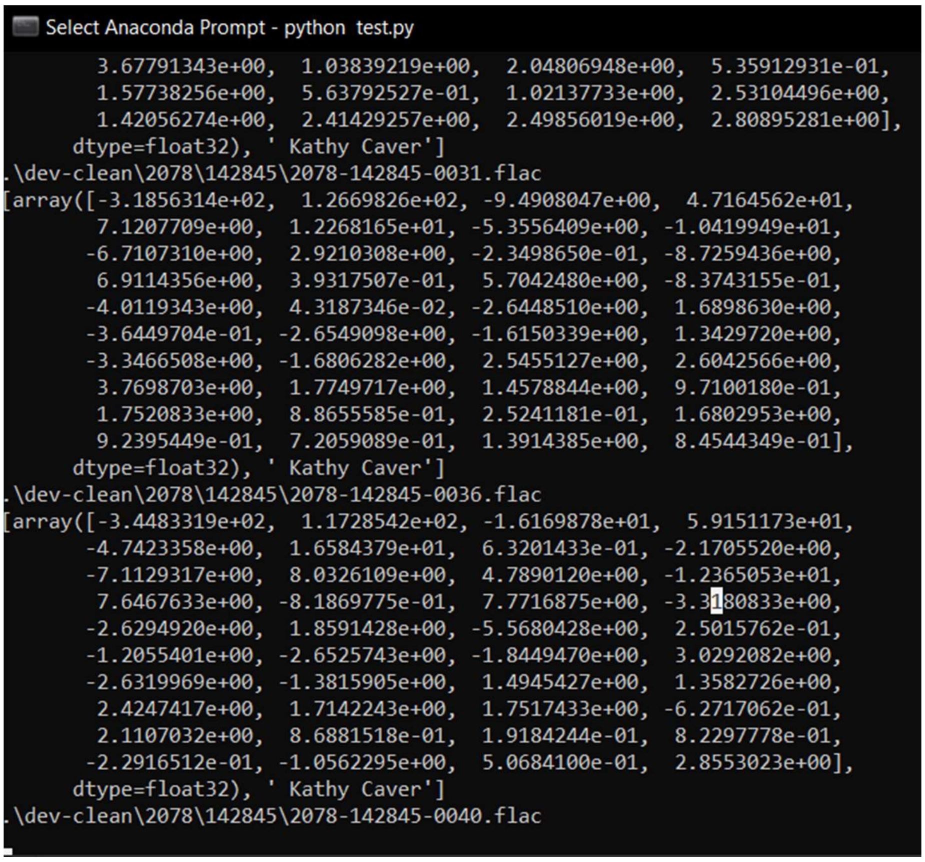
Fig. The Internal Working of Speech Recognition

CHAPTER 5 TESTING



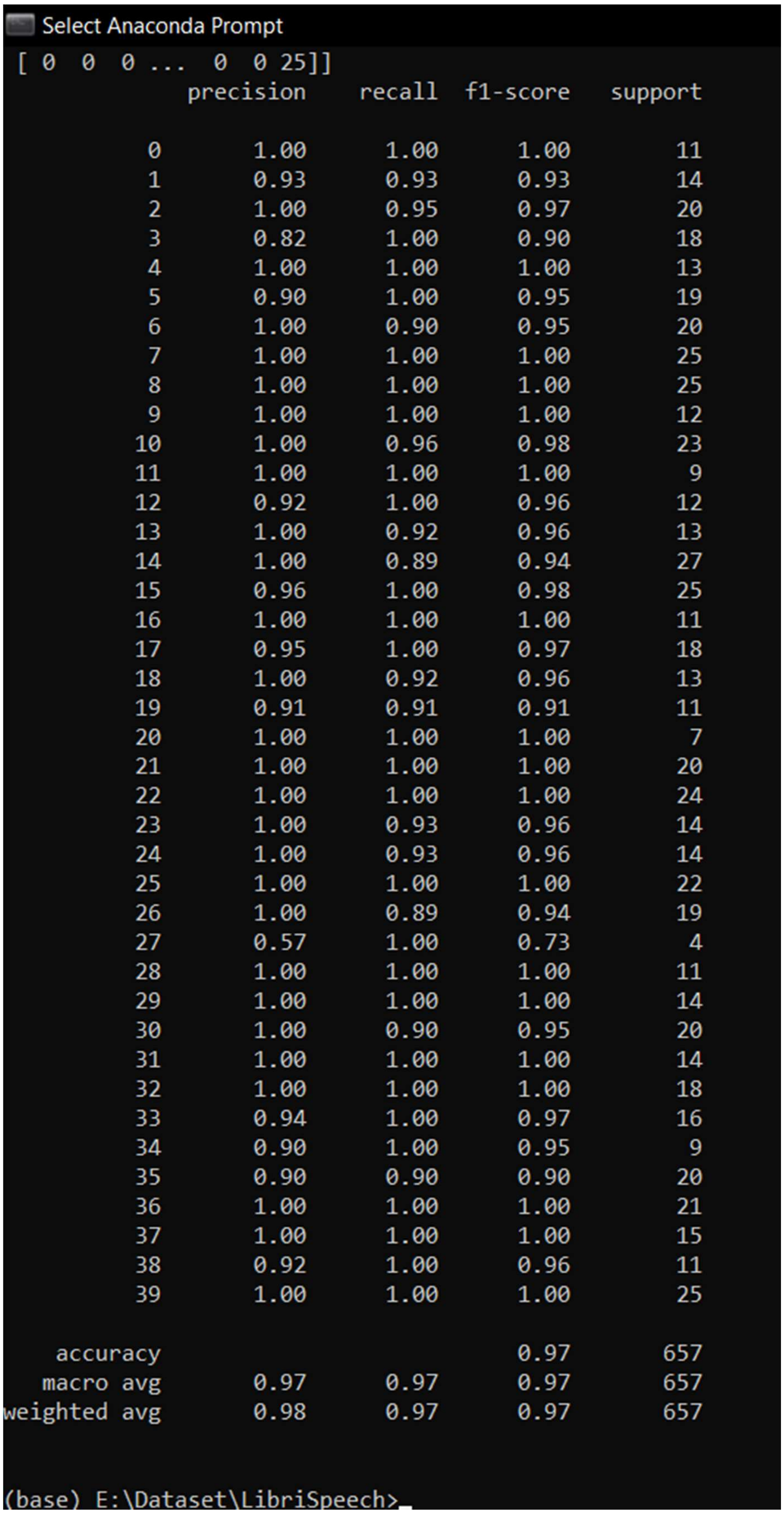
```
(base) C:\Users\admin>E:

(base) E:\>cd Dataset

(base) E:\Dataset>cd LibriSpeech

(base) E:\Dataset\LibriSpeech>python test.py
Using TensorFlow backend.
2020-04-25 14:15:24.212217: I tensorflow/core/platform/cpu_feature_guard.cc:145] This TensorFlow binary is optimized wit
h Intel(R) MKL-DNN to use the following CPU instructions in performance critical operations:  AVX AVX2
To enable them in non-MKL-DNN operations, rebuild TensorFlow with the appropriate compiler flags.
2020-04-25 14:15:24.224750: I tensorflow/core/common_runtime/process_util.cc:115] Creating new thread pool with default
inter op setting: 8. Tune using inter_op_parallelism_threads for best performance.
```

Testing 1: Firing the command for Testing



```
        3.67791343e+00,  1.03839219e+00,  2.04806948e+00,  5.35912931e-01,
        1.57738256e+00,  5.63792527e-01,  1.02137733e+00,  2.53104496e+00,
        1.42056274e+00,  2.41429257e+00,  2.49856019e+00,  2.80895281e+00],
      dtype=float32), ' Kathy Caver']
.\dev-clean\2078\142845\2078-142845-0031.flac
[array([-3.1856314e+02,  1.2669826e+02, -9.4908047e+00,  4.7164562e+01,
        7.1207709e+00,  1.2268165e+01, -5.3556409e+00, -1.0419949e+01,
       -6.7107310e+00,  2.9210308e+00, -2.3498650e-01, -8.7259436e+00,
        6.9114356e+00,  3.9317507e-01,  5.7042480e+00, -8.3743155e-01,
       -4.0119343e+00,  4.3187346e-02, -2.6448510e+00,  1.6898630e+00,
       -3.6449704e-01, -2.6549098e+00, -1.6150339e+00,  1.3429720e+00,
       -3.3466508e+00, -1.6806282e+00,  2.5455127e+00,  2.6042566e+00,
        3.7698703e+00,  1.7749717e+00,  1.4578844e+00,  9.7100180e-01,
        1.7520833e+00,  8.8655585e-01,  2.5241181e-01,  1.6802953e+00,
        9.2395449e-01,  7.2059089e-01,  1.3914385e+00,  8.4544349e-01],
      dtype=float32), ' Kathy Caver']
.\dev-clean\2078\142845\2078-142845-0036.flac
[array([-3.4483319e+02,  1.1728542e+02, -1.6169878e+01,  5.9151173e+01,
       -4.7423358e+00,  1.6584379e+01,  6.3201433e-01, -2.1705520e+00,
       -7.1129317e+00,  8.0326109e+00,  4.7890120e+00, -1.2365053e+01,
        7.6467633e+00, -8.1869775e-01,  7.7716875e+00, -3.3180833e+00,
       -2.6294920e+00,  1.8591428e+00, -5.5680428e+00,  2.5015762e-01,
       -1.2055401e+00, -2.6525743e+00, -1.8449470e+00,  3.0292082e+00,
       -2.6319969e+00, -1.3815905e+00,  1.4945427e+00,  1.3582726e+00,
        2.4247417e+00,  1.7142243e+00,  1.7517433e+00, -6.2717062e-01,
        2.1107032e+00,  8.6881518e-01,  1.9184244e-01,  8.2297778e-01,
       -2.2916512e-01, -1.0562295e+00,  5.0684100e-01,  2.8553023e+00],
      dtype=float32), ' Kathy Caver']
.\dev-clean\2078\142845\2078-142845-0040.flac
```

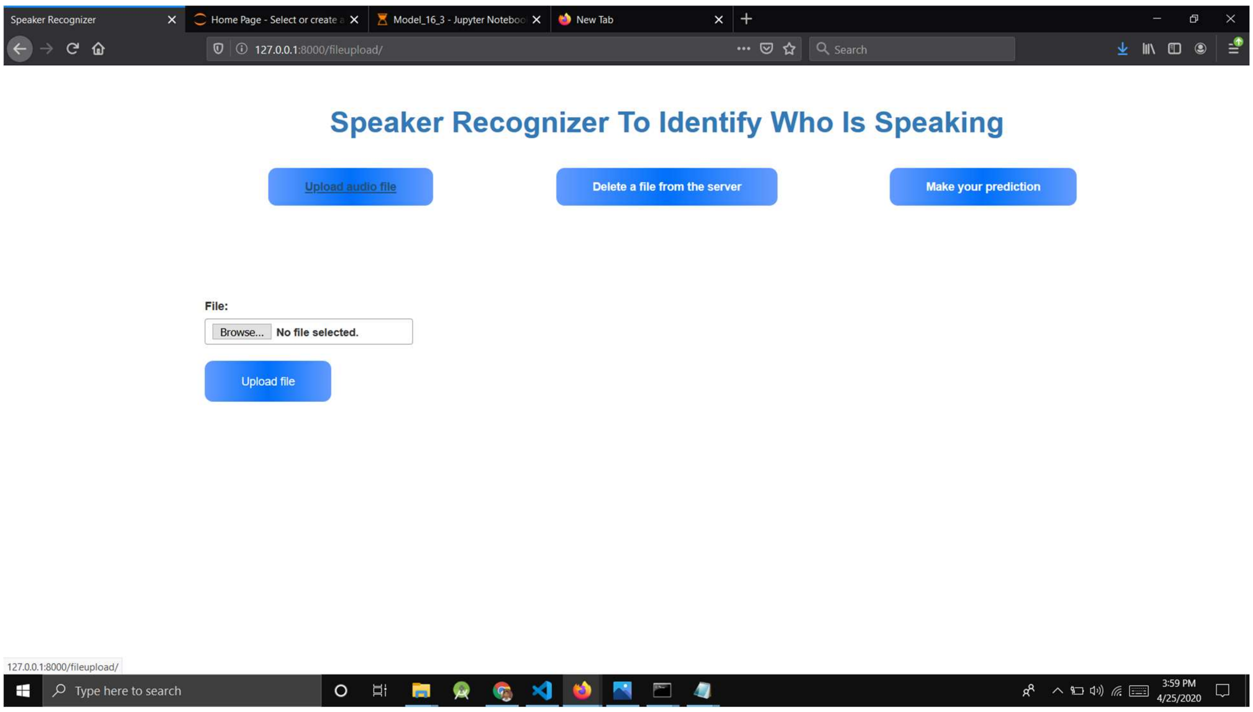
Testing 2: Running Testing



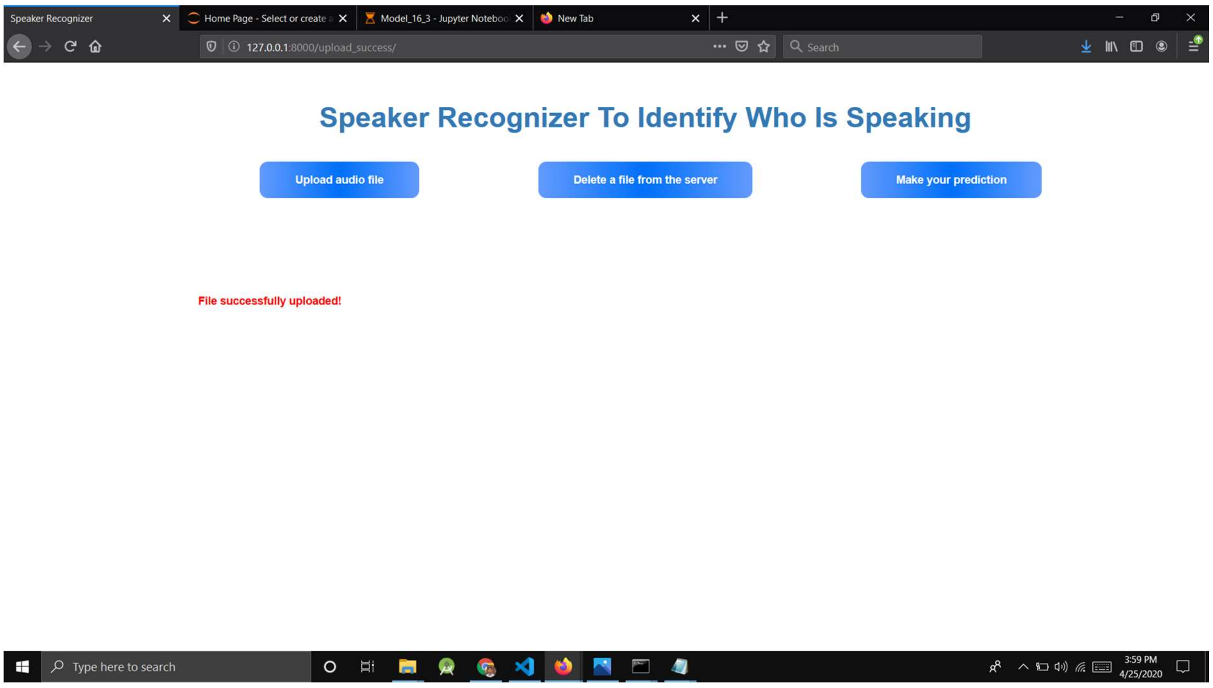
```
[ 0  0  0 ...  0  0 25]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       0.93      0.93      0.93        14
           2       1.00      0.95      0.97        20
           3       0.82      1.00      0.90        18
           4       1.00      1.00      1.00        13
           5       0.90      1.00      0.95        19
           6       1.00      0.90      0.95        20
           7       1.00      1.00      1.00        25
           8       1.00      1.00      1.00        25
           9       1.00      1.00      1.00        12
          10       1.00      0.96      0.98        23
          11       1.00      1.00      1.00         9
          12       0.92      1.00      0.96        12
          13       1.00      0.92      0.96        13
          14       1.00      0.89      0.94        27
          15       0.96      1.00      0.98        25
          16       1.00      1.00      1.00        11
          17       0.95      1.00      0.97        18
          18       1.00      0.92      0.96        13
          19       0.91      0.91      0.91        11
          20       1.00      1.00      1.00         7
          21       1.00      1.00      1.00        20
          22       1.00      1.00      1.00        24
          23       1.00      0.93      0.96        14
          24       1.00      0.93      0.96        14
          25       1.00      1.00      1.00        22
          26       1.00      0.89      0.94        19
          27       0.57      1.00      0.73         4
          28       1.00      1.00      1.00        11
          29       1.00      1.00      1.00        14
          30       1.00      0.90      0.95        20
          31       1.00      1.00      1.00        14
          32       1.00      1.00      1.00        18
          33       0.94      1.00      0.97        16
          34       0.90      1.00      0.95         9
          35       0.90      0.90      0.90        20
          36       1.00      1.00      1.00        21
          37       1.00      1.00      1.00        15
          38       0.92      1.00      0.96        11
          39       1.00      1.00      1.00        25

    accuracy                           0.97       657
   macro avg       0.97      0.97      0.97       657
weighted avg       0.98      0.97      0.97       657


(base) E:\Dataset\LibriSpeech>_
```

Testing 3: Test Result

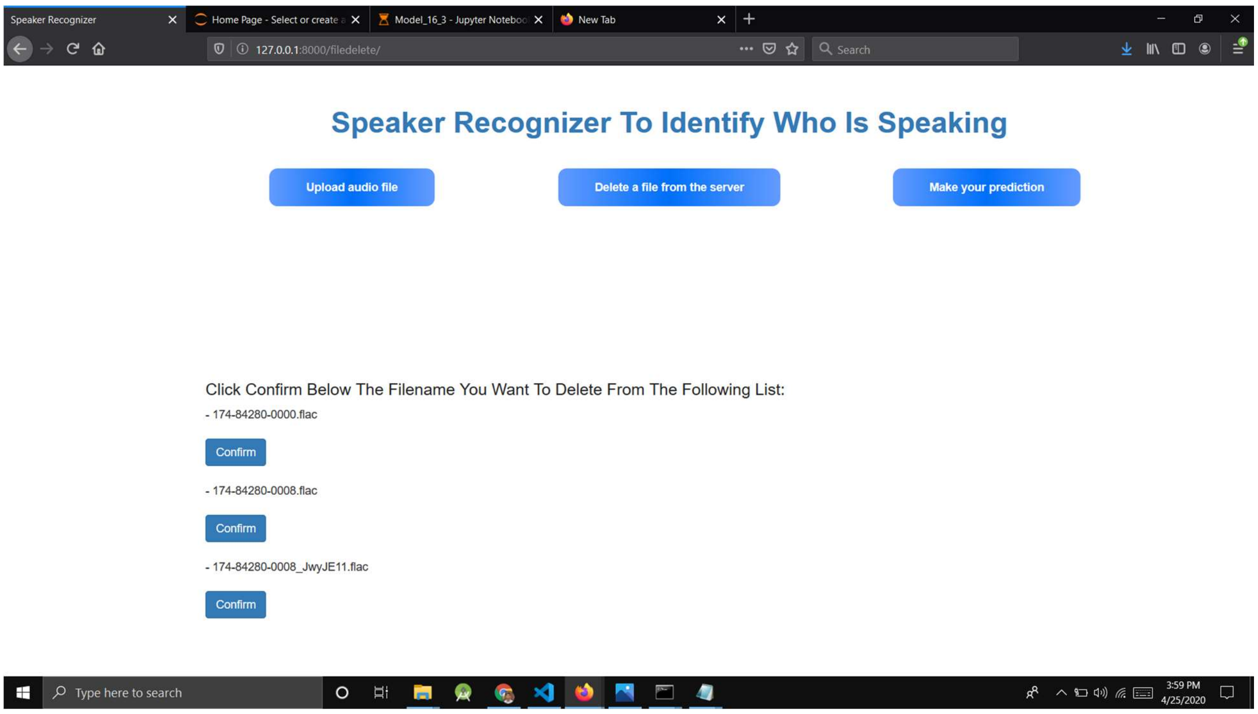
CHAPTER 6
SCREENSHOTS


Speaker Recognizer
Home Page - Select or create a
Model_16_3 - Jupyter Notebook
New Tab
127.0.0.1:8000/fileupload/
Search
Speaker Recognizer To Identify Who Is Speaking
Upload audio file
Delete a file from the server
Make your prediction
File:
Browse... No file selected.
Upload file
127.0.0.1:8000/fileupload/
Type here to search
3:59 PM
4/25/2020

Screenshot 1: Home Page of Web Application


Speaker Recognizer
Home Page - Select or create a
Model_16_3 - Jupyter Notebook
New Tab
127.0.0.1:8000/upload_success/
Search
Speaker Recognizer To Identify Who Is Speaking
Upload audio file
Delete a file from the server
Make your prediction
File successfully uploaded!
Type here to search
3:59 PM
4/25/2020

Screenshot 2: Uploading a file to server


Speaker Recognizer
Home Page - Select or create a
Model_16_3 - Jupyter Notebook
New Tab
127.0.0.1:8000/filedelete/
Search
Upload audio file
Delete a file from the server
Make your prediction
Type here to search
3:59 PM
4/25/2020

Speaker Recognizer To Identify Who Is Speaking

Click Confirm Below The Filename You Want To Delete From The Following List:

- 174-84280-0000.flac

Confirm

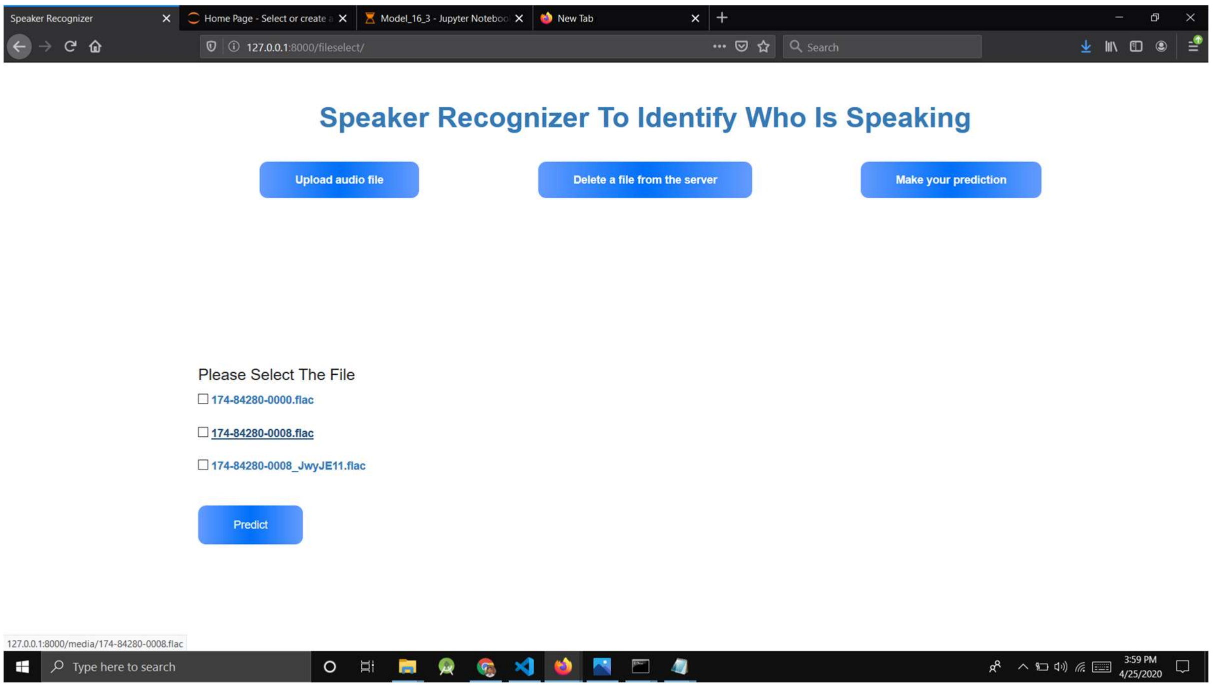
- 174-84280-0008.flac

Confirm

- 174-84280-0008_JwyJE11.flac

Confirm

Screenshot 3: Selection of audio file for deletion from server


Speaker Recognizer
Home Page - Select or create a
Model_16_3 - Jupyter Notebook
New Tab
127.0.0.1:8000/fileselect/
Search
Upload audio file
Delete a file from the server
Make your prediction
127.0.0.1:8000/media/174-84280-0008.flac
Type here to search
3:59 PM
4/25/2020

Speaker Recognizer To Identify Who Is Speaking

Please Select The File

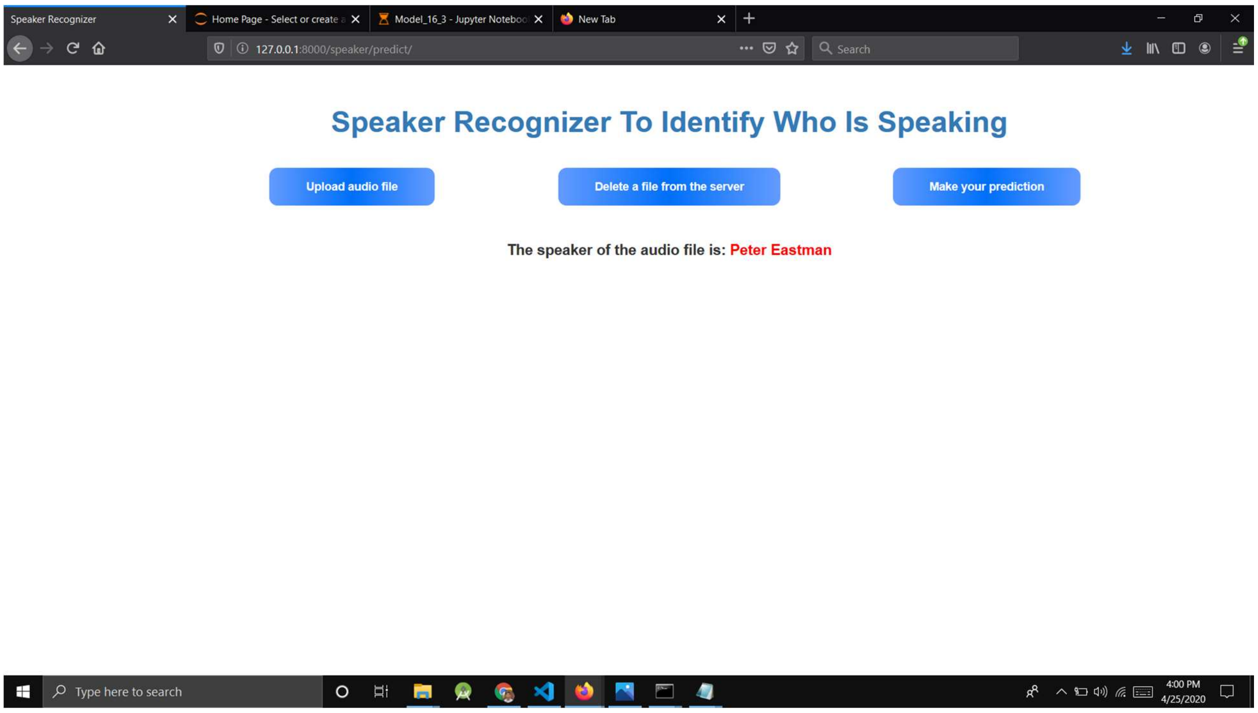
☐ 174-84280-0000.flac

☐ 174-84280-0008.flac

☐ 174-84280-0008_JwyJE11.flac

Predict

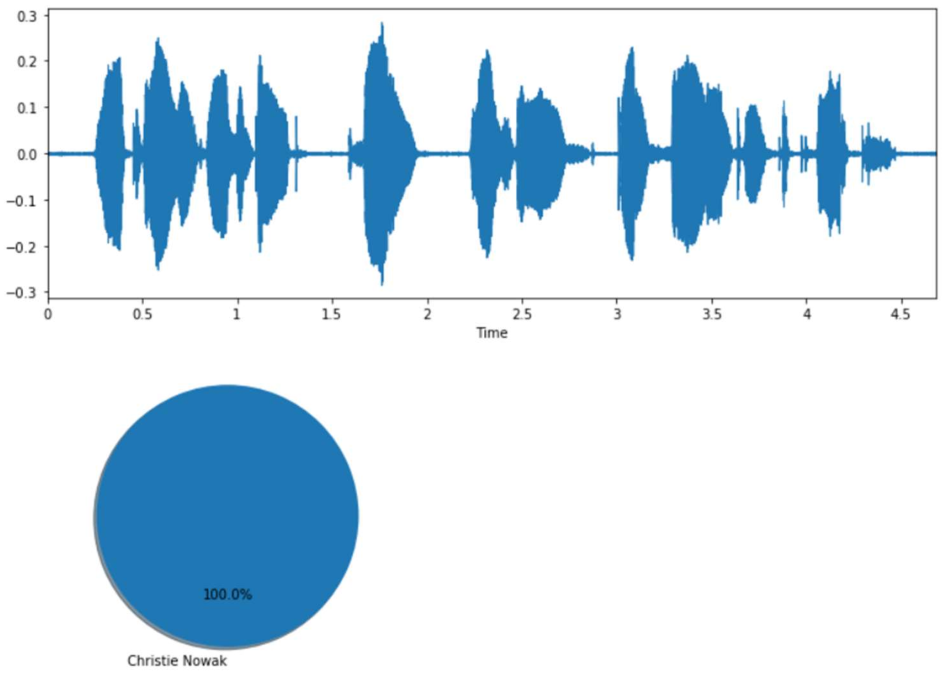
Screenshot 4: Selection of file for identification


Speaker Recognizer
Home Page - Select or create a
Model_16_3 - Jupyter Notebook
New Tab
127.0.0.1:8000/speaker/predict/
Search
Upload audio file
Delete a file from the server
Make your prediction
Type here to search
4:00 PM
4/25/2020

Speaker Recognizer To Identify Who Is Speaking

The speaker of the audio file is: Peter Eastman

Screenshot 5: Speaker identified successfully



Screenshot 6: Prediction along with Pie Chart representing accuracy

CHAPTER 7
CONCLUSION

The functionality implementation in the system was done after understanding all the system modules according to their particular requirements.

Functionalities that are successfully implemented are:

- Input Audio
- Play Audio File
- Recognition of Speaker
- Addition of New Speaker

After the successful implementation of the system along with all the modules and functionalities, comprehensive testing was performed to determine the loopholes and possible flaws/errors in the system.

CHAPTER 8 LIMITATIONS AND FUTURE ENHANCEMENTS

8.1 Limitations

- Direct audio input from microphone is not supported.
- User should have basic knowledge to handle the app.
- Non-technical user cannot add new speaker.
- It is Web Based Application.

8.2 Future Extension

- Feature to add manual audio clips for speaker recognition.
- Faster and more accurate detection of speaker.
- Development of Mobile Application for the same.
- Better UI.

CHAPTER 9
BIBLIOGRAPHY

- Stack Overflow:

  https://stackoverflow.com/

- Medium:

  https://medium.com/

- Analytics Vidhya:

  https://www.analyticsvidhya.com/

- GitHub:

  https://github.com/

- Tensorflow:

  https://www.tensorflow.org/tutorials