MACHİNE LEARNİNG FOR SİGNAL PROCESSİNG – EHB-328E

PROJECT PROGRESS REPORT
SPEECH EMOTİON RECOGNITION

Prof. Dr. Ender Mete Ekşioğlu

Osman Kaan Kurtça

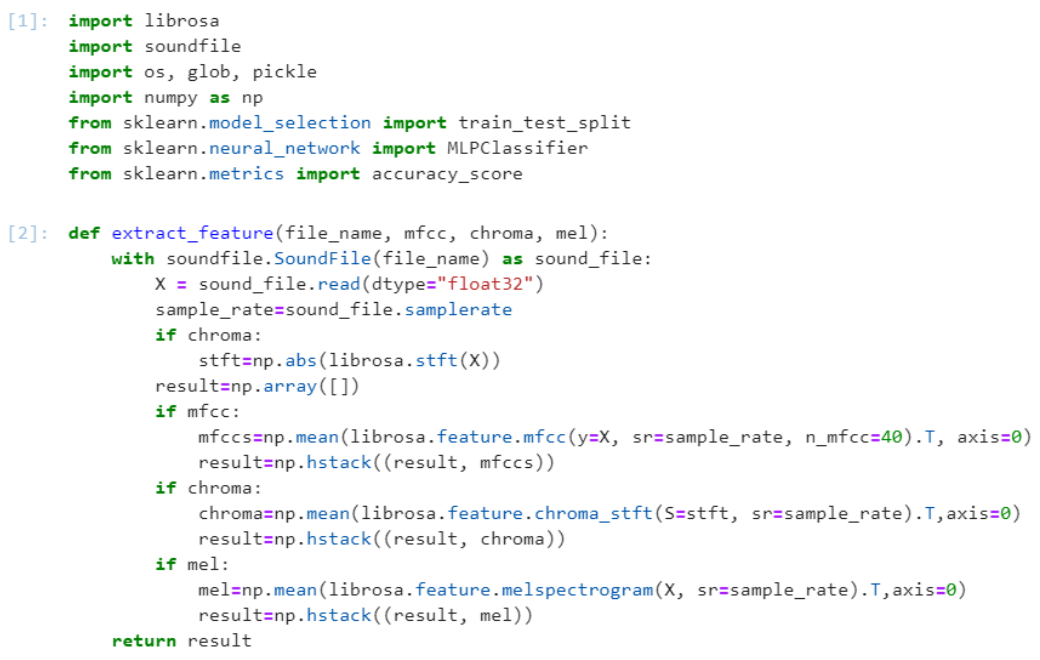
040160090

Aziz Gündoğdu

040160112

Speech Emotion Recognition Project Progress Report

We did some research and looked at some solutions that were made. We searched the datasets we will use for our model. SAVEE, RAVDESS, TESS and CREMA datasets include human speech audio files with various emotions. For our project, we plan to combine all of these datasets. At this stage of our project, we just used one of them and made it ready for training. We have handled the RAVDESS dataset and converted the audio files into a matrix by extracting features of audio files. While each row of the matrix represents a different human speech, each column except the last one represents a feature. The last column represents the class of the data.



```
[1]: import librosa
     import soundfile
     import os, glob, pickle
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.neural_network import MLPClassifier
     from sklearn.metrics import accuracy_score

[2]: def extract_feature(file_name, mfcc, chroma, mel):
         with soundfile.SoundFile(file_name) as sound_file:
             X = sound_file.read(dtype="float32")
             sample_rate=sound_file.samplerate
             if chroma:
                 stft=np.abs(librosa.stft(X))
             result=np.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:
                 chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
                 result=np.hstack((result, chroma))
             if mel:
                 mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
                 result=np.hstack((result, mel))
         return result
```

Firstly, we import necessary libraries. After that, we define a function that extract 3 features of sound file. Features are mfcc, chroma and mel.

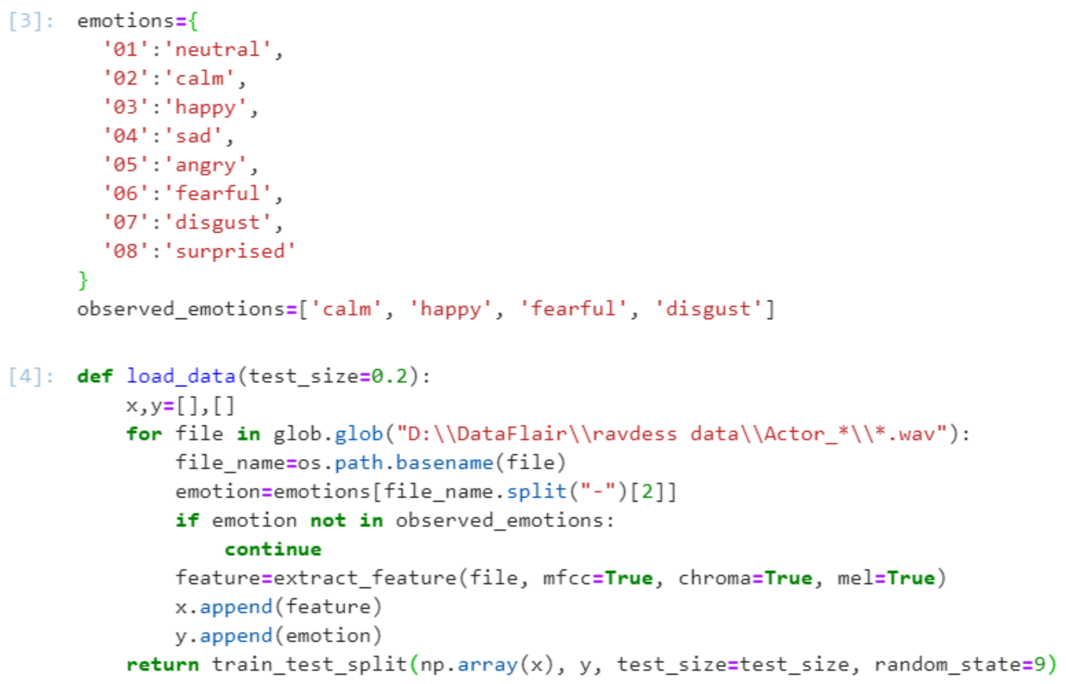
MFCC: Mel Frequency Cepstral Coefficient, represents the short-term power spectrum of a sound

MFCCs are commonly derived as follows

1. Take the Fourier Transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, triangular overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

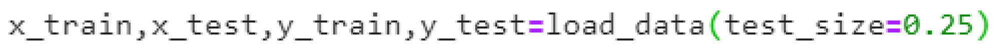
Chroma: Chroma represents 12 different pitch classes. One main property of chroma features is that they capture harmonic and melodic characteristics of music

MEL: The mel scale is a non-linear transformation of frequency scale based on the perception of pitches. The mel scale is calculated so that two pairs of frequencies separated by a delta in the mel scale are perceived by humans as being equidistant.



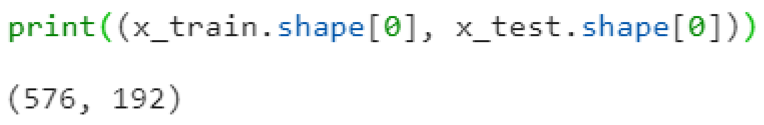
```
[3]:  emotions={
        '01':'neutral',
        '02':'calm',
        '03':'happy',
        '04':'sad',
        '05':'angry',
        '06':'fearful',
        '07':'disgust',
        '08':'surprised'
      }
      observed_emotions=['calm', 'happy', 'fearful', 'disgust']

[4]:  def load_data(test_size=0.2):
          x,y=[],[]
          for file in glob.glob("D:\\DataFlair\\ravdess data\\Actor_*\\*.wav"):
              file_name=os.path.basename(file)
              emotion=emotions[file_name.split("-")[2]]
              if emotion not in observed_emotions:
                  continue
              feature=extract_feature(file, mfcc=True, chroma=True, mel=True)
              x.append(feature)
              y.append(emotion)
          return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
```

We defined a dictionary to hold numbers and the emotions available in the RAVDESS dataset. After that, we defined another function that load the data and outputs the training and test set. Our function checks whether this emotion is in our list of "observed_emotions"; if not, it continues to the next file. It makes a call to extract_feature and stores what is returned in 'feature'. Then, it appends the feature to x and the emotion to y.



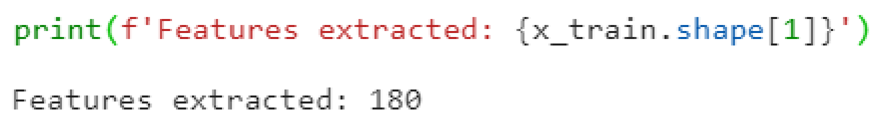
```
x_train,x_test,y_train,y_test=load_data(test_size=0.25)
```

We called the function that we defined above. Thus, training and testing set was created.



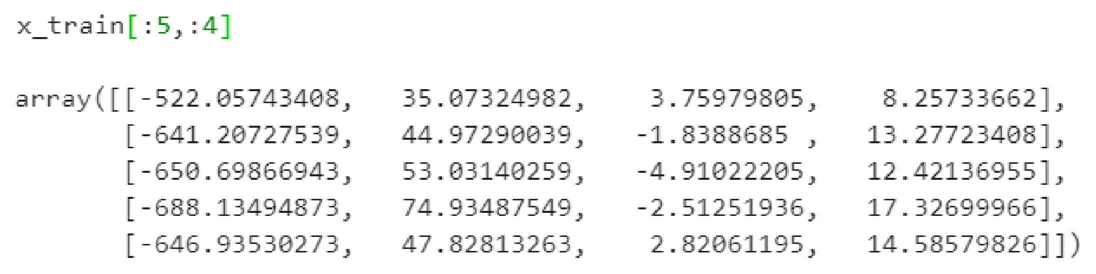
```
print((x_train.shape[0], x_test.shape[0]))

(576, 192)
```

While train set includes 576 human speech, test set includes 192.

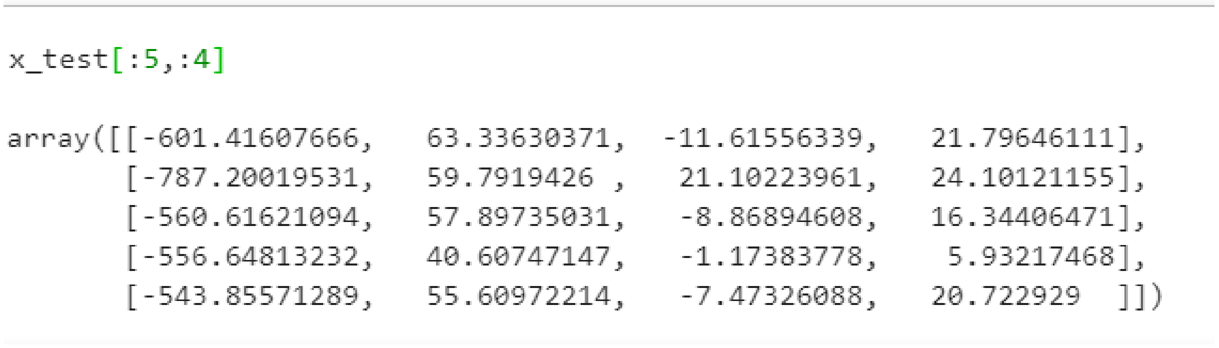


```
print(f'Features extracted: {x_train.shape[1]}')

Features extracted: 180
```

We have 180 features for all human speech data.

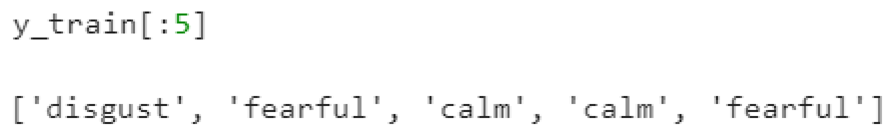


```
x_train[:5,:4]

array([[-522.05743408,   35.07324982,    3.75979805,    8.25733662],
       [-641.20727539,   44.97290039,   -1.8388685 ,   13.27723408],
       [-650.69866943,   53.03140259,   -4.91022205,   12.42136955],
       [-688.13494873,   74.93487549,   -2.51251936,   17.32699966],
       [-646.93530273,   47.82813263,    2.82061195,   14.58579826]])
```

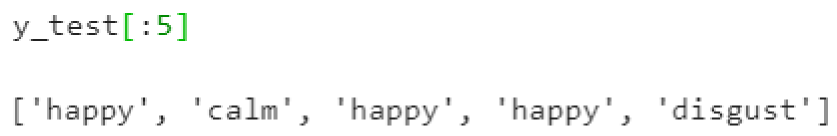


```
x_test[:5,:4]

array([[-601.41607666,   63.33630371,  -11.61556339,   21.79646111],
       [-787.20019531,   59.7919426 ,   21.10223961,   24.10121155],
       [-560.61621094,   57.89735031,   -8.86894608,   16.34406471],
       [-556.64813232,   40.60747147,   -1.17383778,    5.93217468],
       [-543.85571289,   55.60972214,   -7.47326088,   20.722929  ]])
```

The first 4 features for first 5 data in the train set and test set.



```
y_train[:5]

['disgust', 'fearful', 'calm', 'calm', 'fearful']
```



```
y_test[:5]

['happy', 'calm', 'happy', 'happy', 'disgust']
```

The first 5 output for the train set and test set.

As a result, we have a dataset in matrix format for use in training. We are considering including other datasets so that our model does not learn from just one of them. Our next step is to choose the machine learning model for our problem and to decide on the best model by examining the training and test results.