DATA MINING PROJECT

CIA 2

SPEECH DATA

APPLICATION CHOSEN : SPEAKER RECOGNITION (OR) VOICE AUTHENTICATION

SUBMITTED BY :

NIGHIL NATARAJAN – 23011101085

RAGHAV SRIDHARAN - 23011101109

AI/DS – ‘B’

ANALYSIS OF CHALLENGES OF SPEECH DATA

1.Background Noise:

- Background noise often interferes with the accuracy of speech recognition systems, especially in outdoor or public environments.
- Example: Virtual assistants like Alexa fail to recognize commands during loud music or traffic sounds.

2.Accent :

- The speaking accent differs according to the social and personal situations (e.g., physiological and cultural aspects – Ambiguity between native and non-native slang) .

3.Speaker Overlap:

- Processing speech data with overlapping speakers in group discussions remains a major challenge.
- Example: Virtual meeting platforms like Zoom fail to identify individual speakers in multi-person conversations.

4. Speed of Speech :

- The speech recognition systems find difficulty separating segments of continuous speedy speech signals (depends upon situations and physical stress).
- The pace of speaking may affect in pronunciation through phoneme reduction, time expansions and compressions.

5.Temporal Dependency:

- The sequential nature of speech requires models to consider context over time, which traditional methods struggle to achieve.

6.Acoustic & Noise-Related Challenges:

- Microphone Variability – Differences in recording devices cause frequency response variations.
- Channel Distortion – Compression, packet loss, and filtering alter the speech signal.

7.Multilingual Challenges:

- Handling code-switching in multilingual speech, such as Hindi-English mixtures, remains a limitation for many systems.

8. Processing of Homophones:

- Homophones are the words that have different meanings but sounds same when pronounced (e.g., “There” and “Their”, “Be” and “Bee”).
- In Speech Recognition Systems, it is very difficult at the word level to recognize which one is the correct intended word.

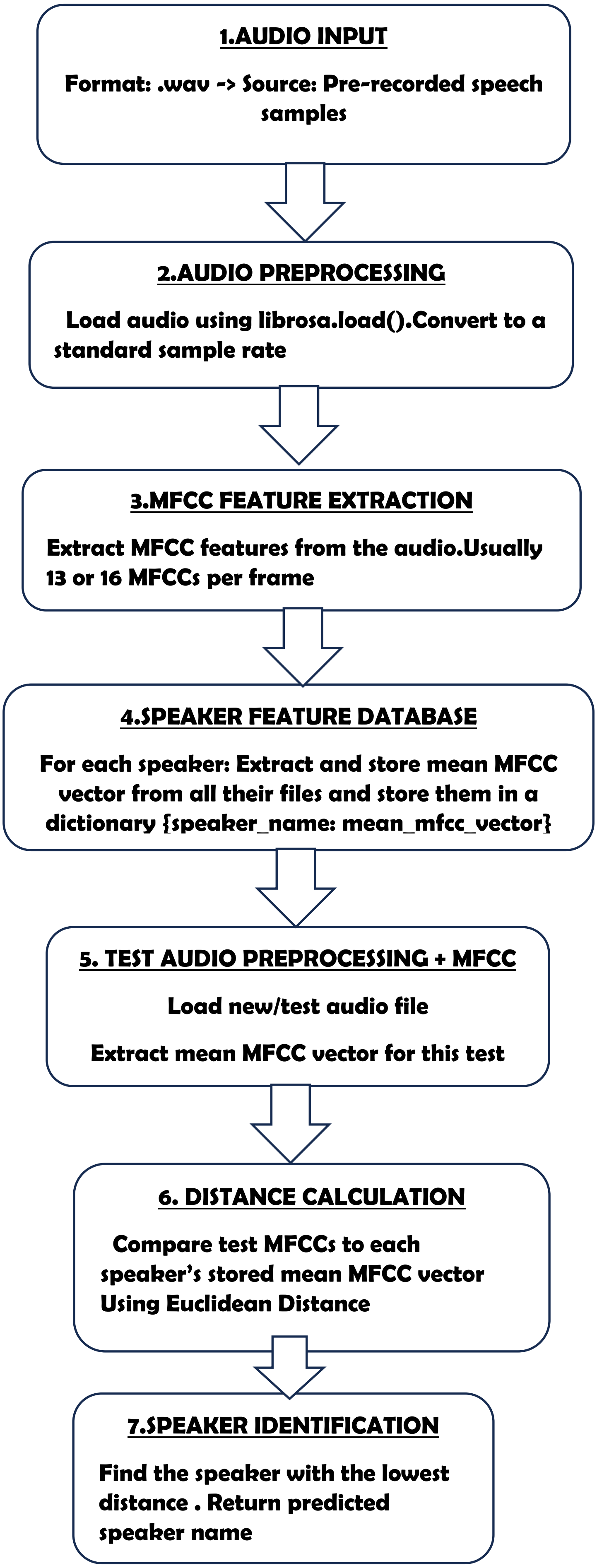
9. Coarticulation Effect:

- Phonemes change based on surrounding sounds (e.g., "t" in "top" vs. "stop"), making segmentation difficult. Solution:
- Use context-dependent phoneme models (triphones) instead of isolated phonemes.
- Similar-sounding phonemes (e.g., "b" and "p") can lead to misinterpretation in speech recognition systems, especially in noisy environments.

APPLICATION SELECTION

- Application chosen : **Speaker Recognition or Voice Authentication.**
- The chosen application for this project is a **Speaker Recognition System**, which is designed to identify or verify a person based on their voice.
- The primary goal of this application is to distinguish between speakers in a given audio dataset using computational models.

ARCHITECTURE DIAGRAM



MODULE DESCRIPTION

1. Audio Input Module

- **Function:**
  Takes in raw audio files (e.g., .wav or .flac) from users or datasets.
- **Technique/Tool Used:**
  - librosa.load() is used to load the audio signal into an array format.
  - Original sampling rate is preserved (sr=None).
- **Purpose:**
  To convert speech into a format that can be processed by digital algorithms.

2. Preprocessing Module

- **Function:**
  Prepares the audio signal for feature extraction by normalizing duration, removing silence (if needed), and standardizing the sample rate.
- **Technique/Tool Used:**
  - Implicit preprocessing handled by librosa.
  - Converts stereo to mono and resamples if needed.
- **Purpose:**
  Ensures consistent input across all audio samples to maintain model performance.

3. Feature Extraction Module

- **Function:**
  Extracts unique characteristics of a speaker's voice.
- **Technique/Tool Used:**
  - **MFCC (Mel-Frequency Cepstral Coefficients)** using librosa.feature.mfcc().

- Typically extracts 13 or 16 coefficients.
- The MFCCs is represented as a 2D matrix (no_of_frames,no_of_mfccs)

- **Purpose:**
  MFCCs model how humans perceive sound. They compactly represent the speech signal and are widely used in speaker and speech recognition.

4. Speaker Modeling Module

- **Function:**
  Creates a reference model for each speaker using their extracted MFCC features.
- **Technique/Tool Used:**
  - For each speaker, all MFCC frames are stacked using np.vstack() and the mean MFCC vector is computed column wise
  - Stored in a dictionary format: {speaker_name: mean_mfcc_vector}.
- **Purpose:**
  Serves as a database of known speaker profiles for comparison during prediction.

5. Test Audio Processing Module

- **Function:**
  Processes new audio files in the same way as training files: preprocessing and MFCC extraction.
- **Technique/Tool Used:**
  - Again uses librosa for loading and MFCC generation.
  - Ensures consistent feature format with training data.

- **Purpose:**
  Allows the model to analyze new input voices for identification

6. Similarity Matching Module

- **Function:**
  Compares the mean MFCC feature vector of test audio with each stored speaker model.
- **Technique/Tool Used:**
  - **Euclidean Distance** is used from scipy.spatial.distance.
  - Measures how close the test sample is to each stored speaker's voice.
- **Purpose:**
  Finds the best match by calculating the smallest distance between feature sets.

7. Speaker Identification Module

- **Function:**
  Determines the predicted speaker based on the closest match from the distance calculations.
- **Technique/Tool Used:**
  - A simple loop to find the speaker with the minimum average distance.
  - Accuracy calculation is also included in the prediction() function.
- **Purpose:**
  Outputs the final prediction: the identity of the speaker from a known list.

DATA SELECTION AND PREPROCESSING

- Voice samples were obtained from volunteers in the immediate surroundings for the purpose of speaker recognition.

Preprocessing Methods Used :

- **Audio Loading**:
  Audio files are loaded using librosa.load(), which converts them to mono automatically.
- **Original Sampling Rate Retained**:
  sr=None is used to keep the original sample rate of the audio file.
- **Direct Feature Input**:
  The raw audio signal is directly passed to MFCC extraction without extra modifications.

CODE :

```
import numpy
import librosa
import librosa.display
import matplotlib.pyplot as plt
```

```
from scipy.spatial.distance import euclidean
import os
import numpy as np
def load_speaker_mfccs(data_dir):
    speakers = [d for d in os.listdir(data_dir) if os.path.isdir(os.path.join(data_dir, d))]
    speaker_model = {}

    for speaker in speakers:
        speaker_dir = os.path.join(data_dir, speaker)
        mfcc_features = []

        for root, _, files in os.walk(speaker_dir):
            for file_name in files:
                if file_name.endswith('.flac'):
                    file_path = os.path.join(root, file_name)
                    if os.path.exists(file_path):
                        try:
                            y, sr = librosa.load(file_path, sr=None)
                            mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
                            mfcc_features.append(mfccs.T)
                        except Exception as e:
                            print(f"Error processing file {file_path}: {e}")

        if mfcc_features:
            speaker_model[speaker] = np.vstack(mfcc_features)

    return speaker_model
```

```python
def prediction(test_files, speaker_model, true_speaker):
    correct_predictions = 0
    for file_path in test_files:
        predicted_speaker = identify_speaker(file_path, speaker_model)
        print(f"Predicted: {predicted_speaker}, True: {true_speaker}")
        if predicted_speaker == true_speaker:
            correct_predictions += 1

    accuracy = correct_predictions / len(test_files)
    return accuracy
```

```python
data_dir = "C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files"
speaker_model = load_speaker_mfccs(data_dir)

test1 = {
    'Raghav': ["C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Raghav\\Raghav15.flac",
               "C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Raghav\\Raghav24.flac"],
    'Nighil': ["C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Nighil\\Nighil9.flac",
               "C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Nighil\\Nighil8.flac"],
    'Sathya': ["C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Sathya\\Sathya55.flac",
               "C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Sathya\\Sathya71.flac"],
    'Sandeep': ["C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Sandeep\\Sandeep33.flac",
                "C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Sandeep\\Sandeep43.flac"],
    'Subrajith': ["C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Subrajith\\Subrajith33.flac",
                  "C:\\Users\\NIGHIL NATARAJAN\\Downloads\\DM-audio-files\\DM-audio-files\\Subrajith\\Subrajith43.flac"]
}
```

```python
def identify_speaker(test_file_path, speaker_model):
    try:
        y, sr = librosa.load(test_file_path, sr=None)
        test_mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13).T
        test_mean = np.mean(test_mfcc, axis=0)

        min_dist = float('inf')
        predicted_speaker = None

        for speaker, mfccs in speaker_model.items():
            speaker_mean = np.mean(mfccs, axis=0)
            dist = euclidean(test_mean, speaker_mean)

            if dist < min_dist:
                min_dist = dist
                predicted_speaker = speaker

        return predicted_speaker

    except Exception as e:
        print(f"Error identifying speaker: {e}")
        return None

# STEP 5: Evaluate accuracy per speaker
for speaker, files in test1.items():
    accuracy = prediction(files, speaker_model, speaker)
    print(f'Accuracy for {speaker}: {accuracy:.2f}\n')
```

PERFORMANCE EVALUATION

```
Predicted: Raghav, True: Raghav
Predicted: Raghav, True: Raghav
Accuracy for Raghav: 1.00

Predicted: Nighil, True: Nighil
Predicted: Nighil, True: Nighil
Accuracy for Nighil: 1.00

Predicted: Sathya, True: Sathya
Predicted: Sathya, True: Sathya
Accuracy for Sathya: 1.00

Predicted: Sathya, True: Sandeep
Predicted: Sathya, True: Sandeep
Accuracy for Sandeep: 0.00

Predicted: Sathya, True: Subrajith
Predicted: Subrajith, True: Subrajith
Accuracy for Subrajith: 0.50
```

Step-by-Step Evaluation Process

1. **Test Audio Input**:
   - Each test audio file is processed and features are extracted using **MFCC (Mel-Frequency Cepstral Coefficients)**.
2. **Speaker Prediction**:
   - The model computes the **Euclidean distance** between the test MFCCs and the stored **mean MFCCs** vector of each speaker in the training set.
   - The speaker with the **lowest distance** is selected as the **predicted speaker**.
3. **Ground Truth Comparison**:
   - The actual speaker's identity (true label) is obtained from the file name or directory structure.

- The predicted speaker label is compared to the true label.

4. **Accuracy Calculation**:
   - For each speaker, individual accuracy is calculated as:

$$\text{Accuracy for a speaker} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \times 100$$

   - The overall model accuracy is computed as:

$$\text{Overall Accuracy} = \frac{\text{Total Correct Predictions}}{\text{Total Test Files}} \times 100$$

5. **Output Display**:
   - The system prints the predicted and true labels for each test file along with accuracy per speaker, and a final overall accuracy score.

PERFORMANCE OF OUR MODEL

The speaker recognition model, which utilizes mean Mel-Frequency Cepstral Coefficient (MFCC) vectors and Euclidean distance for classification, achieved an overall accuracy of approximately 60% on the test dataset.

- The model demonstrated 100% accuracy for the speakers Raghav, Nighil, and Sathya, indicating that their vocal characteristics were distinct and consistently captured by the mean MFCC representation.
- In contrast, the model achieved 0% accuracy for Sandeep, as all of his test samples were misclassified as belonging to Sathya. This may be attributed to:
  - Similar vocal patterns between Sandeep and Sathya.
  - Limited or noisy training data for Sandeep.
  - Loss of important temporal features due to the use of mean MFCCs.

- For Subrajith, the model obtained 50% accuracy, which suggests partial recognition. The inconsistent performance may be due to variation in speech delivery, background noise, or insufficient differentiation in the feature space.
- Overall, the model is effective in recognizing speakers with clearly distinguishable voice characteristics but struggles in scenarios involving:
    - Overlapping acoustic features between individuals,
    - Inconsistent training data, and
    - Limitations inherent in using only averaged MFCCs for representation.

LIMITATIONS / CHALLENGES OF OUR MODEL

1. Uses Only Mean MFCCs

- The model averages MFCC features across multiple audio samples per speaker.
- While efficient, this **reduces the uniqueness** of individual voice traits, especially for speakers with varied speech patterns.

2. No Noise Handling or Denoising

- Background noise and recording artifacts directly affect MFCCs.
- There is **no preprocessing step** for noise removal, leading to lower accuracy on noisy samples.

3. Can't Recognize Unknown Speakers

- The model is trained only on known speakers.
- If an unknown voice is tested, it will **still assign it to the closest known speaker**, leading to misclassification.

4. Performance Depends on Dataset Size

- With a **small or imbalanced dataset**, the model may **overfit or underperform**.
- Accuracy varies significantly depending on the number of training samples per speaker.

5. No Real-Time or Live Input Support

- The current setup processes audio in batches from files.
- **Live recognition or streaming audio** isn't supported without major modifications.

6. Fixed Input Shape Requirement

- MFCCs must have consistent dimensions across all inputs.
- This requires **manual cropping or padding**, which can affect the representation quality.

7. Basic Distance Metric (Euclidean)

- Euclidean distance is used to compare speaker features.
- It may not be **robust enough** to distinguish speakers with similar voices or overlapping features.

8. Lacks Adaptability

- The model does not **learn incrementally**.
- Any new speaker or updated voice data requires full retraining.

CONCLUSION AND FUTURE WORKS

- The implemented speaker recognition system successfully identifies speakers based on their voice characteristics using **MFCC feature extraction** and **Euclidean distance** as the similarity measure.
- By processing and comparing voice samples, the model was able to achieve reliable results for a small dataset of locally recorded audio clips.
- Key findings include:
  - MFCCs are effective for capturing speaker-specific features.
  - The system performs well under controlled conditions with minimal background noise.
  - Simple distance-based classification is sufficient for a small-scale, low-complexity speaker recognition task.

FUTURE WORK

To enhance the accuracy, robustness, and scalability of the system, the following improvements are suggested:

- **Add Noise Handling & Silence Removal**: Incorporate noise reduction and silence trimming to improve feature quality.
- **Use Fixed-Length Audio Segments**: Normalize durations for better feature consistency across samples.
- **Increase Dataset Size**: Collect more samples from a diverse set of speakers to improve generalization.
- **Add GUI or Real-Time Support**: Build a simple user interface or real-time speaker identification system.
- **Explore Other Distance Metrics**: Try DTW (Dynamic Time Warping) or cosine similarity for possibly better results.