CS328 Assignment 4: Emotion recognition in speech

In this assignment, you will implement two core functions in a program for emotion recognition from audio data. The audio data files are .wav files, and they each have an associated emotion label. You will use the librosa library to extract features from each audio file, and then use the scikit-learn library to train a random forest classifier on these features. The goal of the classifier is to predict the emotion associated with each audio file.

In [5]:

```
# Import necessary libraries and packages.
import os
import pandas as pd
import numpy as np
import sys
import librosa
import re
import glob
import librosa.display
from IPython.display import Audio

import plotly.express as px
from sklearn import tree, metrics
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, ConfusionMatrixDisplay
from scipy.signal import butter, filtfilt, find_peaks
from sklearn.tree import DecisionTreeClassifier,export_graphviz
from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

Tasks

Function: extract_features(data)

Function to extract audio features from a given signal data.

In this function, you will

- extract four types of audio features: zero-crossing rate (ZCR), Mel-frequency cepstral coefficients (MFCC), root-mean-square (RMS), and Mel spectrogram.
- For each type of feature, compute the feature and then average across all frames in the audio signal.
- Concatenate the averaged features into a single row.
- Return it as a pandas DataFrame containing the feature vector.

Input: An array, data, that contains the audio signal.

Output: A DataFrame containing the averaged features. The shape of the DataFrame should be 1x150.

Function: train_random_forest(frames)

Function to train a random forest classifier given a DataFrame of features and labels.

In this function, you will train a random forest classifier on the features and labels.

- Use the first 150 columns of the DataFrame as the features (X), and the last column as the labels (y).
- Split the data into training and test sets (with a test size of 0.3 and random_state of 42 - note that these values have to be *exact* for our autograder).
- Train a RandomForestClassifier from scikit-learn on the training set.
- Evaluate the model on the test set, and print the classification report and confusion matrix.
- Return the trained model, the confusion matrix, and the test accuracy.

Input: A DataFrame, frames, where the first 150 columns are features and the last column is the label.

Output: The trained RandomForestClassifier model, the confusion matrix, and the test accuracy.

In [2]:

```
def extract_features(data):
    # Extract features
    zcr = librosa.feature.zero_crossing_rate(y=data)
    mfcc = librosa.feature.mfcc(y=data, sr=sample_rate)
    rms = librosa.feature.rms(y=data)
    mel = librosa.feature.melspectrogram(y=data, sr=sample_rate)

    # Average across columns (axis=1)
    zcr_avg = np.mean(zcr, axis=1)
    mfcc_avg = np.mean(mfcc, axis=1)
    rms_avg = np.mean(rms, axis=1)
    mel_avg = np.mean(mel, axis=1)

    # Concatenate into single row
    features = np.concatenate([zcr_avg, mfcc_avg, rms_avg, mel_avg])

    # Convert to dataframe and transpose it so that the shape is 1x150
    df = pd.DataFrame(features).T

    return df
```

In [3]:

```
def train_random_forest(frames):

    # Use pandas iloc fn to extract the first 150 columns as features.
    # Careful about how the indexing works (cols start from 0)
    X = frames.iloc[: , 0:150]

    # Use pandas iloc function to extract the 151st column as the prediction target.
    # Again, careful about how indexing works (col numbers start from 0)
    y = frames.iloc[: , 150]

    # Split data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

    Emotion_rf = RandomForestClassifier()
    Emotion_rf_model = Emotion_rf.fit(X_train, y_train)
    Emotion_rf_pred = Emotion_rf_model.predict(X_test)
    print(classification_report(y_test, Emotion_rf_pred))

    # Evaluate on test set
    acc = Emotion_rf_model.score(X_test, y_test)

    Emotion_rf_cm = confusion_matrix(y_test, Emotion_rf_pred)
    print(Emotion_rf_cm)

    return Emotion_rf_model, Emotion_rf_cm, acc
```

Provided Code: Main Script

The main part of the script does the following:

- Collects all .wav file paths and creating an empty DataFrame, frames.
- For each file, it extracts the associated emotion from the filename, loads the audio data, extracts features using the extract_features function that you will write, and creates a DataFrame row combining these features and the emotion label.
- This row is then appended to the frames DataFrame.
- After processing all files, the column names for frames are set.
- Finally, the train_random_forest function that you will write is called with frames as input to train the random forest classifier and evaluate its performance.

In [4]:

```python
filenames = glob.glob("data/Emotion/*/*.wav")
frames = pd.DataFrame()

for filename in filenames:
    # Extract the SAMPLE_RATE from the filename
    emotion = re.search(r'Emotion/(\w*)/', filename).group(1)

    # duration and offset are used to take care of the no audio in start and the ending of each auc
    data, sample_rate = librosa.load(filename, duration=2.5, offset=0.6)

    feature_df = extract_features(data)
    emotion_df = pd.DataFrame([emotion])

    # Assuming feature_df only has one row, you can directly concatenate along the columns
    combined_df = pd.concat([feature_df, emotion_df], axis=1)
    frames = frames.append(combined_df, ignore_index=True)

# Create column names
col_names = [f'feat_{i}' for i in range(150)] + ['label']
frames.columns = col_names

Emotion_rf_model, Emotion_rf_cm, acc = train_random_forest(frames)
```

```
              precision    recall  f1-score   support

       angry       0.62      0.56      0.59        18
     disgust       0.59      0.42      0.49        24
        fear       0.53      0.64      0.58        14
       happy       0.52      0.46      0.49        24
     neutral       0.67      0.96      0.79        27
         sad       0.94      0.73      0.82        22
    surprise       0.41      0.47      0.44        15

    accuracy                           0.62       144
   macro avg       0.61      0.60      0.60       144
weighted avg       0.63      0.62      0.61       144

[[10  3  0  5  0  0  0]
 [ 2 10  4  0  7  0  1]
 [ 0  0  9  2  1  0  2]
 [ 3  3  0 11  0  0  7]
 [ 0  0  0  0 26  1  0]
 [ 1  0  0  0  5 16  0]
 [ 0  1  4  3  0  0  7]]
```

In [ ]: