Speech Emotion Classification

A Project set by Visium SA

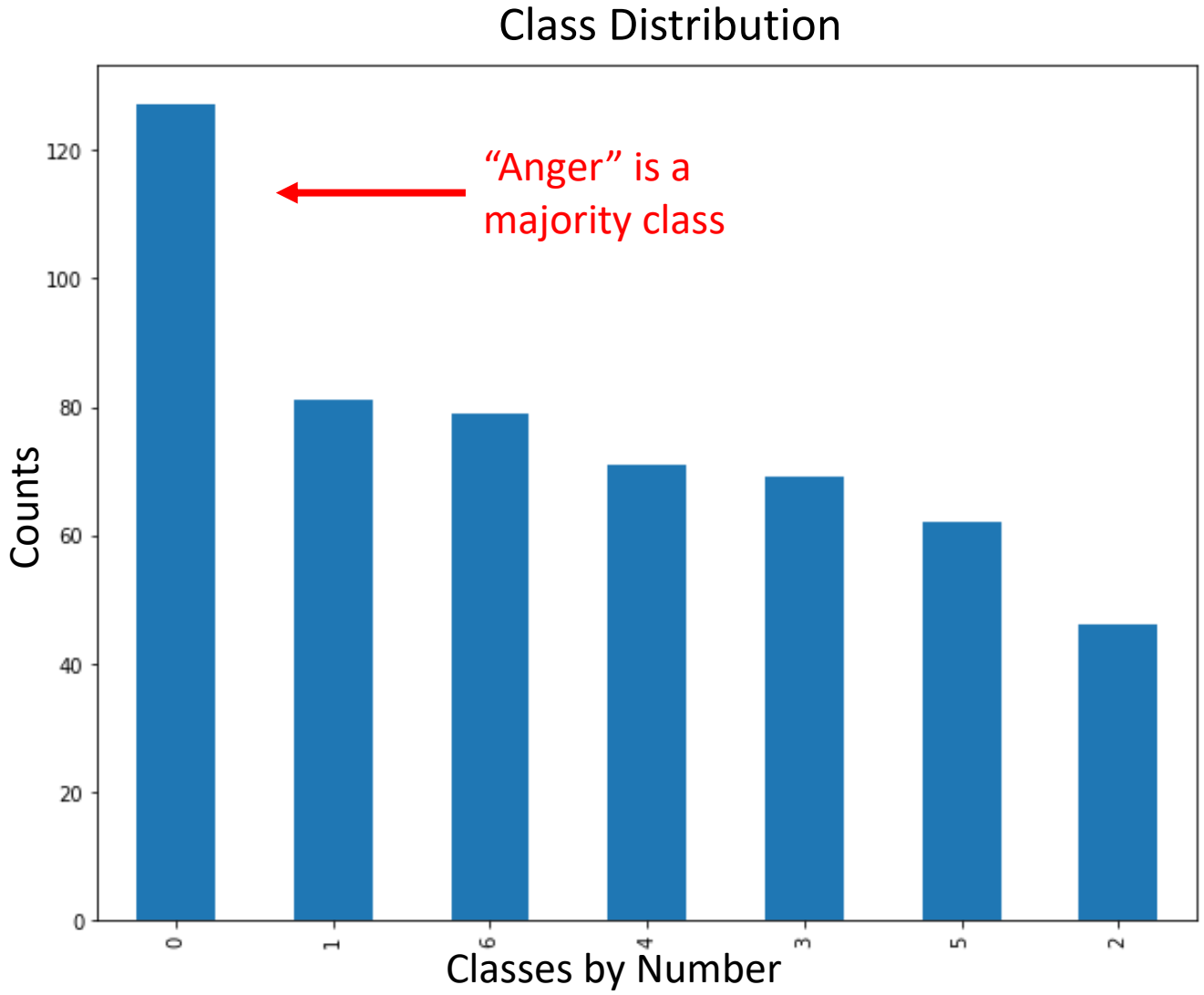
Edited by Claudio Fancoi

Exploratory Data Analysis

Classes:

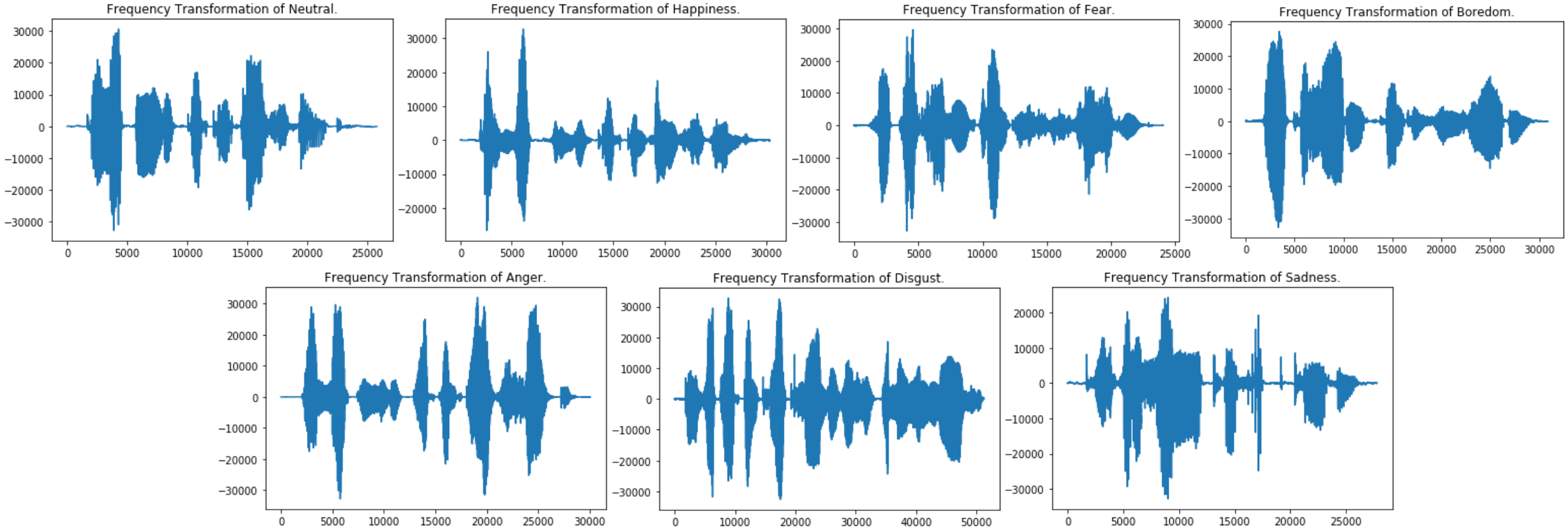
0. **Anger**
1. **Boredom**
2. **Disgust**
3. **Fear**
4. **Happiness**
5. **Sadness**
6. **Neutral**

Total Number of Samples: 535



Exploratory Data Analysis

The speech files are saved as .wav files, let's have a closer look at them.



Feature Creation

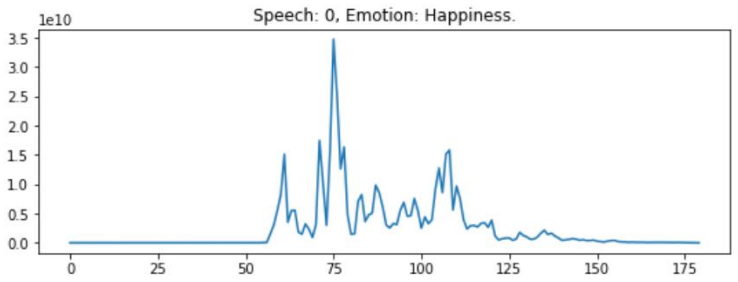
Because the frequency transformation (Fourier Transformation) of the sound waves did not yield any successful classification results, I had create new features.

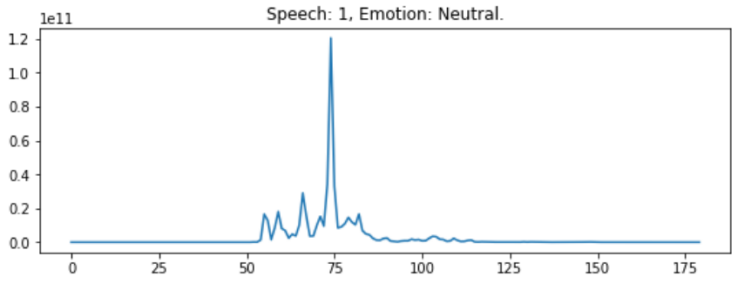
After a short internet search, I came along the Mel-Frequency Cepstrum Coefficients (MFCC). Prior art had shown to be successful using this feature transformation.

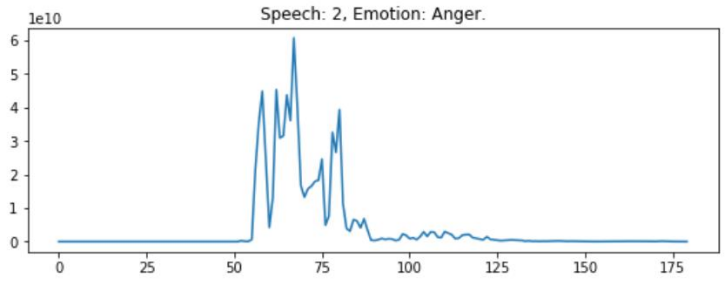
```python
# Source: https://data-flair.training/blogs/python-mini-project-speech-emotion-recognition/


def extract_feature(X, sample_rate,
                    mfcc=True, chroma= False, mel= False):
    """
    This function creates features, to be used for speech emotion recognition
    params:
        X: the frequencies of the speech
        sample_rate: the rate of the frequencies
        mfcc (default = True): Mel-frequency cepstrum coefficients - which bins the various
                               frequencies into a heatmap.
        chroma (default = False): Additional Chroma transformation
        mel (default = False): Addtional mel transormation
    return:
        Features based on the MFCC transformation
    """
    X = X.astype("float32")
    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
```

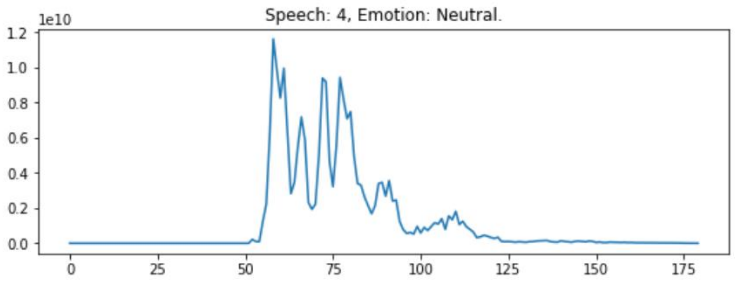
"In sound processing, the **mel-frequency cepstrum (MFC)** is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of alog power spectrum on a non-linear mel scale of frequency."
- Wikipedia

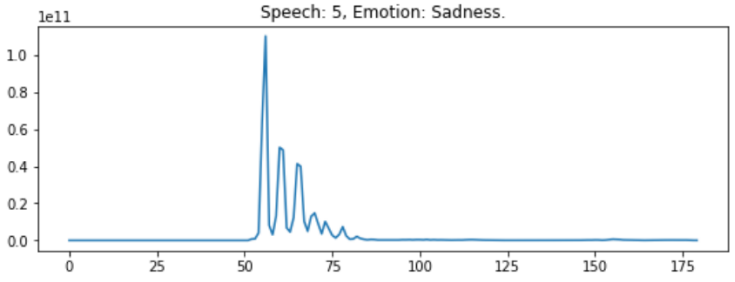
Feature Creation (MFCC)

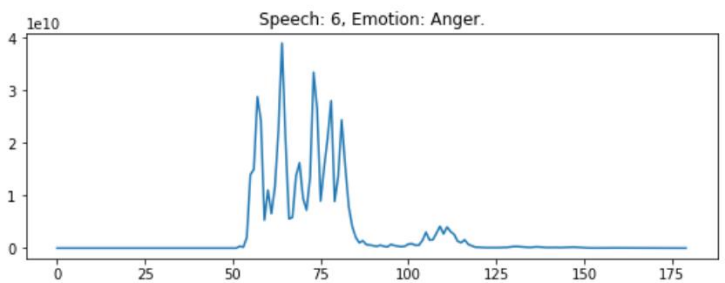












Classifier

After testing various classification algorithms, I have come to the conclusion, that a Support Vector Machine works best. For this, I used the scikit-learn library. As a representable metric, I decided to use F1 Macro, as the dataset is not perfectly balanced.

```
In [16]: x_train, x_val, y_train, y_val = tts(x_scaled,y, test_size=0.2)
         stf = StratifiedKFold(n_splits=4, random_state=42, shuffle=True)

In [17]: model = SVC(C = 10, class_weight="balanced")

In [18]: scores = cross_val_score(model, x_scaled, y, cv = stf, scoring="f1_macro")
         print(np.array(scores).mean())
         print(np.array(scores).std())

         0.7576901073242364
         0.019862131387328362
```

Implementation

$$Precision_M \quad \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l}$$

$$Recall_M \quad \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l}$$

$$Fscore_M \quad \frac{(\beta^2 + 1) Precision_M Recall_M}{\beta^2 Precision_M + Recall_M}$$

Metric Explanation

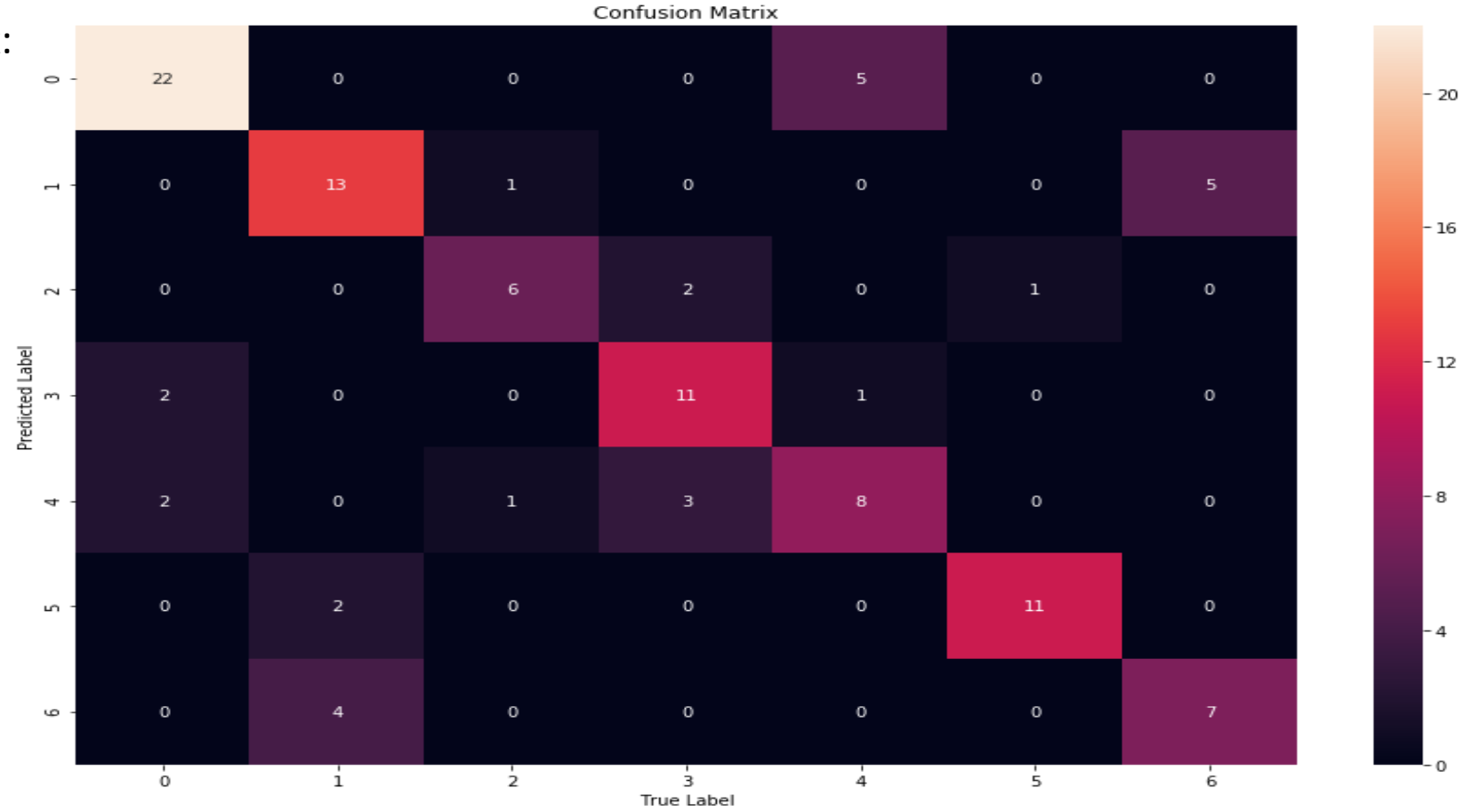
Classifier Results

F1 Macro Score:

0.758 (Mean, after 4 cross-validation cycles.)

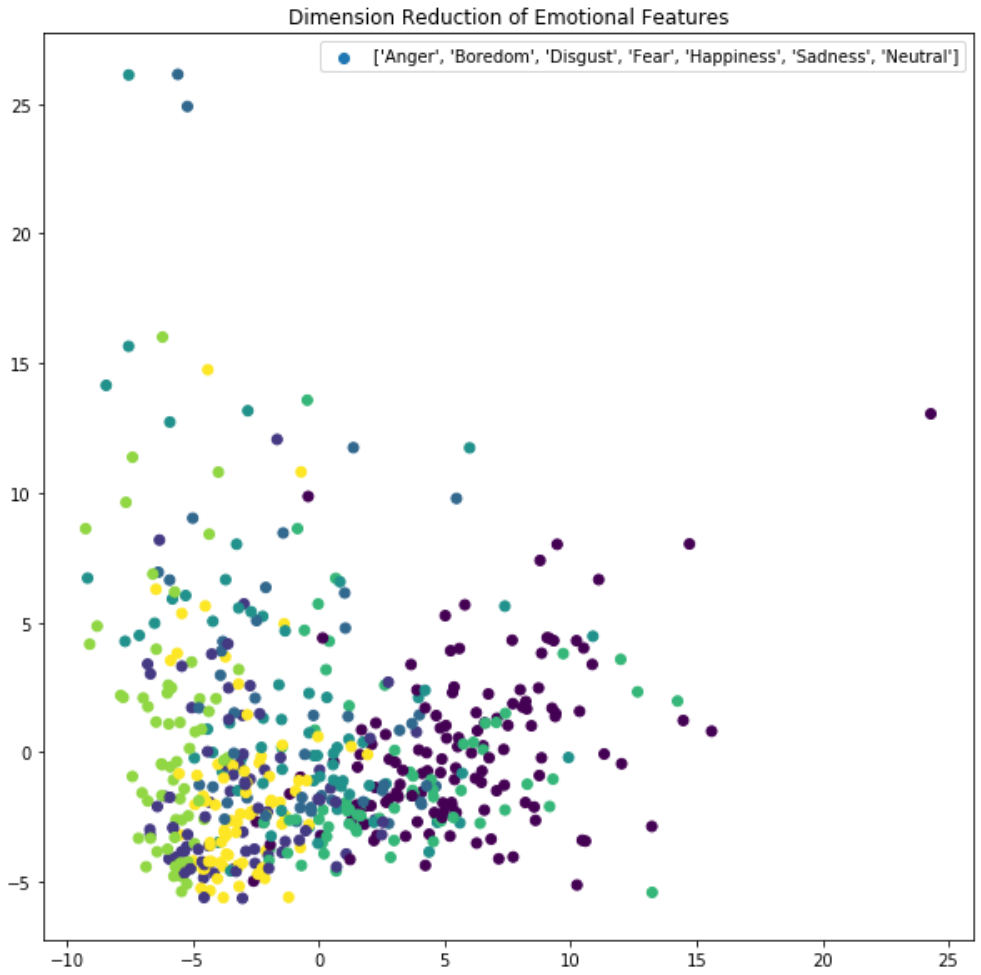
0.198 (Standard Deviation, after 4 cross-validation cycles.)

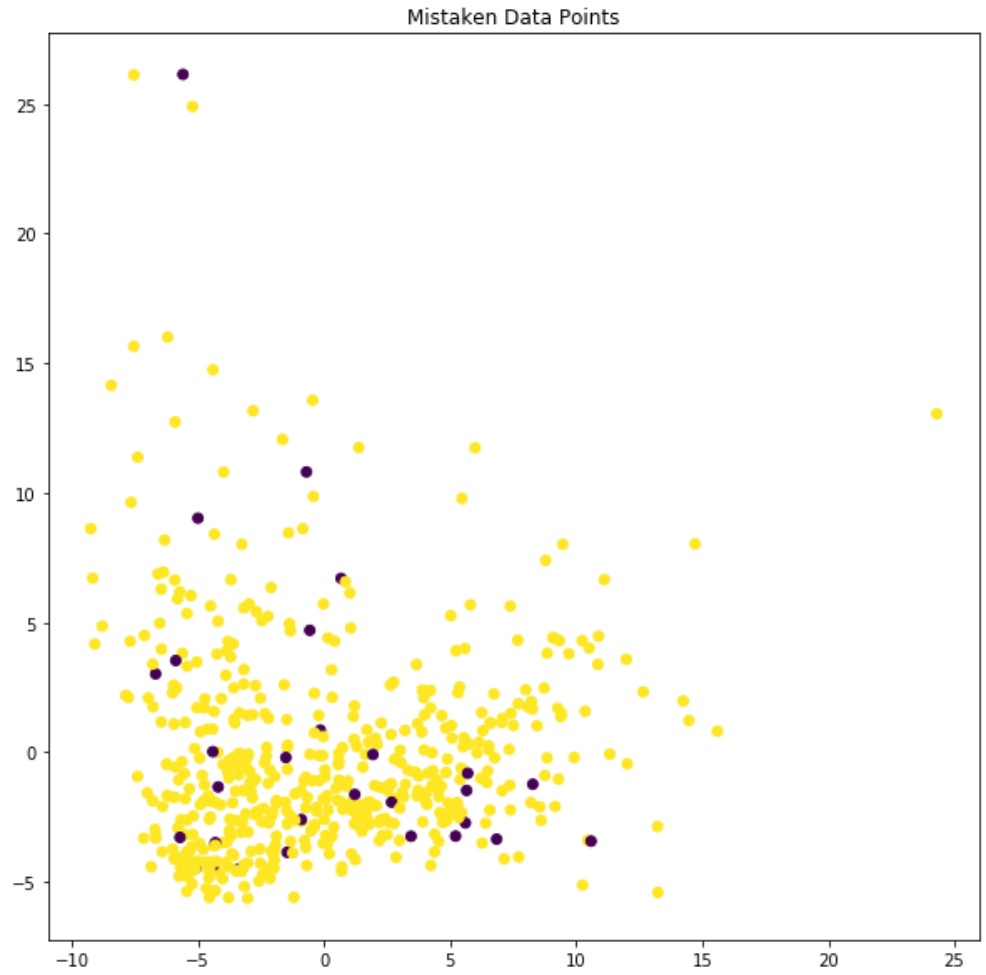
Confusion Matrix:



Dimension Reduction

After applying PCA dimension reduction, the data can be visualized better in 2 Dimensions:





Flask Implementation

- After the Docker container is started, it can be reached via two endpoints:
  - “/train” (GET) : By calling this route, the model is retrained on the whole dataset. It is also cross-validated, and the mean and standard deviation of the F1 macro score are returned via JSON.

```
response = {
    "STATUS" : "FITTED",
    "F1_MACRO_MEAN" : mean_cv_score,
    "F1_MACRO_STD" : std_cv_score
    }
return jsonify(response)
```

  - “/predict” (POST) : By calling this route, including a JSON body, which contains the name of a file, it returns the classified emotion, as well as the probability in a JSON string.

```
# Build response:
response = {
    "FILENAME" : filename,
    "EMOTION" : emotion,
    "PROBABILITY" : y_proba.max()
}
return jsonify(response)
```

ML Service Workflow