LAHORE GARRISON UNIVERSITY

NAME: ADNAN NAEEM & ALI RAZA KHAN

ROLL NO: SP21/BSCS/079-B

SP21/BSCS/094-B

SUBJECT: MACHINE LEARNING (ML)

SUBMITTED TO: DR. SADAF HUSSAIN

DATE: 03/02/2024

Speech Emotion Recognition (SER) using Machine Learning

Problem Domain:

Speech Emotion Recognition (SER) is a fascinating area within the field of machine learning where the objective is to develop models that can identify the emotional state of a person based on their speech patterns. The project focuses on building a Speech Emotion Recognition system using a dataset of audio recordings. The dataset is the RAVDESS (Ryerson Audio-Visual Database of Emotional Speech and Song) dataset, which includes recordings of different 24 actors expressing various emotions.

1) Objective:

- Build a Speech Emotion Recognition system using machine learning techniques.
- Evaluate the model's performance on a given dataset.
- Visualize dataset characteristics and analyze the simulation results.

2) Dataset:

- Use the RAVDESS dataset for training and testing.
- The dataset consists of audio recordings labeled with eight different emotions.

3) Methodology:

a) Feature Extraction:

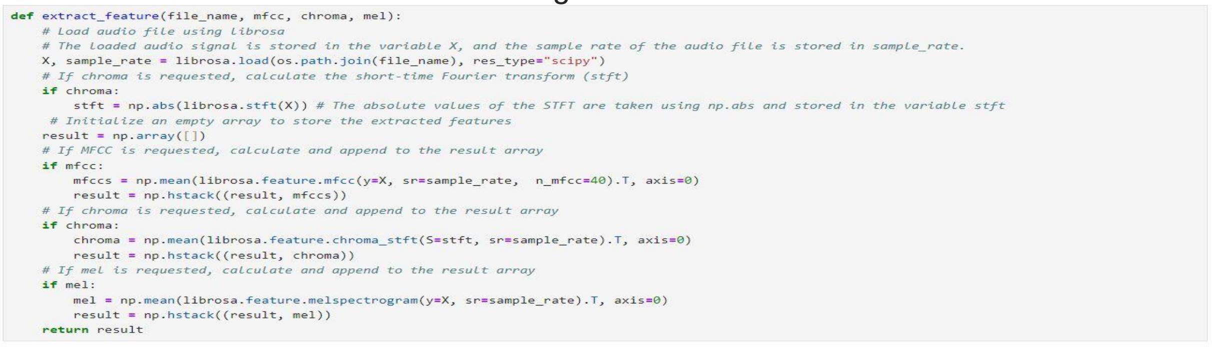
- Extract relevant features from audio files, including MFCCs (Mel-Frequency Cepstral Coefficients), Chroma, and Mel Spectrogram.
- Utilize the librosa library for audio processing**.**

b) Model Selection:

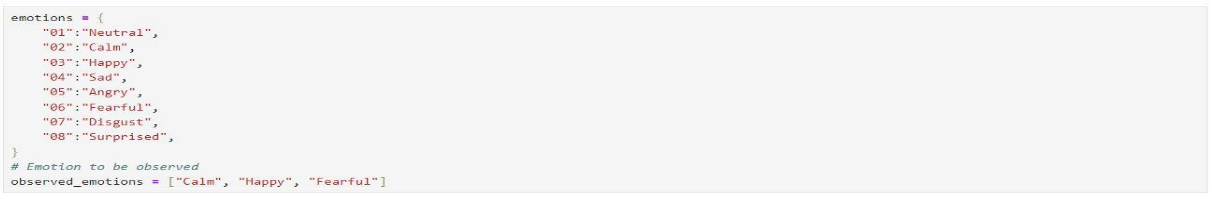
- Choose a suitable machine learning model for Speech Emotion Recognition. In this case, a Multilayer Perceptron (MLP) classifier has been implemented**.**

c) Training and Testing:

- Split the dataset into training and testing sets.
- Train the selected model on the training set.
- Evaluate the model on the testing set.



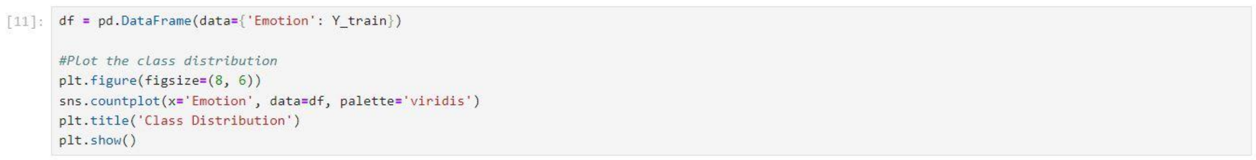
```python
def extract_feature(file_name, mfcc, chroma, mel):
    # Load audio file using librosa
    # The loaded audio signal is stored in the variable X, and the sample rate of the audio file is stored in sample_rate.
    X, sample_rate = librosa.load(os.path.join(file_name), res_type="scipy")
    # If chroma is requested, calculate the short-time Fourier transform (stft)
    if chroma:
        stft = np.abs(librosa.stft(X)) # The absolute values of the STFT are taken using np.abs and stored in the variable stft
     # Initialize an empty array to store the extracted features
    result = np.array([])
    # If MFCC is requested, calculate and append to the result array
    if mfcc:
        mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate,  n_mfcc=40).T, axis=0)
        result = np.hstack((result, mfccs))
    # If chroma is requested, calculate and append to the result array
    if chroma:
        chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T, axis=0)
        result = np.hstack((result, chroma))
    # If mel is requested, calculate and append to the result array
    if mel:
        mel = np.mean(librosa.feature.melspectrogram(y=X, sr=sample_rate).T, axis=0)
        result = np.hstack((result, mel))
    return result
```



```python
emotions = {
    "01":"Neutral",
    "02":"Calm",
    "03":"Happy",
    "04":"Sad",
    "05":"Angry",
    "06":"Fearful",
    "07":"Disgust",
    "08":"Surprised",
}
# Emotion to be observed
observed_emotions = ["Calm", "Happy", "Fearful"]
```

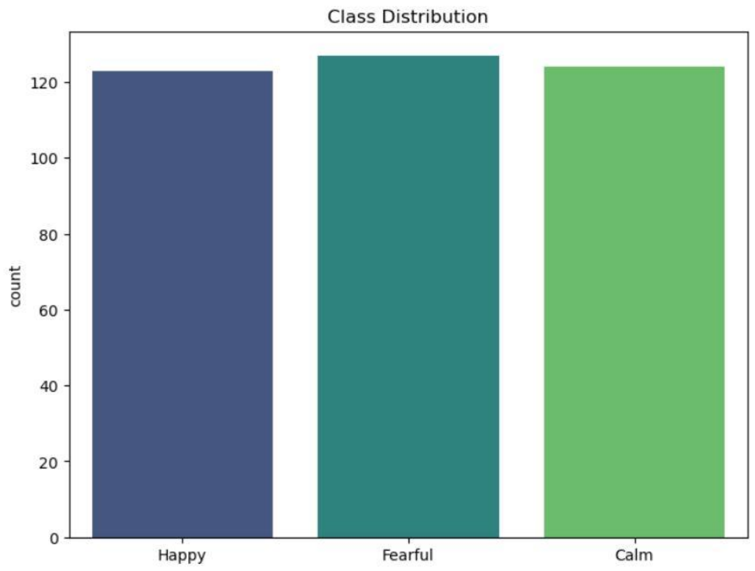
4) <u>Visualization</u>:

- Visualize the shape of training and testing datasets.
- Plot the class distribution of emotions in the training dataset.



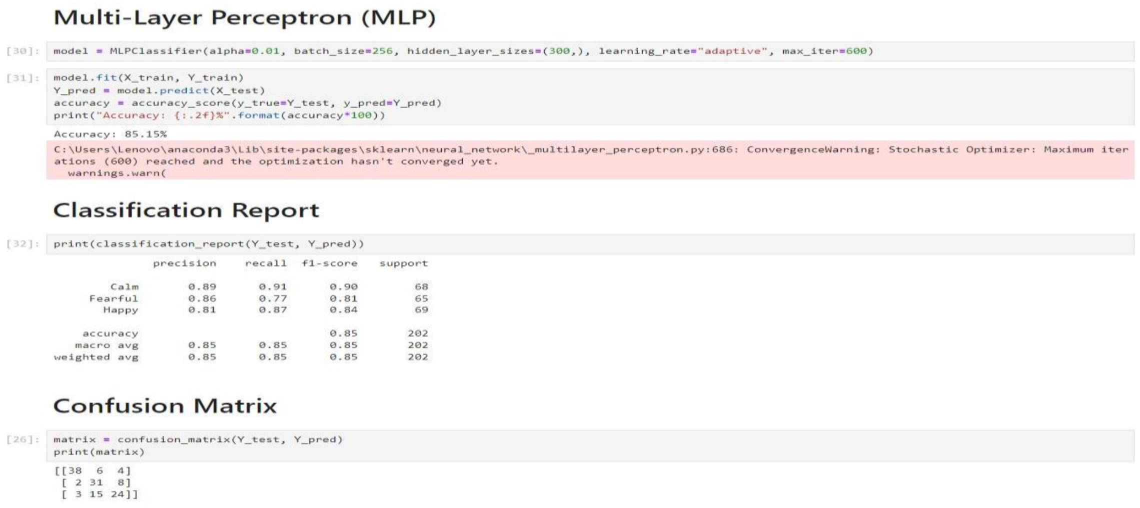
```
[11]: df = pd.DataFrame(data={'Emotion': Y_train})

      #Plot the class distribution
      plt.figure(figsize=(8, 6))
      sns.countplot(x='Emotion', data=df, palette='viridis')
      plt.title('Class Distribution')
      plt.show()
```



5) <u>Results and Analysis</u>:

- Provide simulation results, including accuracy, classification report, and confusion matrix.
- Discuss the performance of the model and its ability to recognize different emotions.



Multi-Layer Perceptron (MLP)

```
[30]: model = MLPClassifier(alpha=0.01, batch_size=256, hidden_layer_sizes=(300,), learning_rate="adaptive", max_iter=600)

[31]: model.fit(X_train, Y_train)
      Y_pred = model.predict(X_test)
      accuracy = accuracy_score(y_true=Y_test, y_pred=Y_pred)
      print("Accuracy: {:.2f}%".format(accuracy*100))

      Accuracy: 85.15%
      C:\Users\Lenovo\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic Optimizer: Maximum iter
      ations (600) reached and the optimization hasn't converged yet.
        warnings.warn(
```

Classification Report

```
[32]: print(classification_report(Y_test, Y_pred))

                    precision    recall  f1-score   support

              Calm       0.89      0.91      0.90        68
           Fearful       0.86      0.77      0.81        65
             Happy       0.81      0.87      0.84        69

          accuracy                           0.85       202
         macro avg       0.85      0.85      0.85       202
      weighted avg       0.85      0.85      0.85       202
```

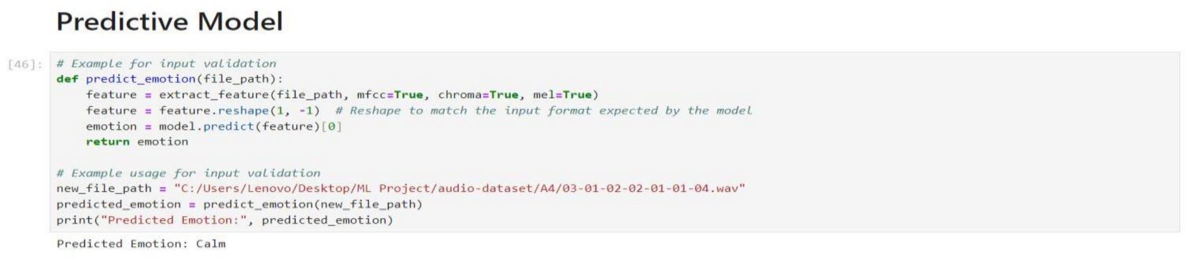
Confusion Matrix

```
[26]: matrix = confusion_matrix(Y_test, Y_pred)
      print(matrix)

      [[38  6  4]
       [ 2 31  8]
       [ 3 15 24]]
```

6) <u>Input Validation</u>:

- Include an example for input validation using a new audio file to predict the emotion.



Predictive Model

```
[46]: # Example for input validation
      def predict_emotion(file_path):
          feature = extract_feature(file_path, mfcc=True, chroma=True, mel=True)
          feature = feature.reshape(1, -1)  # Reshape to match the input format expected by the model
          emotion = model.predict(feature)[0]
          return emotion

      # Example usage for input validation
      new_file_path = "C:/Users/Lenovo/Desktop/ML Project/audio-dataset/A4/03-01-02-02-01-01-04.wav"
      predicted_emotion = predict_emotion(new_file_path)
      print("Predicted Emotion:", predicted_emotion)

      Predicted Emotion: Calm
```