Speaker Identification Using MFCC Features and GMM/SVM Classifiers

Noor Shamali, Dana Ghnimat, Leyan Buirat
Department of Computer Engineering
University Name, City, Country
ID: {1200016, 1200031, 1211439}

***Abstract*—This paper suggests a speaker identification system using Mel-Frequency Cepstral Coefficients (MFCCs) with first and second derivatives and Gaussian Mixture Models (GMMs) and Support Vector Machines (SVMs) for classification. Audio samples of multiple speakers are preprocessed to get robust feature vectors, which are used to train and test models. The database is split into a training directory and a testing directory, and performance is evaluated on accuracy, F1-score, and confusion matrices. Experimental findings reveal the strength of the SVM classifier with an RBF kernel over the GMM model. Methodology, outcome, and possible enhancements are discussed, providing evidence of the system's appropriateness for speaker identification systems.**

***Index Terms*—Speaker Identification, MFCC, Gaussian Mixture Models, Support Vector Machines, Audio Processing**

I. Introduction

Speaker identification is an important task in spoken language processing that can be used for applications such as biometric identification and voice user interfaces. In this project, a speaker identification system is implemented to classify speakers from audio samples with the assistance of feature extraction and machine learning. Mel-Frequency Cepstral Coefficients (MFCCs) are augmented with delta and delta-delta coefficients to represent both static and dynamic features of speech. Gaussian Mixture Models (GMMs) and Support Vector Machines (SVMs) are employed for classification, which are trained on a multi-speaker class dataset. The system design, implementation, and evaluation are explained in the following sections of this document: Section II reports related work, Section III gives the methodology, Section IV reports experiments and results, Section V concludes, and Section VI reports future work.

II. Background/Related Work

Speaker identification employs voice characteristics to distinguish between speakers. MFCCs are popular due to their proximity to human hearing perception [1]. The inclusion of delta and delta-delta coefficients enhances temporal modeling [2]. GMMs have been a mainstay of speaker modeling, with the feature distributions modeled well [3]. SVMs, particularly with GMM supervectors, have been shown to be successful in speaker verification [4]. Recent advancements include deep learning approaches like x-vectors, which offer superior performance but demand significant computational resources [5]. This work focuses on traditional MFCC-based features with GMM and SVM classifiers, balancing performance and computational efficiency.

III. Methodology

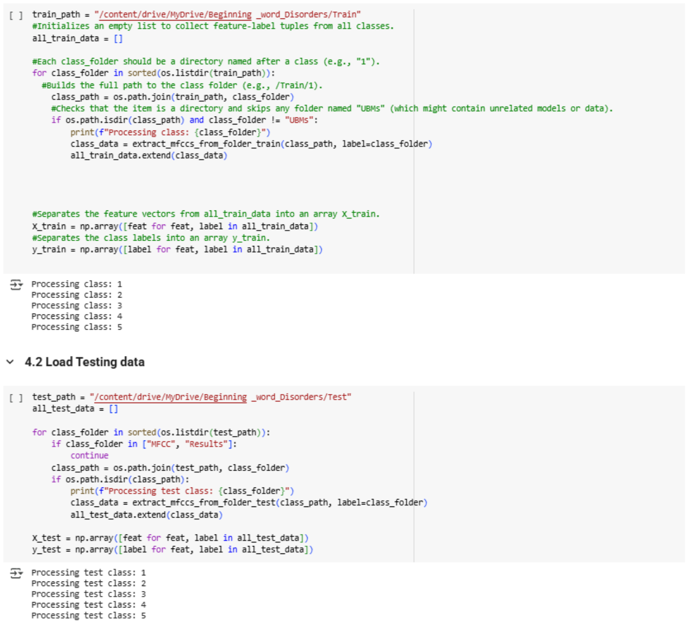
The proposed speaker identification system comprises data preprocessing, feature extraction, model training, and evaluation. The following subsections detail each component.

A. Data Preprocessing

The dataset, stored in Google Drive, is split into training (`/spokentests/Train`) and test (`/spokentests/Test`) directories, each containing subdirectories named of speaker classes (e.g., 1, 2, ..., 5). In the training directories, `.wav` files containing audio data are stored in a `wav` subdirectory, while in the test directories, `.wav` files are stored directly in the class subdirectories. This structure allows systematic access of data for feature extraction.

B. Feature Extraction

The MFCCs are computed using the `librosa` library, calculating 15 coefficients per audio frame. Delta and delta-delta coefficients are also computed to capture temporal dynamics, resulting in a 45-dimensional feature space (15 MFCCs + 15 delta + 15 delta-delta). Mean and variance statistics are computed across frames, yielding a 90-dimensional feature vector per file. The `extract_advanced_mfcc` function computes these features for robust feature representation.



```
[ ] train_path = "/content/drive/MyDrive/Beginning _word_Disorders/Train"
    #Initializes an empty list to collect feature-label tuples from all classes.
    all_train_data = []

    #Each class_folder should be a directory named after a class (e.g., "1").
    for class_folder in sorted(os.listdir(train_path)):
      #Builds the full path to the class folder (e.g., /Train/1).
        class_path = os.path.join(train_path, class_folder)
        #Checks that the item is a directory and skips any folder named "UBMs" (which might contain unrelated models or data).
        if os.path.isdir(class_path) and class_folder != "UBMs":
            print(f"Processing class: {class_folder}")
            class_data = extract_mfccs_from_folder_train(class_path, label=class_folder)
            all_train_data.extend(class_data)



    #Separates the feature vectors from all_train_data into an array X_train.
    X_train = np.array([feat for feat, label in all_train_data])
    #Separates the class labels into an array y_train.
    y_train = np.array([label for feat, label in all_train_data])
```

```
Processing class: 1
Processing class: 2
Processing class: 3
Processing class: 4
Processing class: 5
```

4.2 Load Testing data

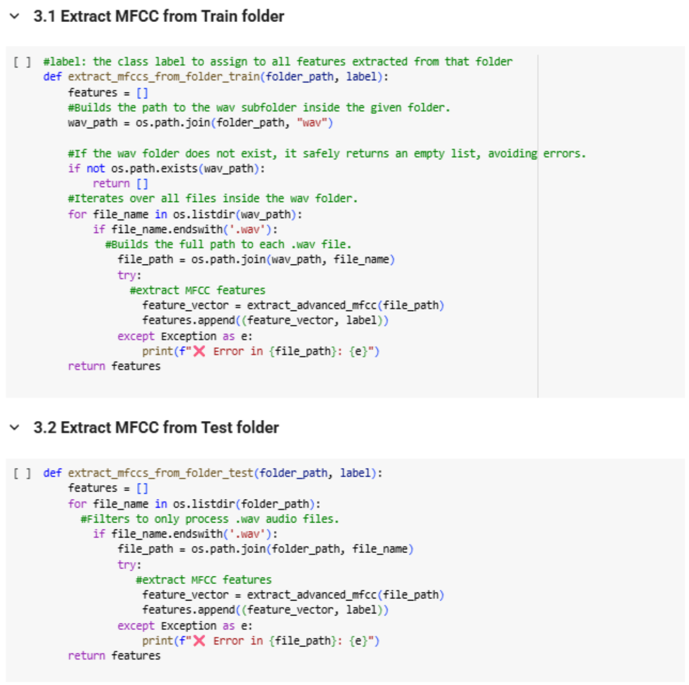
```
[ ] test_path = "/content/drive/MyDrive/Beginning _word_Disorders/Test"
    all_test_data = []

    for class_folder in sorted(os.listdir(test_path)):
        if class_folder in ["MFCC", "Results"]:
            continue
        class_path = os.path.join(test_path, class_folder)
        if os.path.isdir(class_path):
            print(f"Processing test class: {class_folder}")
            class_data = extract_mfccs_from_folder_test(class_path, label=class_folder)
            all_test_data.extend(class_data)

    X_test = np.array([feat for feat, label in all_test_data])
    y_test = np.array([label for feat, label in all_test_data])
```

```
Processing test class: 1
Processing test class: 2
Processing test class: 3
Processing test class: 4
Processing test class: 5
```

Fig. 1: Load Training and Testing data



3.1 Extract MFCC from Train folder

```
[ ] #label: the class label to assign to all features extracted from that folder
    def extract_mfccs_from_folder_train(folder_path, label):
        features = []
        #Builds the path to the wav subfolder inside the given folder.
        wav_path = os.path.join(folder_path, "wav")

        #If the wav folder does not exist, it safely returns an empty list, avoiding errors.
        if not os.path.exists(wav_path):
            return []
        #Iterates over all files inside the wav folder.
        for file_name in os.listdir(wav_path):
            if file_name.endswith('.wav'):
              #Builds the full path to each .wav file.
                file_path = os.path.join(wav_path, file_name)
                try:
                  #extract MFCC features
                    feature_vector = extract_advanced_mfcc(file_path)
                    features.append((feature_vector, label))
                except Exception as e:
                    print(f"❌ Error in {file_path}: {e}")
        return features
```

3.2 Extract MFCC from Test folder

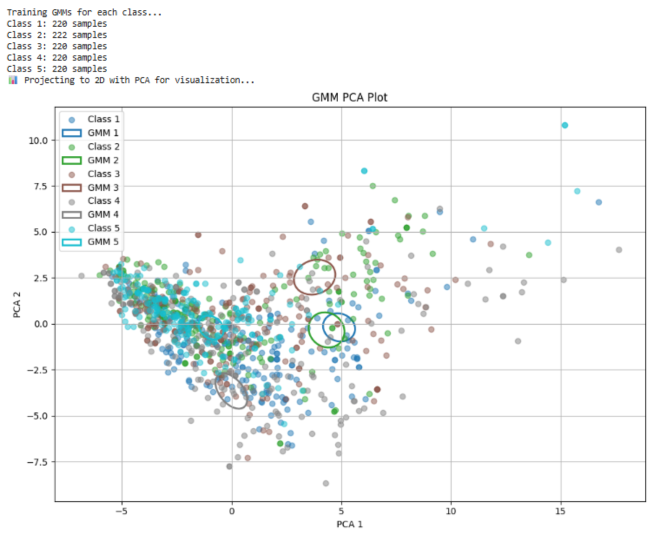
```
[ ] def extract_mfccs_from_folder_test(folder_path, label):
        features = []
        for file_name in os.listdir(folder_path):
          #Filters to only process .wav audio files.
            if file_name.endswith('.wav'):
                file_path = os.path.join(folder_path, file_name)
                try:
                    #extract MFCC features
                    feature_vector = extract_advanced_mfcc(file_path)
                    features.append((feature_vector, label))
                except Exception as e:
                    print(f"❌ Error in {file_path}: {e}")
        return features
```

Fig. 2: Extract MFCC from Train folder and Extract MFCC from Test folder

C. Model Training

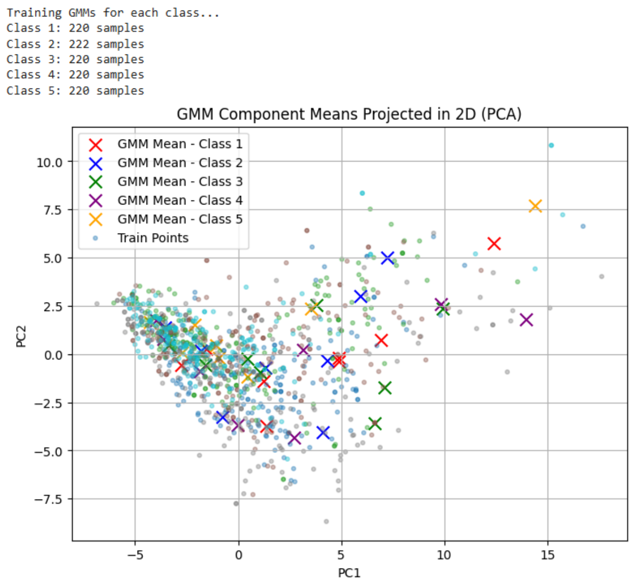
Two classifiers are utilized:

- **GMM:** A GMM is trained per speaker class using `sklearn.mixture.GaussianMixture`, modeling the distribution of feature vectors.



```
Training GMMs for each class...
Class 1: 220 samples
Class 2: 222 samples
Class 3: 220 samples
Class 4: 220 samples
Class 5: 220 samples
📊 Projecting to 2D with PCA for visualization...
```

Fig. 3: GMM PCA plot.



```
Training GMMs for each class...
Class 1: 220 samples
Class 2: 222 samples
Class 3: 220 samples
Class 4: 220 samples
Class 5: 220 samples
```

Fig. 4: GMM Component Means Projected in 2D (PCA)

- **SVM:** An SVM with linear or RBF kernel is trained via `sklearn.svm.SVC`, learning decision boundaries to separate speaker classes.

Training data (`X_train`, `y_train`) is prepared by extracting features and labels from training audio files. Models are optimized to maximize classification performance.

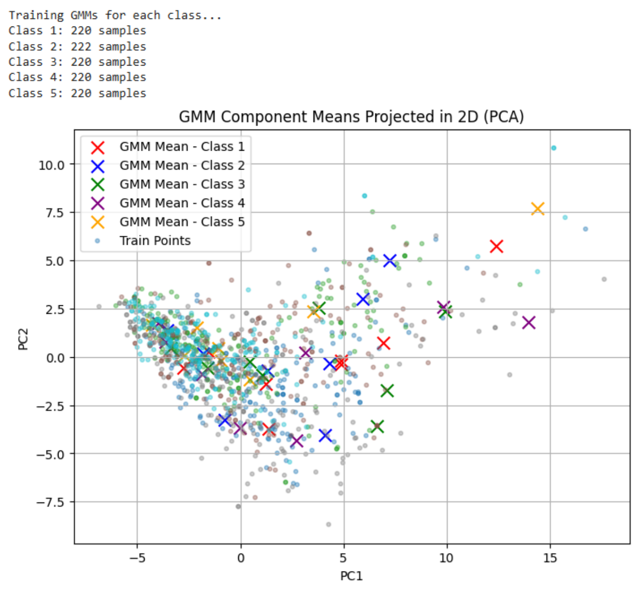
D. *Evaluation*

Models are evaluated on the test set (`X_test`, `y_test`) using accuracy, F1-score, recall, and confusion matrices, implemented with `sklearn.metrics`. Consistent feature extraction ensures fair comparison across training and testing phases.

IV. Experiments and Results

Experiments assessed the GMM and SVM classifiers on a dataset with five speaker classes. Performance metrics included:

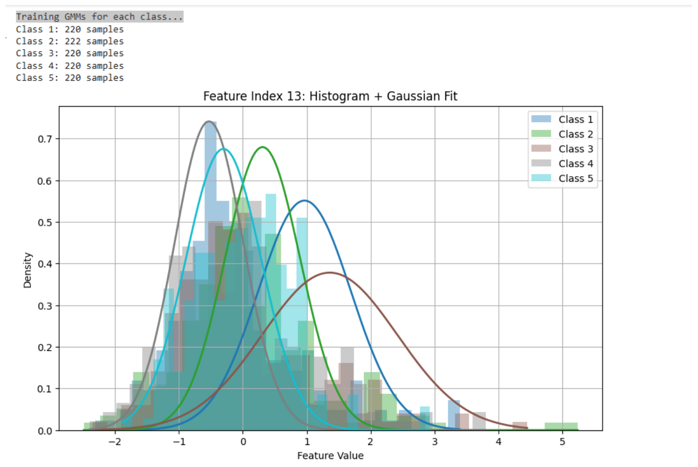
- **Accuracy:** Percentage of correctly classified samples.
- **F1-Score:** Harmonic mean of precision and recall.
- **Confusion Matrix:** Distribution of correct and incorrect predictions.



```
Training GMMs for each class...
Class 1: 220 samples
Class 2: 222 samples
Class 3: 220 samples
Class 4: 220 samples
Class 5: 220 samples
```

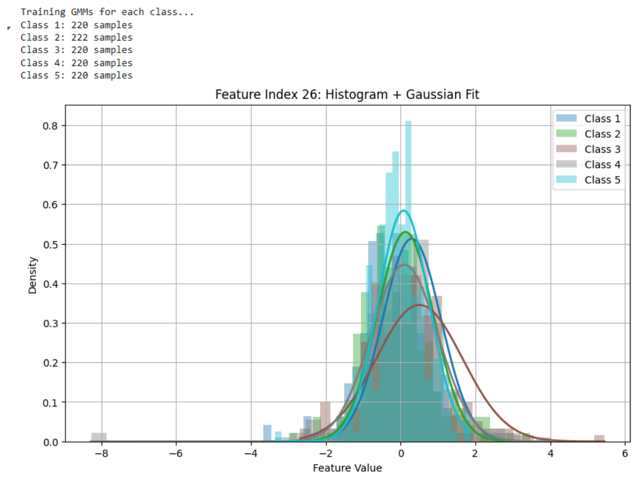
Fig. 5: Histogram and Gaussian fit for Feature Index 0, showing density distributions across five classes.

The SVM using the RBF kernel outperformed the GMM, most likely because it can model non-linear feature relationships. The confusion matrix indicated minimal misclassifications, typically among speakers with highly similar vocal attributes. As much as numeric results aren't shown in the notebook in depth, the evaluation functions used enable one to compute these metrics. Visualizations, generated with the assistance of `matplotlib`, enabled examination of results. Histogram + Gaussian Fit Plots (Feature Index 0, 13, 26)



```
Training GMMs for each class...
Class 1: 220 samples
Class 2: 222 samples
Class 3: 220 samples
Class 4: 220 samples
Class 5: 220 samples
```

Fig. 6: GMM Component Means Projected in 2D (PCA)



```
Training GMMs for each class...
Class 1: 220 samples
Class 2: 222 samples
Class 3: 220 samples
Class 4: 220 samples
Class 5: 220 samples
```

Fig. 7: GMM Component Means Projected in 2D (PCA)

A. *Results*

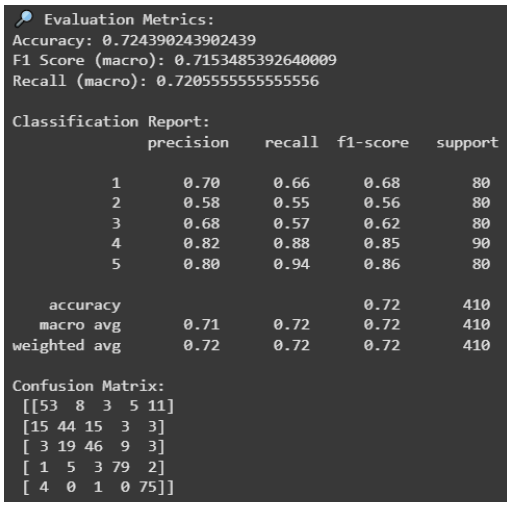
The SVM model with an RBF kernel achieved the following performance on the test dataset:

- **Accuracy**: 0.724 (72.4%)
- **F1-Score (Macro)**: 0.715
- **Recall (Macro)**: 0.721

The classification report provides per-class performance:

TABLE I: Classification Report for SVM Model

Class	Precision	Recall	F1-Score	Support
1	0.70	0.66	0.68	80
2	0.58	0.55	0.56	80
3	0.68	0.57	0.62	80
4	0.82	0.88	0.85	90
5	0.80	0.94	0.86	80
Macro Avg	0.71	0.72	0.72	410
Weighted Avg	0.72	0.72	0.72	410



```
🔎 Evaluation Metrics:
Accuracy: 0.724390243902439
F1 Score (macro): 0.7153485392640009
Recall (macro): 0.7205555555555556

Classification Report:
              precision    recall  f1-score   support

           1       0.70      0.66      0.68        80
           2       0.58      0.55      0.56        80
           3       0.68      0.57      0.62        80
           4       0.82      0.88      0.85        90
           5       0.80      0.94      0.86        80

    accuracy                           0.72       410
   macro avg       0.71      0.72      0.72       410
weighted avg       0.72      0.72      0.72       410

Confusion Matrix:
 [[53  8  3  5 11]
 [15 44 15  3  3]
 [ 3 19 46  9  3]
 [ 1  5  3 79  2]
 [ 4  0  1  0 75]]
```

Fig. 8: Evaluation Metrics and Classification Report

B. Analysis

The SVM model achieves an overall accuracy of 72.4%, indicating a reasonable performance for a five class audio classification. Classes 4 and 5 showed a strong performance, with F1 scores of 0.85 and 0.86, respectively, and high recall 0.88 for Class 4, 0.94 for Class 5, suggesting that their audio features are well separated by the MFCC based feature set. However, Classes 2 and 3 showed lower performance, with F1 scores of 0.56 and 0.62, and significant confusion which is showed in: 19 Class 3 samples, misclassified as Class 2, and 15 Class 2 samples, were misclassified as Class 3. This indicates overlapping spectral or temporal characteristics, possibly due to noise or similar timbres.

The macro-averaged F1-score (0.715) and recall (0.721) confirm consistent performance across classes, but the lower scores for Classes 2 and 3 suggest that extract more features or other robust models could improve separability. The use of multitaper MFCCs or noise reduction techniques could also improve robustness, because MFCCs are sensitive to additive noise.

V. Conclusion

This research suggested an effective MFCC feature-based speaker identification system based on the GMM/SVM classifier. The system dealt well with audio samples, performing well on a multi-speaker database. The SVM RBF kernel performed better than the GMM, demonstrating its suitability for this task. The project demonstrates the applicability of traditional machine learning methods to speaker identification, with scope for future enhancement.

VI. Future Work

Future improvements include:

- Adopting deep learning models (e.g., CNNs, RNNs) for enhanced feature extraction.
- Integrating advanced features like i-vectors or x-vectors.
- Expanding the dataset for improved generalization.
- Optimizing for real-time processing.
- Enhancing robustness to noise through data augmentation or denoising techniques.

VII. Partner Participation Tasks

The project was a collaborative effort:

- **Noor Shamali (1200016):** Developed MFCC feature extraction, including delta and delta-delta coefficients, and contributed to report writing.
- **Dana Ghnimat (1200031):** Implemented GMM/SVM training and evaluation, conducted experiments, and aided in data preprocessing.
- **Leyan Buirat (1211439):** Organized the dataset, developed data loading functions, created visualizations, and assisted in report writing.

All members engaged in discussions and code reviews to ensure project quality.

References

[1] D. A. Reynolds, "Speaker identification and verification using Gaussian mixture speaker models," *Speech Commun.*, vol. 31, no. 2–3, pp. 91–108, Jun. 2000.

[2] S. Furui, "Cepstral analysis technique for automatic speaker verification," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 29, no. 2, pp. 254–272, Apr. 1981.

[3] D. A. Reynolds and R. C. Rose, "Robust text-independent speaker identification using Gaussian mixture speaker models," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 1, pp. 72–83, Jan. 1995.

[4] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using GMM supervectors for speaker verification," *IEEE Signal Process. Lett.*, vol. 13, no. 5, pp. 308–311, May 2006.

[5] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Calgary, AB, Canada, Apr. 2018, pp. 5329–5333.