5. Component Development of Voice Recognition Model for Vernacular Gujarati Dialects

5.1 Introduction

This chapter discusses the systematic manner in which the key components necessary in designing a voice recognition system for the Gujarati dialect are developed. Introducing the proposed model, the chapter also offers a detailed description of the model's architecture in terms of the stages of the audio data processing and analysis. They show how the system takes and pre-processes raw audio inputs for the identification of speakers and dialects. Also, in this chapter, the developed model's interface for users is presented, making the main computerized model easily and convenient to use. Through the consolidation of all the components into the system model of the chapter, it shows how the voice recognition model is established to target the linguistic and regional issues in Gujarati speech.

5.2 Speech Data Collection

The data for the voice recognition model collected from a set of audio sources stored publicly on the internet, including **Gujarati movies**, **news channels**, **phone recordings** from relatives, and **radio conversations** provided by AI4Bharat [1]. This combination of sources is an assortment of wide variation in speech, accent, and tone across different Gujarati dialects. These dialects also have regional variations like Kathiyawadi, Standard Gujarati, Surti and Kutchhi, having their phonetic features and pronunciation. Since this data is unstructured, it was necessary to organize and preprocess it in a way that ensures it is usable for training machine learning models.

This dataset was gathered from sources in unstructured form, not previously labelled, and not in a format considered ready for machine learning. The author structured the data by organizing the audio samples according to key attributes like gender, dialect, speaker's region, and age group. This categorization allows the model to be trained using data from heterogeneous groups regarding speech variability and thereby classify and recognize speakers correctly independent of their background.

These unstructured data was carefully arranged into a structured format, keeping in mind a few key characteristics such as gender, dialect, speaker's region, and age group. This organization was highly necessary to ensure that the model could account for the variation that may be inherent in various dialects of Gujarati, along with the way different speaking styles are adopted based on the gender and age of the person.

This data is used to collect speakers across many major Gujarati dialects like Kathiyawadi, Standard Gujarati, Surti and Kutchhi to represent various linguistic diversities across Gujarat. This ensures that multiple accents and dialects are realized for Speaker Recognition. The speaker's gender and age group information were also used further segmenting data, which helps the model in learning the potential variations the characteristics may have based on these demographics.

This includes formal speech, such as news channels and movies, and informal conversational speech, such as recordings of phone calls. Since this data contains both types of speeches, it adds extra variability that the model will need to learn to cope with. Overall, these sources make the richness and diversity of the dataset greater, hence making it more representative of real-life scenarios where people speak across different contexts and environments.

Here, the researcher has organized data such that the training process should take into consideration a large variation in accents, different ways of speaking, and other acoustic conditions in the different Gujarati dialects. With a vast gathering and organization of data in such a manner by the researcher, the created dataset named “**vernacVoiceData**” and this dataset will go a long way to give a good backbone to a voice recognition system for successfully handling different Gujarati dialects in practical applications.

5.2.1 Privacy and Security Measures in Voice Data Collection

The data collection process for the development of the voice recognition model for vernacular Gujarati dialects has been done with due consideration to the privacy and security of the resource persons whose data was used. The data was primarily sourced from public repositories, including Gujarati movies, news channels, and phone call recordings from relatives. While acquiring this data, strict ethical guidelines were followed to make sure the individuals' privacy was not compromised.

The data is used strictly for research purposes only, informed consent having been sought and obtained from all parties whose voice data was utilized. Besides, personally identifiable information was anonymized to protect the identity of resource persons. Data collection was informed, with due care taken that the data itself was kept securely in a manner not breaching ethical research standards.

This approach respects the rights of the subjects while allowing for a high-quality and diverse dataset to be created for voice model training.

5.2.2 Sources of the Data

The unstructured audio data was gathered from various sources:

Gujarati Movies: The inclusion of movie dialogues provides a rich source of natural, expressive speech. These samples represent a wide range of emotions, speaking styles, and pronunciations specific to different Gujarati regions. The recognition system needs to learn various dialects. Movie dialogues also offer more structured speech, allowing for clear pronunciation and varied intonations.

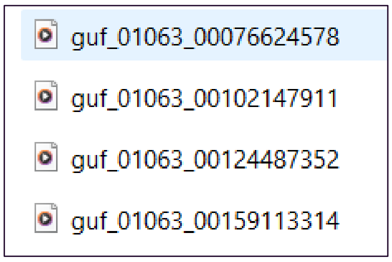
News Channels: The news in Gujarati is often quite formal and clearly presented; thus, this is a very key source for capturing standard patterns of speech and formal language. News anchors pronounce the words more clearly; it gives a contrasting data set to the more colloquial or casual speech present in other sources.

Phone Call Recordings: The telephone call recordings are from relatives and feature more conversational and informal speech, including regional slang, colloquial expressions, and different speech rates. The telephone calls introduce other variability from background noise and different audio quality, adding to the diversity of the dataset.

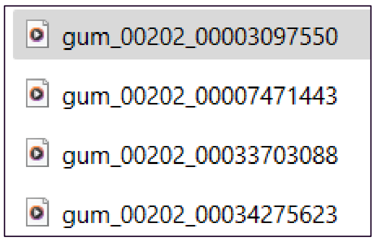
Radio Conversations from AI4Bharat: AI4Bharat [1] provided radio conversation data. These include unscripted and informal speech. Radio broadcasts contain several speakers with distinctive voices and accents, useful in training the model with variety of speech, including spontaneous speech conversations.

5.2.3 Samples of Collected Voice Data (Unstructured)

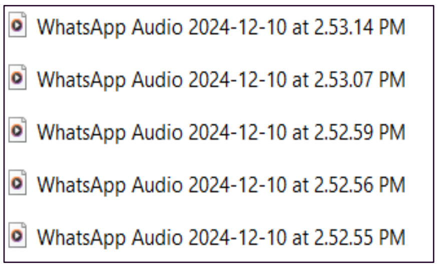
The speech dataset collected from different public resources has been received or downloaded in various form. Here in the below figures, the samples speech data were shown:


guf_01063_00076624578
guf_01063_00102147911
guf_01063_00124487352
guf_01063_00159113314

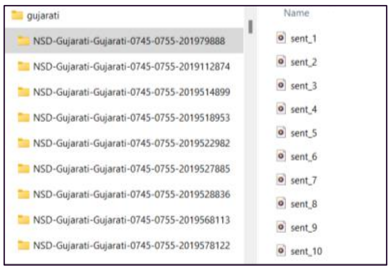
Public Repository


gum_00202_00003097550
gum_00202_00007471443
gum_00202_00033703088
gum_00202_00034275623

Public Repository


WhatsApp Audio 2024-12-10 at 2.53.14 PM
WhatsApp Audio 2024-12-10 at 2.53.07 PM
WhatsApp Audio 2024-12-10 at 2.52.59 PM
WhatsApp Audio 2024-12-10 at 2.52.56 PM
WhatsApp Audio 2024-12-10 at 2.52.55 PM

Call Recordings from Relatives


gujarati
NSD-Gujarati-Gujarati-0745-0755-201979888
NSD-Gujarati-Gujarati-0745-0755-2019112874
NSD-Gujarati-Gujarati-0745-0755-2019514899
NSD-Gujarati-Gujarati-0745-0755-2019518953
NSD-Gujarati-Gujarati-0745-0755-2019522982
NSD-Gujarati-Gujarati-0745-0755-2019527885
NSD-Gujarati-Gujarati-0745-0755-2019528836
NSD-Gujarati-Gujarati-0745-0755-2019568113
NSD-Gujarati-Gujarati-0745-0755-2019578122

Name
sent_1
sent_2
sent_3
sent_4
sent_5
sent_6
sent_7
sent_8
sent_9
sent_10

Radio Conversation by AI4Bharat

Figure 5- 2 Samples of collected data from various public resources

5.2.4 Nature of the Data (Unstructured Data)

Data from these sources is unstructured initially, with the audio files not having any predefined labels or organized format suitable for machine learning tasks. The unstructured nature of this data includes:

- Variations in audio quality, some of which include background noise, varying volume levels, and different types of distortions.
- Speech in different dialects comprises specific phonetic features, intonation, and accent.

- A mix of formal and informal speech, depending on source: movies, news, phone calls, radio.

5.3 Data Organizations

This unstructured data was at first structured and labelled based on many key factors that influence the process of speaker recognition.

5.3.1 Convert the audio files into standard format

Some basic sound processing operations such as converting the audio files to WAV or any standardized format is very important prior to any processing requirements of the particular speech and audio processing tasks. WAV (Waveform Audio File Format) is preferred due to it is an uncompressed and lossless format that is also commonly compatible with most audio processing tools and libraries. The collected audio data were converted into the WAV format.

5.3.2 Categorization of Unstructured Data

A structured format by categorizing the data into the following attributes:

Gender: The data is divided into male and female speakers to ensure that the model can identify speech patterns based on certain characteristics related to gender, such as pitch and tone.

Dialects: The data in this dataset were categorized according to the dialect of Gujarati, namely Kathiyawadi, Standard Gujarati, Surti, and Kutchhi, each having its different linguistic features, pronunciation, and intonation. This was done so that the model learns to differentiate between these types of dialects, which is again a very important characteristic for any voice recognition model that is being developed.

Geographic Region of Speaker: The speakers were further divided based on their geographic regions because regional influences are strong in the way people speak and pronounce words. This is important for regional variations in Gujarati so that the system can cope with a wide variety of speeches.

5.3.3 Define a Directory Structure

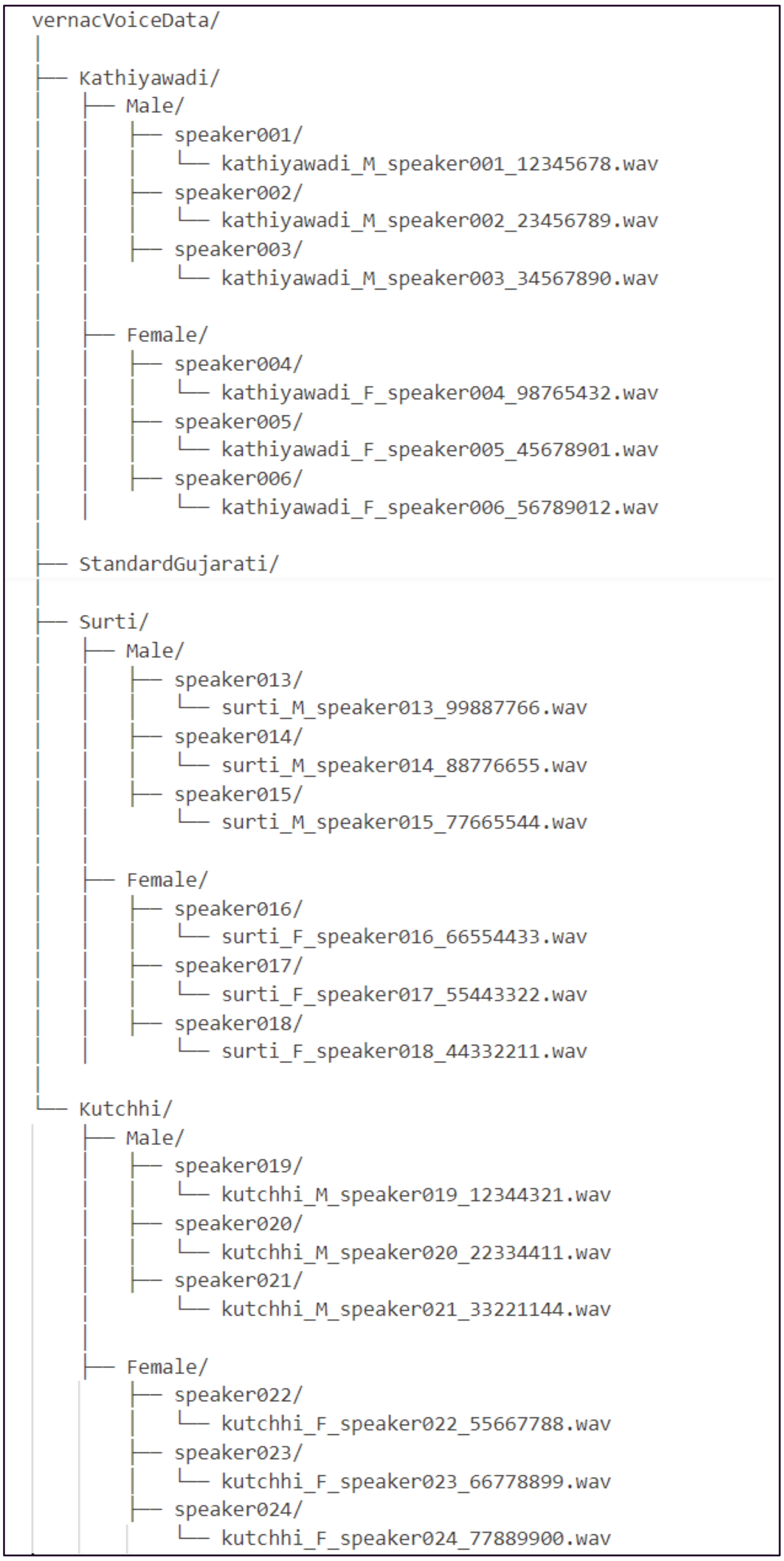
The parent directory is **vernacVoiceData** and this is sub-divided by various dialects. The dialect directory also sub- divides by gender Male and Female for each dialect group Kathiyawadi, Standard Gujarati, Surti and Kutchhi. Within the gender specific directories, subdirectories are given specific speaker ID.

Each speaker directory contains audio files formatted with a consistent naming convention:

- dialects_gender_speakerID_randomnumber.wav

An example audio file name could be:

- kathiyawadi_M_speaker001_0005646465.wav



```
vernacVoiceData/
│
├── Kathiyawadi/
│   ├── Male/
│   │   ├── speaker001/
│   │   │   └── kathiyawadi_M_speaker001_12345678.wav
│   │   ├── speaker002/
│   │   │   └── kathiyawadi_M_speaker002_23456789.wav
│   │   ├── speaker003/
│   │       └── kathiyawadi_M_speaker003_34567890.wav
│   │
│   ├── Female/
│   │   ├── speaker004/
│   │   │   └── kathiyawadi_F_speaker004_98765432.wav
│   │   ├── speaker005/
│   │   │   └── kathiyawadi_F_speaker005_45678901.wav
│   │   ├── speaker006/
│   │       └── kathiyawadi_F_speaker006_56789012.wav
│
├── StandardGujarati/
│
├── Surti/
│   ├── Male/
│   │   ├── speaker013/
│   │   │   └── surti_M_speaker013_99887766.wav
│   │   ├── speaker014/
│   │   │   └── surti_M_speaker014_88776655.wav
│   │   ├── speaker015/
│   │       └── surti_M_speaker015_77665544.wav
│   │
│   ├── Female/
│   │   ├── speaker016/
│   │   │   └── surti_F_speaker016_66554433.wav
│   │   ├── speaker017/
│   │   │   └── surti_F_speaker017_55443322.wav
│   │   ├── speaker018/
│   │       └── surti_F_speaker018_44332211.wav
│
└── Kutchhi/
    ├── Male/
    │   ├── speaker019/
    │   │   └── kutchhi_M_speaker019_12344321.wav
    │   ├── speaker020/
    │   │   └── kutchhi_M_speaker020_22334411.wav
    │   ├── speaker021/
    │       └── kutchhi_M_speaker021_33221144.wav
    │
    ├── Female/
        ├── speaker022/
        │   └── kutchhi_F_speaker022_55667788.wav
        ├── speaker023/
        │   └── kutchhi_F_speaker023_66778899.wav
        ├── speaker024/
            └── kutchhi_F_speaker024_77889900.wav
```

Figure 5- 3 Directory Structure of 'vernacVoiceData' Dataset

5.3.4 Annotations and Metadata

After organizing the audio data based on these attributes, each recording was annotated with relevant metadata. Additionally, metadata for all audio files is stored in a CSV file (shows in Table 5-1) at the root level. This file, named something like metadata.csv, contains the following columns:

Filename: The full name of the audio file.

Speaker Identity: Each speaker's identity is recorded to facilitate speaker identification or verification during the training and testing stages.

Dialect Label: Each audio sample is tagged with its corresponding dialect, which is essential for training the model to differentiate between dialectal variations.

Gender: The gender of the speaker (M for male, F for female).

Table 5- 1 Table shows sample metadata of audio files

Filename	Speaker	Gender	Dialect	Audio Length
Kathiyawadi_F_speaker001_22434537.wav	speaker001	Female	Kathiyawadi	6.2
Kathiyawadi_F_speaker001_11375426.wav	speaker001	Female	Kathiyawadi	4.1
Kutchhi_M_speaker037_30489034.wav	speaker037	Male	Kutchhi	8.96
Kutchhi_M_speaker037_95423530.wav	speaker037	Male	Kutchhi	9.64
Kutchhi_M_speaker037_70777302.wav	speaker037	Male	Kutchhi	10.15
StandardGujarati_F_speaker038_90537108.wav	speaker038	Female	Standard Gujarati	4.95
StandardGujarati_F_speaker038_07869767.wav	speaker038	Female	Standard Gujarati	5.8
StandardGujarati_F_speaker038_25362323.wav	speaker038	Female	Standard Gujarati	4.69
StandardGujarati_F_speaker038_09682410.wav	speaker038	Female	Standard Gujarati	4.95
Surti_M_speaker079_67775062.wav	speaker079	Male	Surti	14.99
Surti_M_speaker079_94598198.wav	speaker079	Male	Surti	14.99
Surti_M_speaker079_13901542.wav	speaker079	Male	Surti	14.99
Surti_M_speaker079_02255040.wav	speaker079	Male	Surti	14.99

The structured dataset forms the core resource for feature extraction, model training, and validation in the development of the voice recognition system. The diversity and richness of the dataset enable the model to learn the nuances of different Gujarati dialects, helping it recognize speakers from various backgrounds with high accuracy and reliability.

5.4 Designing User Interface for Voice Recognition (MVR) Model of Gujarati Dialects

For Voice Recognition Model of Gujarati Dialects, Researcher has developed a tool with Graphical User Interface named "Meera's Voice Recognition (MVR) Tool" for recognition and identifying a voice of speaker based on different regional dialects. This user interface allow user to choose audio WAV file of speaker's voice speaking in any dialects of regional Gujarati language and display the predicted speaker name on the output screen along with voice transcription and identification confidence for voice verification.

Basic purpose behind designing Voice Recognition Tool – Graphical Interface is to provide user a facility to upload audio file of speaker's voice stored in a hard drive. Voice Recognition Model will process audio given and will display Predicted Speaker's Name of recognized voice.

This tool is designed and developed using Jupyter Notebook. 'Tkinter' library of Python Programming Language is used to design and develop GUI, which generates the output. Audio browsed using this Tool is passed to Voice Recognition Model for further processing The Figure 5-3 shows the screenshot of user interface of MVR Tool.

5.4.1 Features of Voice Recognition (MVR) Tool

- Navigation Bar with required operations to deal with tools.
- Navigation bar has 4 Menus to interact with the Tools: Recognize Voice, Enrol New Speaker, Clear, Quit.
- Providing two windows for user options to recognize voice or enrol new voice of existing speakers by two menus in navigation bar.
- Allows user to choose audio file of speaker's voice in WAV format.

- Operation can be performed by simple clicking on predict voice button after the file upload option.
- Tool is designed in such a manner that is easy to use for a user who is familiar with any windows-based software application, it doesn’t require extra skills to interact with this Tool.

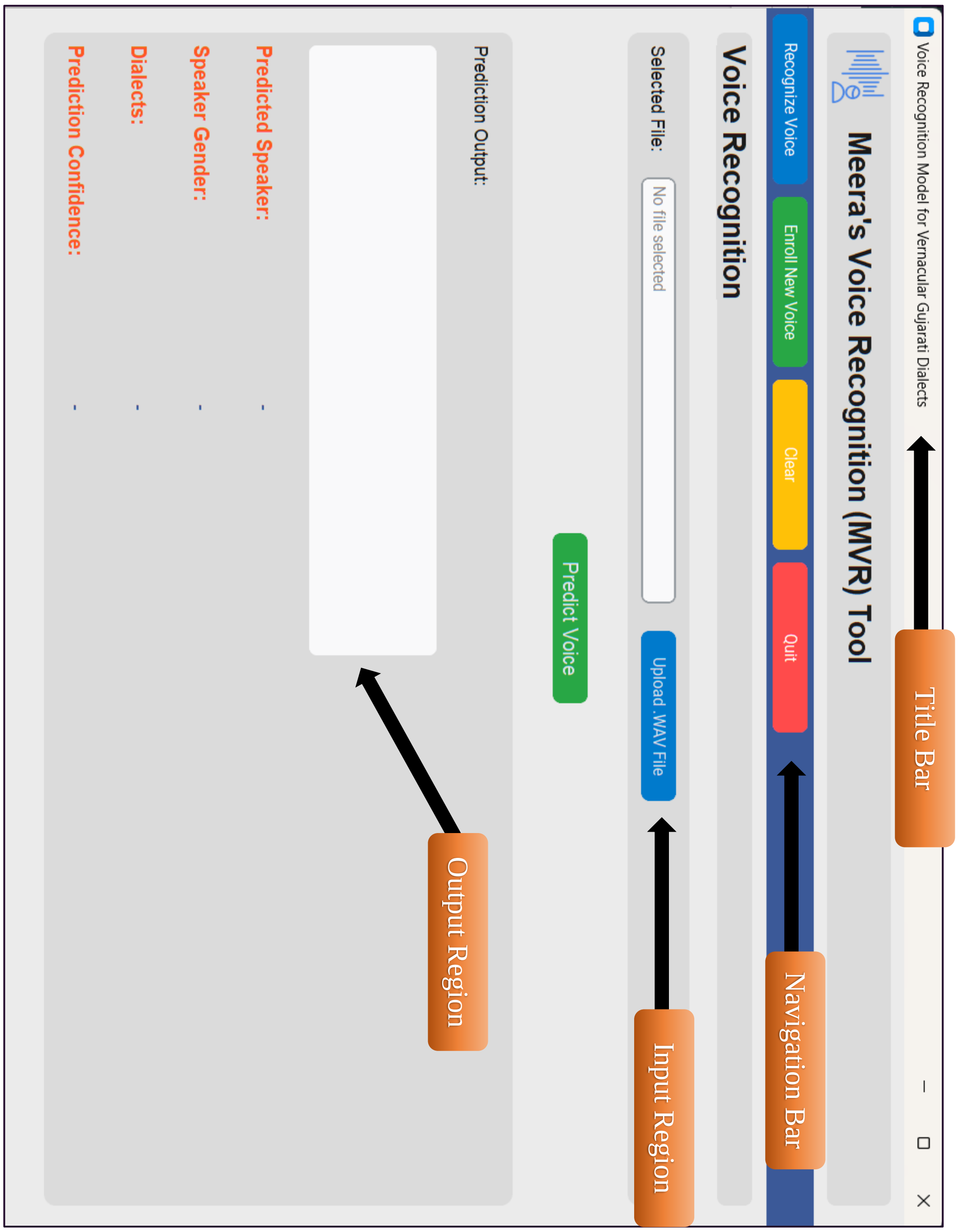


Figure 5- 5 Screenshot of Voice Recognition (MVR) Tool

5.4.2 Component of Voice Recognition (MVR) Tool

Title Bar: It consists of logo followed by text "Voice Recognition Tool" indicating title of Graphical User Interface shows in Figure 5-4.

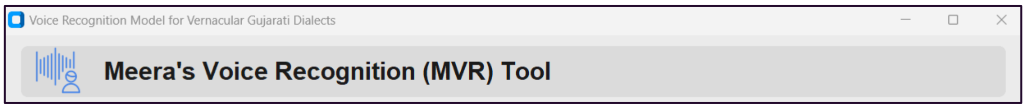

Voice Recognition Model for Vernacular Gujarati Dialects
Meera's Voice Recognition (MVR) Tool

Figure 5- 6 Title Bar of MVR Tool

Navigation Bar: Navigation bar shows in Figure 5-5 contains 4 Menus that indicates different operations. The first Menu 'Recognize Voice' used to recognize the speaker's voice based on audio file given. The second menu 'Enrol New Voice' for enrolment of new voice for existing speaker, third 'Clear' menu used to clear the input and out regions for both the screen and 'Quit' used to exit from the tool and terminate all the process.

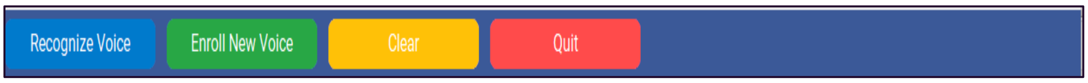


Figure 5- 9 Navigation Bar of MVR Tool

Recognize Voice Screen: This screen divided into two region Input Region and Output Region as show in figure 5-3.

Input Region: When clicked the 'upload' button, prompts directory name and a path from where user can upload the audio file for further processing. It is mandatory to provide this .WAV format audio file by user. Next user has to click on "Predict Voice" to recognise the voice of uploaded file.

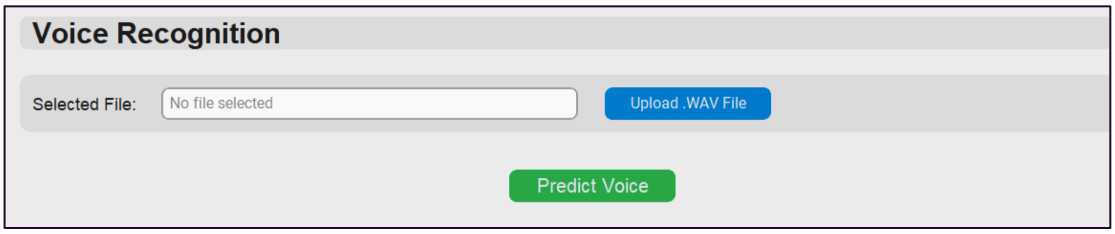
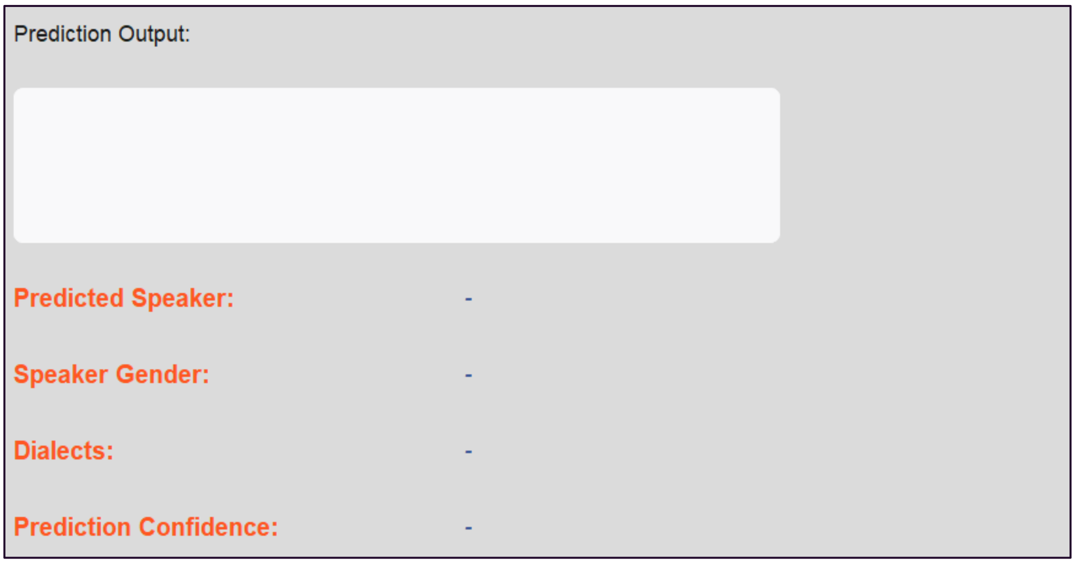


Figure 5- 12 Input Region of Recognize Voice window of MVR Tool

Output Region: The output region of Recognize screen have one text box that shows the process logs of Voice Recognition Model for uploaded audio file processing to recognize the speaker's voice. The label below textbox will shows the Predicted Speaker Name, gender and Dialect of uploaded audio along with confidence percentage as shown in Figure 5-7.

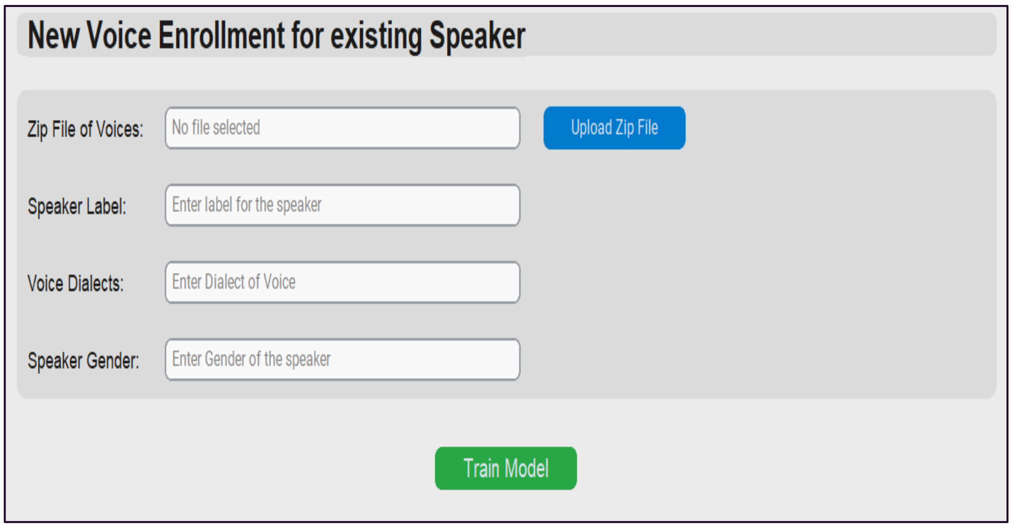


Prediction Output:

Predicted Speaker:	-
Speaker Gender:	-
Dialects:	-
Prediction Confidence:	-

Figure 5- 13 Output Region of Recognize Voice window of MVR Tool

Enrol New Voice Screen: This screen has multiple fields for uploading details of speaker like Zip file, Speaker Label, Voice Dialect and gender as show in figure 5-8 used for model re-training for new voice of existing speaker.



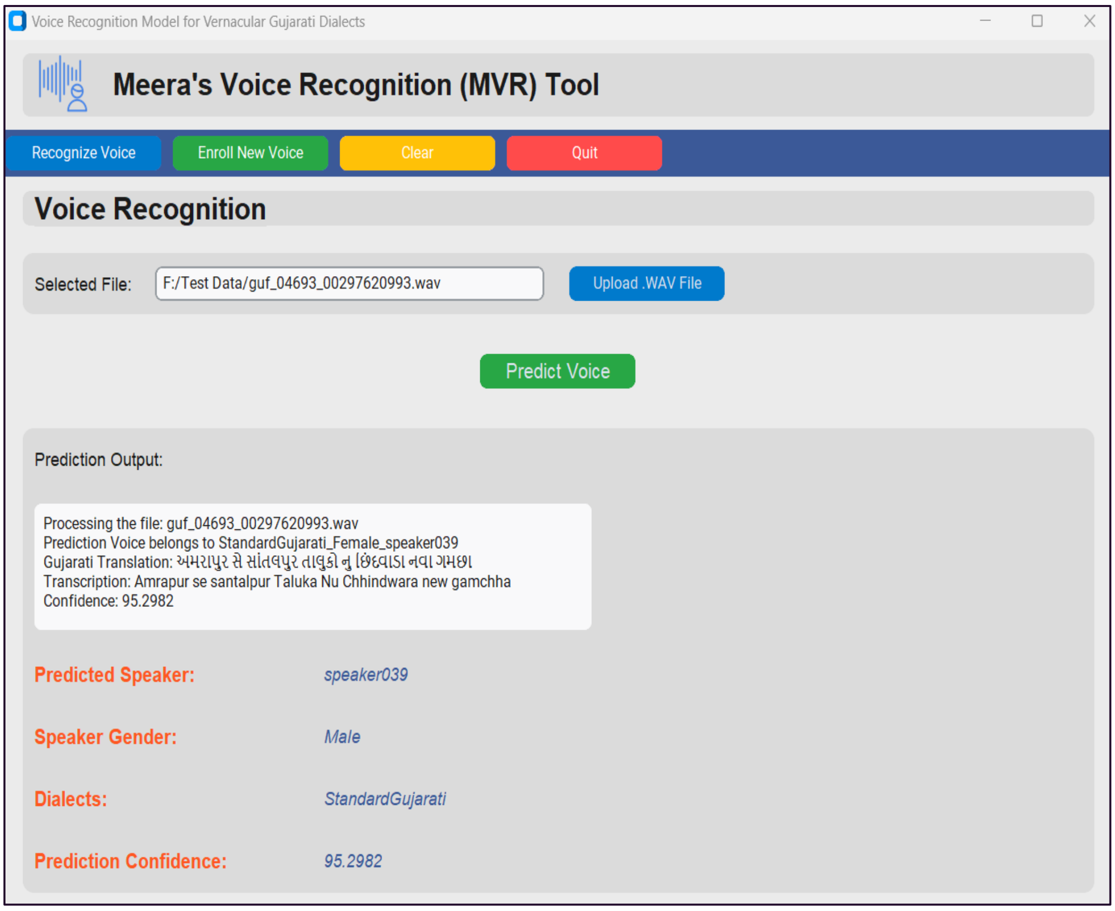
New Voice Enrollment for existing Speaker

Zip File of Voices:	No file selected	Upload Zip File
Speaker Label:	Enter label for the speaker	
Voice Dialects:	Enter Dialect of Voice	
Speaker Gender:	Enter Gender of the speaker	

Train Model

Figure 5- 16 Enrol New Voice window of MVR Tool

5.4.3 Sample Outputs


Voice Recognition Model for Vernacular Gujarati Dialects
Meera's Voice Recognition (MVR) Tool
Recognize Voice
Enroll New Voice
Clear
Quit
Voice Recognition
Upload .WAV File
Predict Voice

Selected File: F:/Test Data/guf_04693_00297620993.wav

Prediction Output:

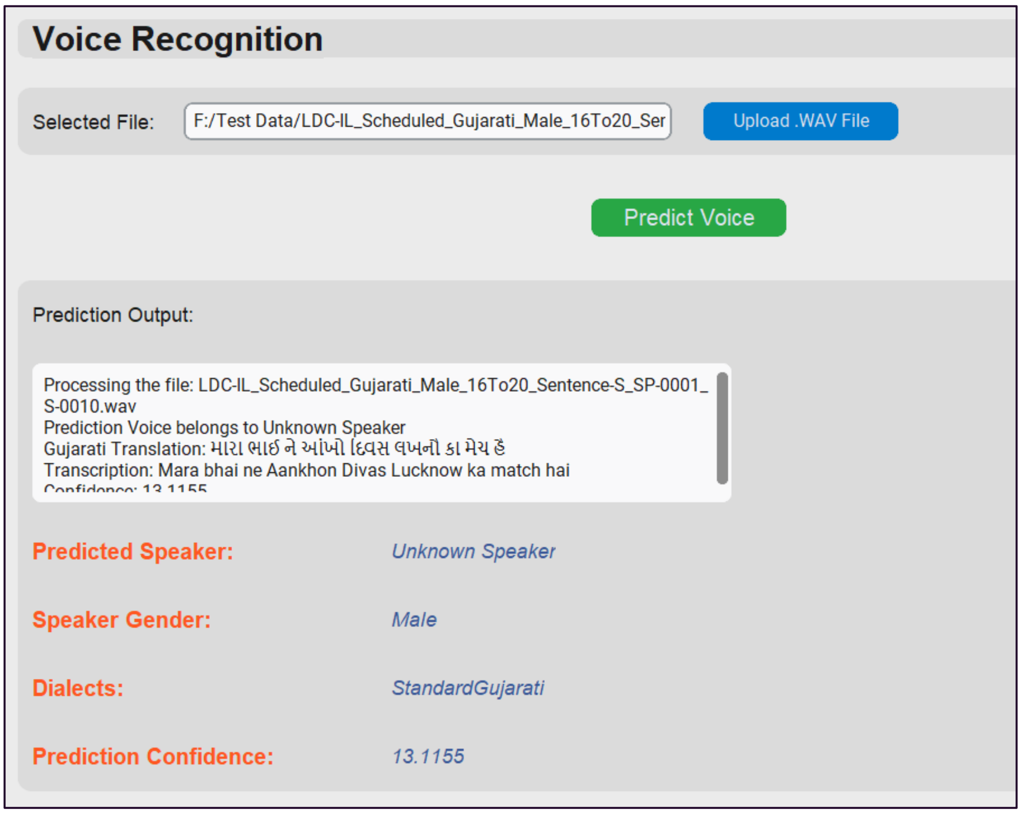
```
Processing the file: guf_04693_00297620993.wav
Prediction Voice belongs to StandardGujarati_Female_speaker039
Gujarati Translation: અમરાપુર સે સાંતલપુર તાલુકો નુ છિંદવાડા નવા ગમછા
Transcription: Amrapur se santalpur Taluka Nu Chhindwara new gamchha
Confidence: 95.2982
```

Predicted Speaker:	speaker039
Speaker Gender:	Male
Dialects:	StandardGujarati
Prediction Confidence:	95.2982

Figure 5- 17 Sample Output displaying Predicted Speaker along with gender, dialect & confidence of voice recognition of uploaded audio

Figure 5-9 shows screen shot of Voice Recognition Tool where audio file is uploaded and its output display in output region that shows the logs of processing the model and Predicted Speaker, Gender of the speaker, dialect of speaker's voice and prediction confidence used to verify the results.

If audio file of any Gujarati dialects is given as input to proposed recognition tool than it processed using the MVR model and then display the results in the bases of prediction confidence. If the prediction confidence is below threshold (0.8) than it will show the Unknown Speaker else it will show the actual speaker ID along with its identified gender and the dialect in which the voice is spoken or recorded. In below figure 5-11 shows the results where the researcher tried to test the MVR tool by providing the voice samples of a speaker that is not enrol during training of the proposed model.



Selected File: F:/Test Data/LDC-IL_Scheduled_Gujarati_Male_16To20_Ser

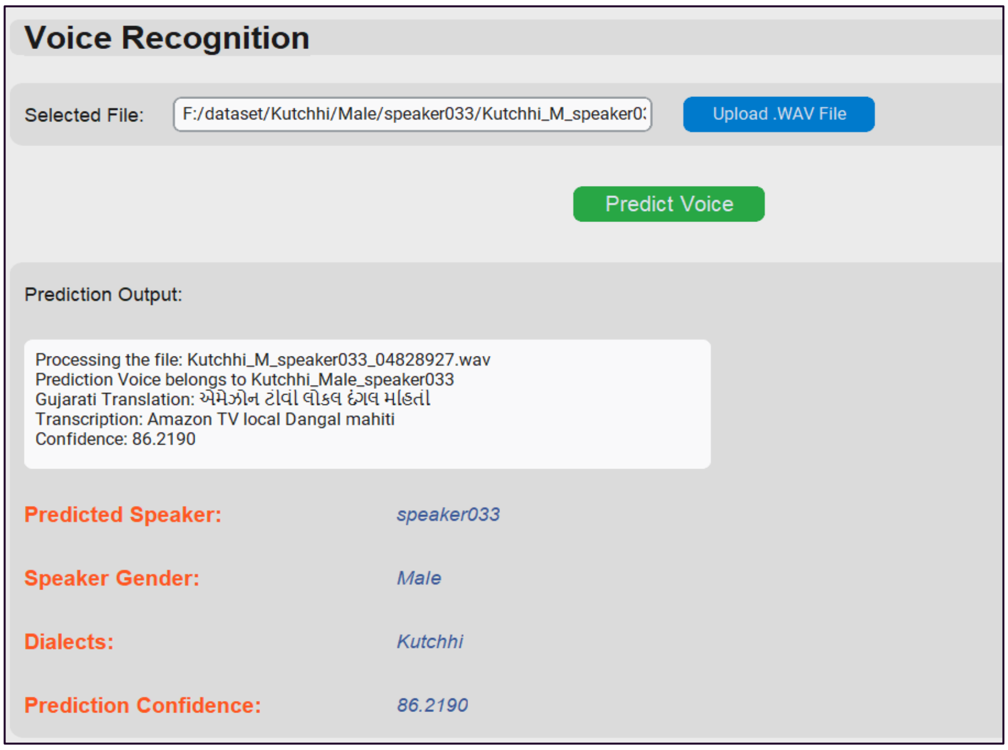
Prediction Output:

```
Processing the file: LDC-IL_Scheduled_Gujarati_Male_16To20_Sentence-S_SP-0001_
S-0010.wav
Prediction Voice belongs to Unknown Speaker
Gujarati Translation: મારા ભાઈ ને આંખો દિવસ લખનૌ કા મેચ હૈ
Transcription: Mara bhai ne Aankhon Divas Lucknow ka match hai
Confidence: 13.1155
```

Predicted Speaker:	Unknown Speaker
Speaker Gender:	Male
Dialects:	StandardGujarati
Prediction Confidence:	13.1155

Figure 5- 18 Sample output of voice sample of random speaker of any dialect

Figure 5-10 represents voice sample input of audio spoken by a male speaker 33 speaking in Kutchhi dialect. Proposed model has correctly identified all the information correctly and is displayed as output.



Selected File: F:/dataset/Kutchhi/Male/speaker033/Kutchhi_M_speaker0:

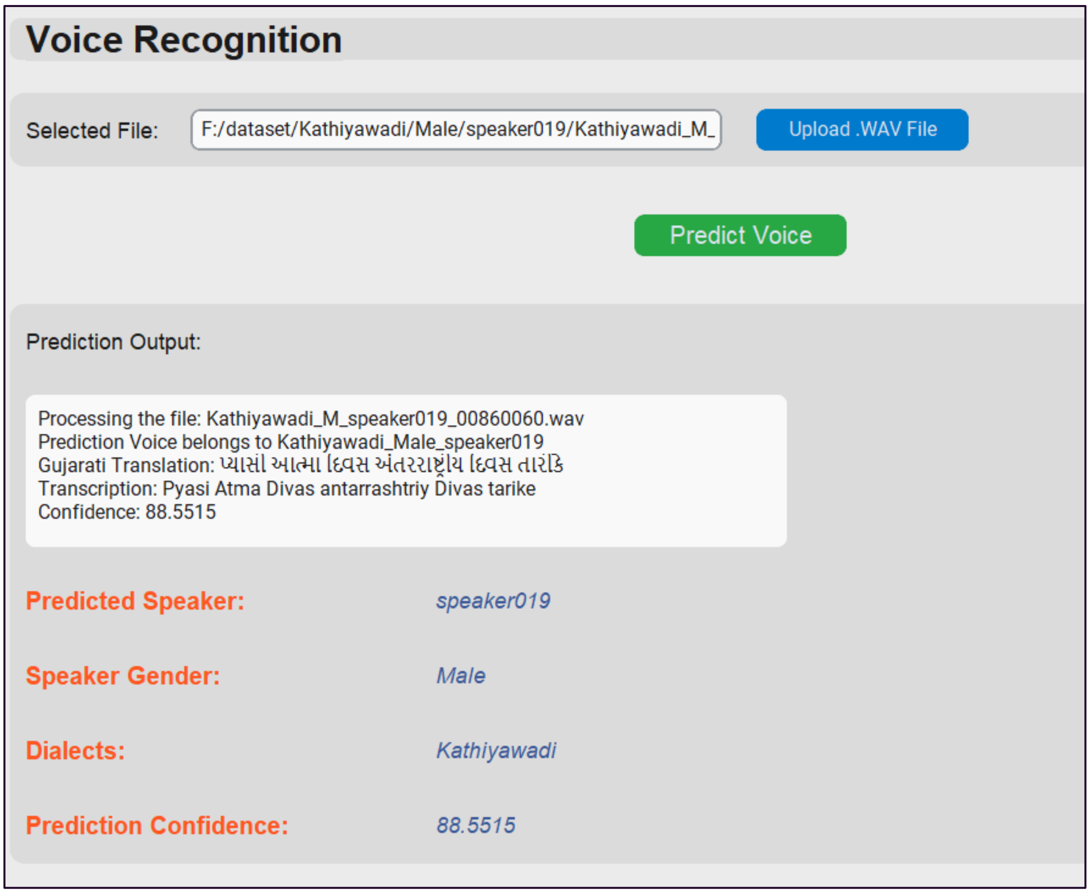
Prediction Output:

```
Processing the file: Kutchhi_M_speaker033_04828927.wav
Prediction Voice belongs to Kutchhi_Male_speaker033
Gujarati Translation: એમેઝોન ટીવી લોકલ દંગલ માહિતી
Transcription: Amazon TV local Dangal mahiti
Confidence: 86.2190
```

Predicted Speaker:	speaker033
Speaker Gender:	Male
Dialects:	Kutchhi
Prediction Confidence:	86.2190

Figure 5- 19 Sample output of voice sample of speaker033 of Kutchhi dialect

Figure 5-12 depicts a voice sample input where the audio is spoken by a male speaker in the Kathiyawadi dialect. The proposed model has accurately identified all the relevant information, including the speaker's gender, dialect, and voice characteristics. The prediction was made with a confidence score of 88.55%, and the final output is displayed accordingly.

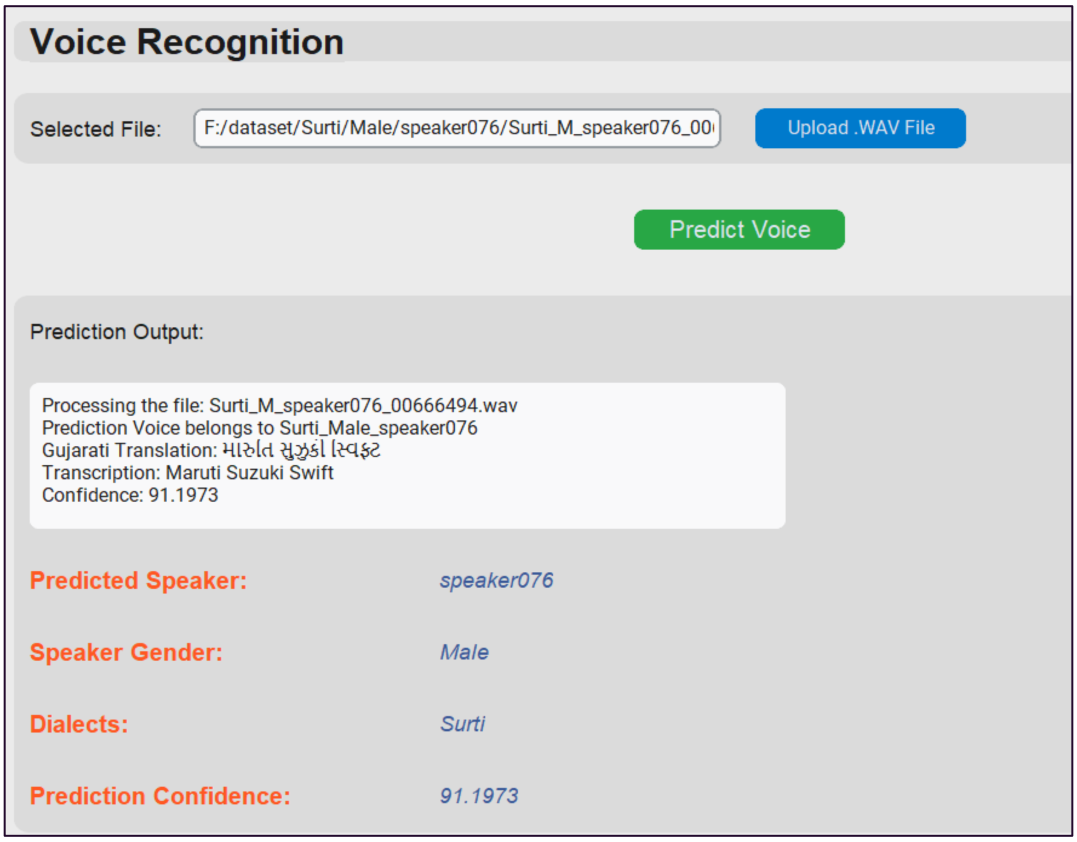


```
Processing the file: Kathiyawadi_M_speaker019_00860060.wav
Prediction Voice belongs to Kathiyawadi_Male_speaker019
Gujarati Translation: પ્યાસીં આત્મા દિવસ અંતરરાષ્ટ્રીય દિવસ તારીકે
Transcription: Pyasi Atma Divas antarrashtriy Divas tarike
Confidence: 88.5515
```

Predicted Speaker:	speaker019
Speaker Gender:	Male
Dialects:	Kathiyawadi
Prediction Confidence:	88.5515

Figure 5- 20 Sample output of voice sample of Kathiyawadi dialect

As displayed in Figure 5-13, the voice sample input entails an audio that was spoken by a male nonprofessional voice actor, who was speaking in the Surti dialect. To the core of the idea, the proposed model has correctly identified all the necessary input parameters. It was satisfied that the system was able to identify the gender of the speaker, dialect and characteristics of the voice. The forecast was given at 91.19% confidence level, and it is presented in the output screen as the final conclusion.



```
Processing the file: Surti_M_speaker076_00666494.wav
Prediction Voice belongs to Surti_Male_speaker076
Gujarati Translation: મારુતિ સુઝુકી સ્વિફ્ટ
Transcription: Maruti Suzuki Swift
Confidence: 91.1973
```

Predicted Speaker:	speaker076
Speaker Gender:	Male
Dialects:	Surti
Prediction Confidence:	91.1973

Figure 5- 21 Sample output of voice sample of Surti dialect

5.5 Pseudo code used to build Voice Recognition Tool and Voice Recognition Model for Vernacular Gujarati Dialects

By presenting the pseudocode for the Voice Recognition Model and developed tool steps, this section aims to provide an intuitive understanding of the logic and algorithms driving the system. The pseudocode serves as a blueprint, detailing the systematic approach employed to build an innovative solution for Gujarati voice recognition. All the procedures show below to implement proposed research work.

Table 5- 2 Pseudo Code for MVR_main

Procedure Name: MVR_main
Purpose: Predict the speaker of an audio file, transcribe the audio, and translate the text into Gujarati.
Input: test_audio_path ← Path to the audio file. model_file ← Path to the saved model. scaler_file ← Path to the scaler file. label_encoder_file ← Path to the label encoder file. threshold ← Confidence threshold for prediction (default = 0.8).
Output: A dictionary containing: Predicted_Speaker, Prediction_Confidence, Transcription, Gujarati_Translation
Variables Used: confidence: Prediction confidence score. predicted_speaker: Name of the predicted speaker. transcribed_text: Text obtained from audio transcription. translated_text: Gujarati translation of the transcribed text. result: Dictionary to store the output values.
begin set test_audio_path set model_file set scaler_file

```
set label_encoder_file
set threshold
**call** mvr_main store the returned value in result.
**for** each key-value pair in result:
display key: value.
set test_audio_path
set model_file
set scaler_file
set label_encoder_file
set threshold
**end for**
**end**
```

Table 5- 3 Pseudo Code for MVR_noise_reduction

Procedure Name: MVR_noise_reduction
Purpose: To reduce noise from audio files in a directory and save the cleaned versions in an output directory.
Input: input_dir (directory path for input audio), output_dir (directory path for cleaned audio), noise_duration (duration of noise sample), extension (audio file extension).
Output: Cleaned audio files saved to the specified output directory.
Variables Used: input_path, output_path, y (audio signal), sr (sampling rate), noise_sample, y_denoised.

Pseudocode:

```
Begin
Create output_dir if it doesn’t exist.
Loop through each file in input_dir.
If the file extension matches extension:
    Set input_path and output_path.
    Load the audio file using librosa to get y (audio) and sr (sampling rate).
    Extract noise_sample from the first noise_duration seconds.
    Perform noise reduction using noisereduce with y and noise_sample.
    Save the cleaned audio to output_path using soundfile.
    Print success message with output_path.
End
```

Table 5- 4 Pseudo code for MVR_silence_removal

Procedure Name: MVR_silence_removal
Purpose: To remove silence from audio files in a directory and save the processed files in an output directory.
Input: input_dir (directory path for input audio), output_dir (directory path for processed audio), silence_thresh (silence threshold in dBFS), min_silence_len (minimum duration of silence in ms), extension (audio file extension).
Output: processed audio files with silence removed, saved in the specified output directory.
Variables Used: input_path, output_path, audio, nonsilent_intervals, processed_audio, start, end

Pseudocode:

```
Begin
Create output_dir if it doesn't exist.
Loop through each file in input_dir.
    If the file extension matches extension:
        Set input_path and output_path.
        Load the audio file using AudioSegment.
        Detect non-silent segments using detect_nonsilent with min_silence_len and silence_thresh.
        Combine non-silent segments into processed_audio.
        Export the processed_audio to output_path and print success message.
End
```

Table 5- 5 Pseudo Code for MVR_audio_segmentation

Procedure Name: MVR_audio_segmentation
Purpose: Segments audio files into chunks of specified length (e.g., 10 seconds) if the audio length exceeds a minimum threshold (e.g., 15 seconds).
Input: input_dir (directory path for input audio files), output_dir (directory path for segmented audio files), segment_length_s (desired segment length in seconds), min_length_s (minimum file length to segment in seconds), extension (audio file extension)
Output: Segmented audio files saved to the specified output directory.
Variables Used: segment_length_ms, min_length_ms, input_path, audio, total_duration, segment_number, segment, segment_filename, segment_path.

Pseudocode:

```
Begin
Convert segment_length_s and min_length_s to milliseconds.
Ensure output_dir exists.
Loop through each file in input_dir.
    If file extension matches extension:
    Set input_path and load the audio file.
     If audio length > min_length_ms:
         Split audio into segments of segment_length_ms.
         Save each segment to output_dir with a unique name.
        Print success message for each segment.
     Else, skip file with a message.
End
```

Table 5- 6 Pseudo Code for MVR_audio_segmentation

Procedure Name: MVR_audio_segmentation
Purpose: Segments audio files into chunks of specified length (e.g., 10 seconds) if the audio length exceeds a minimum threshold (e.g., 15 seconds).
Input: input_dir (directory path for input audio files), output_dir (directory path for segmented audio files), segment_length_s (desired segment length in seconds), min_length_s (minimum file length to segment in seconds), extension (audio file extension)
Output: Segmented audio files saved to the specified output directory.
Variables Used: segment_length_ms, min_length_ms, input_path, audio, total_duration, segment_number, segment, segment_filename, segment_path.

Pseudocode:

Begin

```
Convert segment_length_s and min_length_s to milliseconds.
Ensure output_dir exists.
Loop through each file in input_dir.
    If file extension matches extension:
    Set input_path and load the audio file.
     If audio length > min_length_ms:
         Split audio into segments of segment_length_ms.
         Save each segment to output_dir with a unique name.
        Print success message for each segment.
     Else, skip file with a message.
End
```

Table 5- 7 Pseudo Code for MVR_resample_audio_files

Procedure Name: MVR_resample_audio_files
Purpose: Resamples audio files in a directory to a specified sample rate (e.g., 16,000 Hz) and saves them in an output directory.
Input: input_dir (directory path for input audio files), output_dir (directory path for resampled audio files), target_sr (desired sample rate in Hz), extension (audio file extension).
Output: Audio files resampled to the specified sample rate, saved in the output directory.
Variables Used: input_path, output_path, y (audio signal), sr (original sample rate), y_resampled (resampled audio).

Pseudocode:

```
Begin
Ensure output_dir exists.
Loop through each file in input_dir.
    If the file extension matches extension:
        Set input_path and output_path.
        Load the audio file using librosa to get y (audio) and sr (original sample rate).
    If sr != target_sr,
        resample the audio to target_sr and save it to output_path.
    Else, save the audio without resampling to output_path.
        Print whether the file was resampled or skipped.
End
```

Table 5- 8 Pseudo Code for MVR_extract_flat_mfcc

Procedure Name: MVR_extract_flat_mfcc
Purpose: Extracts MFCC (Mel-frequency cepstral coefficients) features from an audio file, ensuring the result is a fixed length by padding or truncating, and flattens the MFCC matrix into a 1D array.
Input: file_path (path to the audio file), n_mfcc (number of MFCC coefficients to extract, default is 13), max_pad_len (maximum length of the feature vector, default is 224).
Output: A flattened 1D array of MFCC features for the given audio file, or None if an error occurs.
Variables Used: audio, sr, mfcc, pad_width.

Pseudocode:

```
Begin
Try to load the audio file at file_path using librosa.load to get audio and sr (sample rate).
If audio is empty, raise an error and return None.
Compute mfcc features using librosa.feature.mfcc with n_mfcc and audio.
Check the shape of mfcc:
    If the number of frames in mfcc < max_pad_len:
        Pad mfcc with zeros to match max_pad_len.
    Else if the number of frames > max_pad_len:
        Truncate mfcc to max_pad_len.
        Flatten mfcc into a 1D array.
        Return the flattened mfcc.
        Catch and print any exceptions that occur, returning None for problematic files.
End
```

Table 5- 9 Pseudo Code MVR_load_data_ml

Procedure Name: MVR_load_data_ml
Purpose: Loads and processes audio data by extracting MFCC features and associating them with speaker labels from a CSV file.
Input: dataset_path (path to the directory containing audio files), label_csv (path to the CSV file containing filenames and speaker labels).
Output: X (numpy array of MFCC feature vectors), y (numpy array of corresponding speaker labels).
Variables Used: labels_df, X, y, file_path, mfcc, label.

Pseudocode:

```
Begin
Read the CSV file at label_csv into a DataFrame labels_df.
Clean the column names in labels_df by stripping whitespace.
Initialize empty lists X (for features) and y (for labels).
Traverse the dataset_path directory using os.walk:
For each file in files:
    If the file extension is .wav:
        Set file_path as the full path to the audio file.
        Check if the file is listed in the filename column of labels_df.
        Extract MFCC features using extract_flat_mfcc(file_path).
        If MFCC extraction is successful (mfcc is not None):
            Append mfcc to X.
            Find the label in labels_df where the filename matches file,
            and  append it to y.
            Convert X and y to numpy arrays.
            Return X and y.
End
```

Table 5- 10 Pseudo Code MVR_train_svm_classifier

Procedure Name: MVR_train_svm_classifier
Purpose: Preprocesses audio features and labels, trains an SVM classifier, and evaluates its performance on training and test datasets.
Input: dataset_path (path to audio data), label_csv (CSV file with labels), test_size (fraction for test data, e.g., 0.3), random_state (seed for reproducibility).
Output: Accuracy scores, classification report for the test data.
Variables Used:

X, y, label_encoder, y_encoded, X_train, X_test, y_train, y_test, scaler, model, y_train_pred, y_test_pred.
Pseudocode: **Begin** **Load** data using load_data_ml, returning X (features) and y (labels). Encode y labels using LabelEncoder. **Split** the dataset into training and test sets using train_test_split with test_size and random_state. Normalize the features: Fit and transform X_train using StandardScaler.   Transform X_test using the same scaler. Initialize an SVM classifier (SVC) with an RBF kernel and random_state. Train the classifier on X_train and y_train. Evaluate the model: Predict training labels (y_train_pred) and calculate training accuracy.   Predict test labels (y_test_pred) and calculate test accuracy. **Print** training and test accuracies. Generate a classification report for y_test predictions, using classification_report with the class names. Print the overall accuracy of the model on the test data. **End**

Table 5- 11 Pseudo Code MVR_predict_speaker_svm

Procedure Name: MVR_predict_speaker_svm
Purpose: Predicts the speaker of an audio file using a pre-trained SVM classifier, ensuring the prediction confidence exceeds a specified threshold.
Input: audio_path (path to the audio file), model_file (path to the saved SVM model), scaler_file (path to the saved scaler), label_encoder_file (path to the saved label encoder),

threshold (confidence threshold, default is 0.7).

Output:

confidence (prediction confidence score), predicted_label (predicted speaker label or "Unknown Speaker").

Variables Used:

model, scaler, label_encoder, mfcc, prediction_proba, predicted_class, confidence.

Pseudocode:

```
Begin
Load the saved model, scaler, and label_encoder using load.
Extract MFCC features from the audio file using extract_flat_mfcc(audio_path).
If MFCC extraction fails (mfcc is None),
    print an error message and return 0.0, "Unknown Speaker".
    Scale the extracted MFCC features using the loaded scaler.
    Use the model to predict the class probabilities (prediction_proba) for the scaled features.
    Identify the class with the highest probability (predicted_class) and its confidence score (confidence).
    If confidence < threshold:
        Print a message indicating low confidence.
        Return confidence, "Unknown Speaker".
    Else, return confidence and the corresponding class label using the label_encoder.
Catch and handle any exceptions during prediction, returning 0.0, "Unknown Speaker" if an error occurs.
End
```

References

[1] B. A. Alejandro Gutman, “The Language Gulper.” **https://www.languagesgulper.com/eng/Gujarati.html**

[2] “Feature Extraction.”

https://www.sciencedirect.com/topics/computer-science/feature-extraction