Machine Learning
UE22CS352A

Description:

In the psychological well-being of humans, emotions are pertinent. It serves as an agent for communicating one's viewpoint or mental condition to others. Analysing audio data involves understanding the nature of sound, extracting meaningful features, and then applying machine learning techniques. Speech Emotion Recognition (SER) is one of the applications of audio analysis, which is used to detect the emotion from the speech signal.

Speech Emotion Recognition, abbreviated as SER, is the act of attempting to recognize human emotion and affective states from speech. This is capitalizing on the fact that voice often reflects underlying emotion through tone and pitch

In this task, basic emotions like calm, happy, neutral, sad, angry, fearful, disgust and surprised are to be analysed from emotional speech signals. You must use a Machine Learning model i.e a Multilayer perceptron Classifier (MLP Classifier) to categorize the given data into respective groups. For achieving this objective, we use a python library called `librosa` to analyse the audio files.

A basic workflow will be provided to help you with this task, but you are free to use alternative approaches if preferred.

About the dataset:

This RAVDESS dataset contains 1440 files: 60 trials per actor x 24 actors = 1440. The RAVDESS contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech emotions includes calm, happy, sad, angry, fearful, surprise, and disgust expressions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

Each of the 1440 files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 03-01-06-01-02-01-12.wav). These identifiers define the stimulus characteristics:

Filename identifiers:

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

Filename example: 03-01-06-01-02-01-12.wav

1. Audio-only (03)
2. Speech (01)
3. Fearful (06)
4. Normal intensity (01)
5. Statement "dogs" (02)
6. 1st Repetition (01)
7. 12th Actor (12)
   Female, as the actor ID number is even.

An overview of the attributes to be considered while building the model:

Below is a brief description about the attributes that will be used for audio signal processing and feature extraction for our Speech Emotion Recognition (SER) task.

MFCC (Mel-Frequency Cepstral Coefficient):

- MFCCs are a representation of the short-term power spectrum of a sound, often used in speech and music processing. They are derived from the mel scale, which mimics the way humans perceive sound. MFCCs capture the timbral aspects of the audio, focusing on frequency patterns that are relevant to how we perceive sounds.

Chromagram:

- A chromagram represents the intensity of different pitch classes (or musical notes) across time, disregarding octave differences. It is widely used in music processing to analyze harmonic content and chord structures. In speech, it can capture tonal characteristics.

Mel-scaled Spectrogram:

- This is a spectrogram that uses a mel scale for the frequency axis. The mel scale is designed to be perceptually linear for human ears. A mel spectrogram shows how energy at various frequencies evolves over time, making it useful for tasks that involve auditory perception, such as SER.

Spectral Contrast:

- Spectral contrast refers to the difference in amplitude between peaks and valleys in a sound spectrum. It highlights the variation in different frequency bands and is useful for distinguishing between sounds with different timbres, which can be important in emotion detection.

Tonnetz:

- Tonnetz (Tonal Centroid Features) represents the harmonic relationship between pitches in a sound. It's often used in music analysis to represent the relationships between notes and chords, but it can also be applied in speech to analyze tonal qualities.

Note: While doing this task, you might have to explore a little more about these features apart from what has been given in this document.

The following outlines the workflow:

1. Installing and Importing Required Libraries

Begin by installing and importing the necessary libraries essential for audio processing and machine learning tasks:

- Librosa: A library for audio analysis that provides tools for feature extraction.
- NumPy: A fundamental package for numerical computations in Python, enabling efficient handling of arrays and matrices.
- Matplotlib: A plotting library used for creating static, animated, and interactive visualizations in Python.
- Scikit-learn: A comprehensive machine learning library that provides various algorithms, tools for model evaluation, and data preprocessing utilities.

2. Defining Emotions

Create a dictionary that maps emotion codes to their corresponding descriptive names. This mapping serves to clarify the labels used in the dataset, facilitating better understanding and interpretation of the model outputs.

3. Feature Extraction

Utilize functions from the Librosa library to load the audio dataset and extract relevant features.

4. Balancing the Dataset

To ensure fairness and avoid bias in model training, balance the dataset by following these steps:

- Count Samples: Determine the number of samples available for each emotion in the dataset.
- Determine Minimum Samples: Identify the emotion class with the fewest samples.
- Select Samples: Ensure that each emotion class has an equal number of samples, which may involve under sampling the larger classes or oversampling the smaller ones.

5. Splitting the Data

Divide the dataset into training and testing subsets, allocating 80% of the data for training the model and 20% for testing its performance. This split allows for effective model evaluation.

6. Feature Visualization

Visualize the extracted features to understand their distribution. This can be done using plots generated by Matplotlib.

Hint: Check the indices in the feature arrays:

- MFCC has 13 features.
- Chromogram has 12 features.
- Mel-scaled spectrogram has 100 features (the number of mel bands can vary, you can assume it as 100 for this task).
- Spectral contrast has 7 features.
- Tonnetz has 6 features.

7. Encoding Labels and Standardizing Features

Use LabelEncoder from Scikit-learn to convert the emotion labels into a numerical format suitable for model training. Additionally, standardize the features using StandardScaler to ensure they are on the same scale, which can improve model performance.

8. Building the MLP Model

Construct a Multi-Layer Perceptron (MLP) model using MLPClassifier from Scikit-learn. Train the model with various parameters and evaluate its performance on the test set.

9. Fine-Tuning Hyperparameters

Optimize the model's hyperparameters using RandomizedSearchCV to explore a range of parameter values and select the best performing combination.

10. Making Predictions

Utilize both the basic model and the optimized model to make predictions on the test dataset.

11. Analysing Different Metrics

Evaluate the performance of both models by analysing various metrics, including precision, recall, F1 score, and confusion matrix.

12. Implementing K-Fold Cross-Validation Techniques

Enhance model evaluation by employing K-fold cross-validation, which helps in assessing the model's performance across different subsets of the data.

13. Performing Comparative Study with Respect to SVM and KNN

Conduct a comparative analysis of the SER model's performance against other classifiers, such as Support Vector Machine (SVM) and K-Nearest Neighbors (KNN). Summarize the findings and provide suitable inferences based on model performance.