Problem Approach For Task 1

Team Name – ML Enthusiasts

Participants – Madhav Mathur, Kshitij Mohan

Task 1 –

In this task we were given a training and testing dataset comprising of several mp3 files and had to build a machine learning model to detect the emotion in each file.

Approach

The first step in any automatic speech recognition system is to extract features. So firstly, the given dataset was preprocessed using the librosa library to extract various features from the given audio files. Several augmentations like spectral contrast, spectral bandwidth etc. were used which were then added as features for the given audio. Since the most of the audio files were of small duration (< 3 sec), We used a sampling rate of 45100 to extract more information from each audio. The picture below shows the function code used for preprocessing each audio file.

```
#returns mfcc features with mean and standard deviation along time
def get_mfcc(name, path):
    b, _ = librosa.core.load(path + name, sr = SAMPLE_RATE)
    assert _ == SAMPLE_RATE
    try:
        ft1 = librosa.feature.mfcc(b, sr = SAMPLE_RATE, n_mfcc = 20)
        ft2 = librosa.feature.zero_crossing_rate(b)[0]
        ft3 = librosa.feature.spectral_rolloff(b)[0]
        ft4 = librosa.feature.spectral_centroid(b)[0]
        ft5 = librosa.feature.spectral_contrast(b)[0]
        ft6 = librosa.feature.spectral_bandwidth(b)[0]
        ft7 = librosa.feature.spectral_flatness(b)[0]
        ft8 = librosa.feature.melspectrogram(b)[0]

        ft1_trunc = np.hstack((np.mean(ft1, axis = 1), np.std(ft1, axis = 1), skew(ft1, axis =
1), np.max(ft1, axis = 1), np.min(ft1, axis = 1), np.sum(ft1, axis = 1)))
        ft2_trunc = np.hstack((np.mean(ft2), np.std(ft2), skew(ft2), np.max(ft2), np.min(ft2),
np.sum(ft2)))
        ft3_trunc = np.hstack((np.mean(ft3), np.std(ft3), skew(ft3), np.max(ft3), np.min(ft3),
np.sum(ft3)))
        ft4_trunc = np.hstack((np.mean(ft4), np.std(ft4), skew(ft4), np.max(ft4), np.min(ft4),
np.sum(ft4)))
        ft5_trunc = np.hstack((np.mean(ft5), np.std(ft5), skew(ft5), np.max(ft5), np.min(ft5),
np.sum(ft5)))
        ft6_trunc = np.hstack((np.mean(ft6), np.std(ft6), skew(ft6), np.max(ft6), np.min(ft6),
np.sum(ft6)))
        ft7_trunc = np.hstack((np.mean(ft7), np.std(ft7), skew(ft7), np.max(ft7), np.min(ft7),
np.sum(ft7)))
        ft8_trunc = np.hstack((np.mean(ft8), np.std(ft8), skew(ft8), np.max(ft8), np.min(ft8),
np.sum(ft8)))
        ft9_trunc = librosa.core.get_duration(b, sr = SAMPLE_RATE)

        return pd.Series(np.hstack((ft1_trunc, ft2_trunc, ft3_trunc, ft4_trunc, ft5_trunc, ft6_
trunc, ft7_trunc, ft8_trunc, ft9_trunc)))

    except:
        print('bad file')
        return pd.Series([0]*115)
```

Model Training & Inference

For training, StratifiedKFold was used so that the proportion of labels in each fold was kept same. The models used were LightGBM Classifier, ExtraTrees Classifier and RandomForest Classifier. These 3 models were then trained on 10 folds of data and weighted ensemble of their prediction probabilities was taken by using Voting Classifier. After that argmax was used to find the max probability index which was then used for identifying the correct class. The models performed decently on both training and testing data. After model training, we applied a simple postprocessing in which we considered those predictions where predicted value was 'neutral' class and the model confidence was between 50 and 51 %. In such cases we took the next most probable class (which had the second highest probability). This gave us a boost on the public leaderboard f1-score (59.47 -> 60.18). Below is a screenshot of my postprocessing function –

The complete training & inference code file is also available in the uploaded zip folder by the name of "intelligence-augmentation-ia-for-ai-challenge.ipynb".

```
postprocess = True
if postprocess:
    count = 0
    final_preds1 = final_preds.copy()
    for i in range(len(final_preds1)):
        temp = np.argmax(final_preds1[i])
        if(temp == 4 and final_preds1[i][temp] > 0.5 and final_preds1[i][temp] <= 0.51):
            final_preds1[i][temp] = 0
            count += 1
    print(count)
    test_data['emotion'] = le.inverse_transform(np.argmax(final_preds1, axis = -1))
    test_data[['filename', 'emotion']].to_csv("submission_postprocess.csv", index = False)
```