PATTERNS RECOGNITION

Assignment #3

Speech Emotions Recognition

COLLABORATORS

i	Name	ID
1	Zeyad Zidan	19015709
2	Abdelrahman Gad	19015894
3	Abdelaziz Mohammed	19015941

You can view the ipynb file on **google colaboratory** here.

Problem Statement

Speech is the most natural way of expressing ourselves as humans. It is only natural then to extend this communication medium to computer applications. We define speech emotion recognition (SER) systems as methodologies that process and classify speech signals to detect embedded emotions. Below we are showing how we achieved the goal of the assignment.

Understanding the Dataset

The dataset is composed of 7442 wav-formatted audio files. Dealing with the dataset is not easy as audio files are not discrete signals. Relying on Python and sampling, we can represent the audio files as samples with a sufficient sampling rate. This method opens the way to formulate the audio files into an understandable dataset. In the following section, we are showing how we used **librosa** to formulate a dataset from the audio files.

Librosa

We used librosa to load the audio files and get the samples of each audio file. Adding each array of samples to an array, we successfully constructed a dataset of samples corresponding to each audio file.

Getting the labels is done by indexing the letters [9, 12] of the file name and then encoding each label to a numeric value.

Librosa also can display the waveform of each audio file.

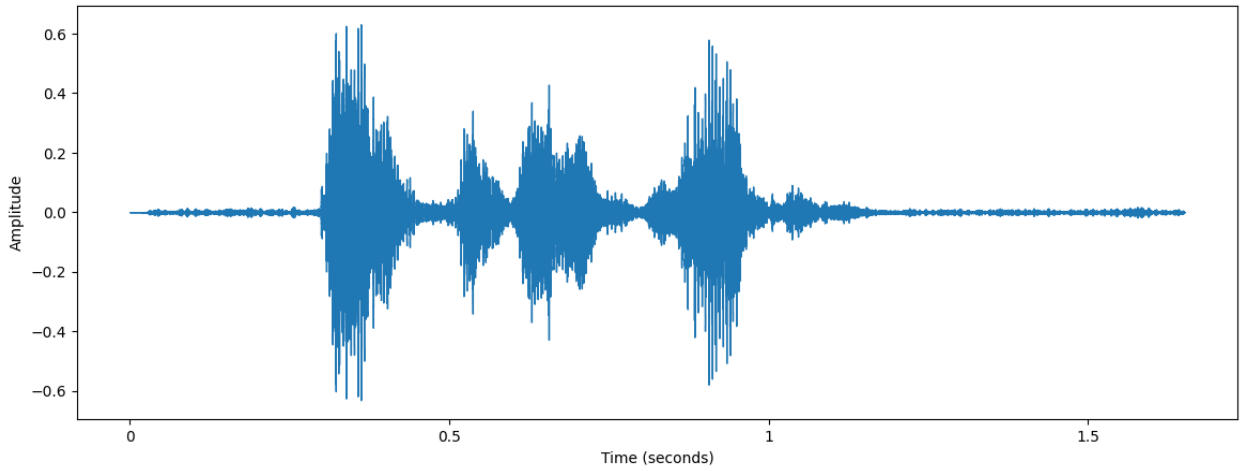


Figure 1. Sample waveform of an audio

Creating the Feature Space

Zero Crossing Rate Feature Space

The zero crossing rate is the rate of sign changes of the signal during the duration of a particular frame. To construct such feature space, librosa offers a built-in method in librosa.feature called zero_crossing_rate. Simply for each audio file, call this line of code:

```
librosa.feature.zero_crossing_rate(y=entry)[0]
```

Finally, append the previous result to the feature space list. Return the list after finishing each entry of the dataset.

Note that each entry in the dataset represents the samples array of each audio file.

Energy Feature Space

The energy of an audio file is the sum of squares of the signal values, normalized by the respective frame length. Librosa offers such an option in the same class as the zero crossing rate. To construct the energy feature space, Simply for each audio file, call this line of code:

```
                frames      =      librosa.util.frame(entry,
frame_length=frame_size, hop_length=hop_length)

        # Compute energy of each frame

        energies = np.sum(frames**2, axis=0) / frame_size

        feature_space.append(energies)
```

Finally, return the feature space array.

Mel Spectrogram

To construct the mel spectrogram feature space, we run the following code

```
1  def construct_melspectrogram(dataset, rate):
2    spectrogram = []
3    for i in range(len(dataset)):
4      spectrogram.append(librosa.feature.melspectrogram(y=dataset[i], sr=rate[i]))
5
6    return spectrogram
```

Building the Model

Data Augmentation

In the preparation step of the feature space, we take each entry and apply an augmentation algorithm to it. The augmentation algorithm we follow applies a number of **A** augmentations, e.g. **4** augmentations. We discuss the algorithm in the following points:

- For each augmentation, apply some **random** noise to a copy of the original feature space, i.e. we have a new unique feature space.
- The number of feature spaces created from the previous step will be a total of **4** newly identified random feature spaces based on the random noise addition by the end of all augmentations.
- In the same epoch (augmentation), we apply **random** shifting operations to a copy of the original feature space.
- The number of feature spaces created from the previous step also will be a total of **4** newly identified random feature spaces based on the random shifting operations by the end of all augmentations.
- In the same epoch, we apply **random** scaling operations to a copy of the original feature space. By the end of all epochs, we will have another **4** newly identified random feature spaces based on the random scaling operations.
- Finally, we concatenate all of the newly created feature spaces (**4 + 4 + 4 = 12)** to the original feature space to have a total of **((3A + 1) x N) x M)**, where 3 stands for the number of applied operations (noise addition, shifting, and scaling), A stands for the number of epochs (augmentations), the 1 stands for the original feature space, N is the total number of entries in the original feature space, and M is the total number of features in the original feature space.

When to use this approach?

We use this approach after splitting the data into training and testing data to not mess with the original testing volume of data, that is we apply data augmentation on the **training data only** after split and exclude the testing data out of operation.

Best Model Results

We have tried multiple models, and so far this was the best model results

Layer (type)	Output Shape	Param #
conv1d_23 (Conv1D)	(None, 192, 1024)	6144
max_pooling1d_23 (MaxPooling1D)	(None, 94, 1024)	0
conv1d_24 (Conv1D)	(None, 90, 1024)	5243904
max_pooling1d_24 (MaxPooling1D)	(None, 43, 1024)	0
conv1d_25 (Conv1D)	(None, 39, 512)	2621952
max_pooling1d_25 (MaxPooling1D)	(None, 18, 512)	0
conv1d_26 (Conv1D)	(None, 14, 256)	655616
max_pooling1d_26 (MaxPooling1D)	(None, 5, 256)	0
flatten_8 (Flatten)	(None, 1280)	0
dense_17 (Dense)	(None, 512)	655872
dropout_2 (Dropout)	(None, 512)	0
dense_18 (Dense)	(None, 256)	131328
dropout_3 (Dropout)	(None, 256)	0
dense_19 (Dense)	(None, 6)	1542
Total params: 9,316,358 **Trainable params: 9,316,358** **Non-trainable params: 0**		

Big Picture

In this section we show a comparison between a variety of models we have tried for each class of models. We discuss the 1D model first, breaking it down into 5 different models with different accuracies and confusing classes, then we discuss the 2D models with a variety of 3 different models.

1D Models

Model #	Most Confusing Class	Most Confusing Class Precision	Accuracy
1	FEAR	0.27	0.47
2	FEAR	0.40	0.49
3	NEUTRAL	0.40	0.49
4	NEUTRAL	0.34	0.47
5	FEAR	0.33	0.44

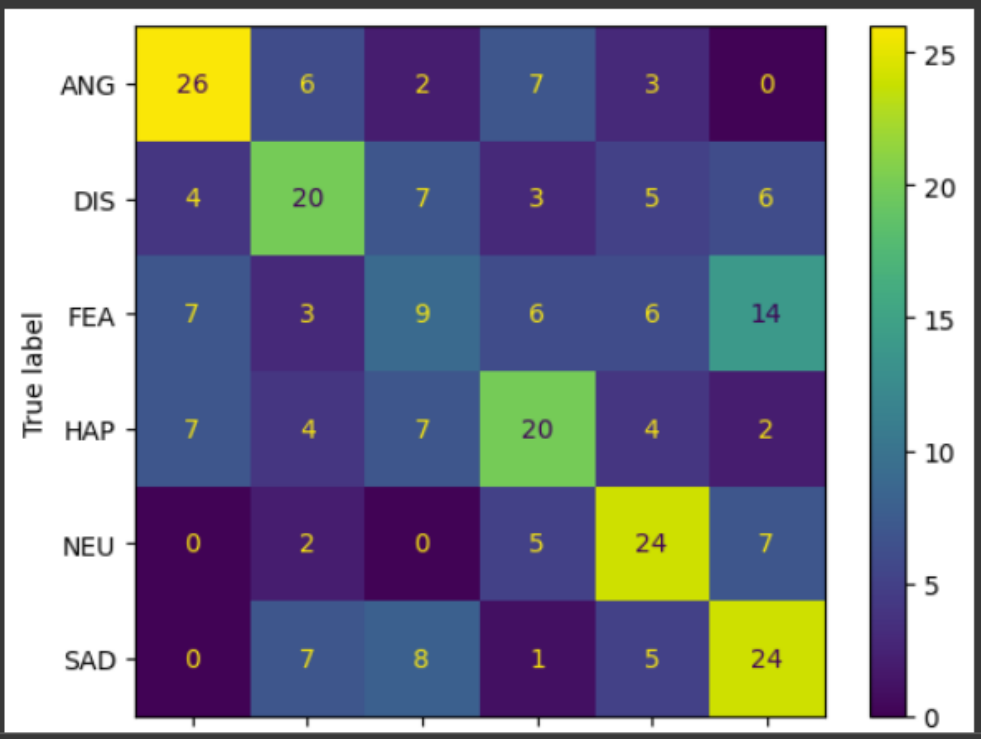
Best accuracy → 0.49 in model #2 and model #3.

First Model

In this model, most confusing class is **FEAR** as its precision equals **0.27**. The accuracy of this model is **0.47**.

```
test_model(model,x_val,y_val)
```

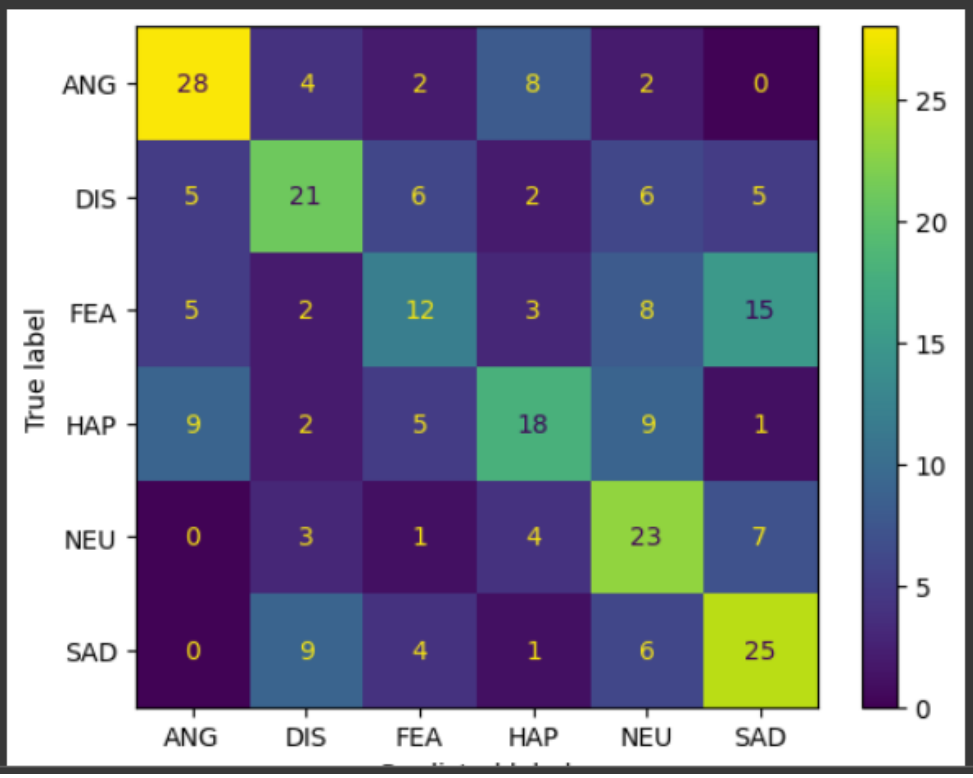
```
9/9 [==============================] - 0s 3ms/step
              precision    recall  f1-score   support

         ANG       0.59      0.59      0.59        44
         DIS       0.48      0.44      0.46        45
         FEA       0.27      0.20      0.23        45
         HAP       0.48      0.45      0.47        44
         NEU       0.51      0.63      0.56        38
         SAD       0.45      0.53      0.49        45

    accuracy                           0.47       261
   macro avg       0.46      0.48      0.47       261
weighted avg       0.46      0.47      0.46       261
```



Second Model

In this model, most confusing class is **FEAR** as its precision equals **0.40**. The accuracy of this model is **0.49**.

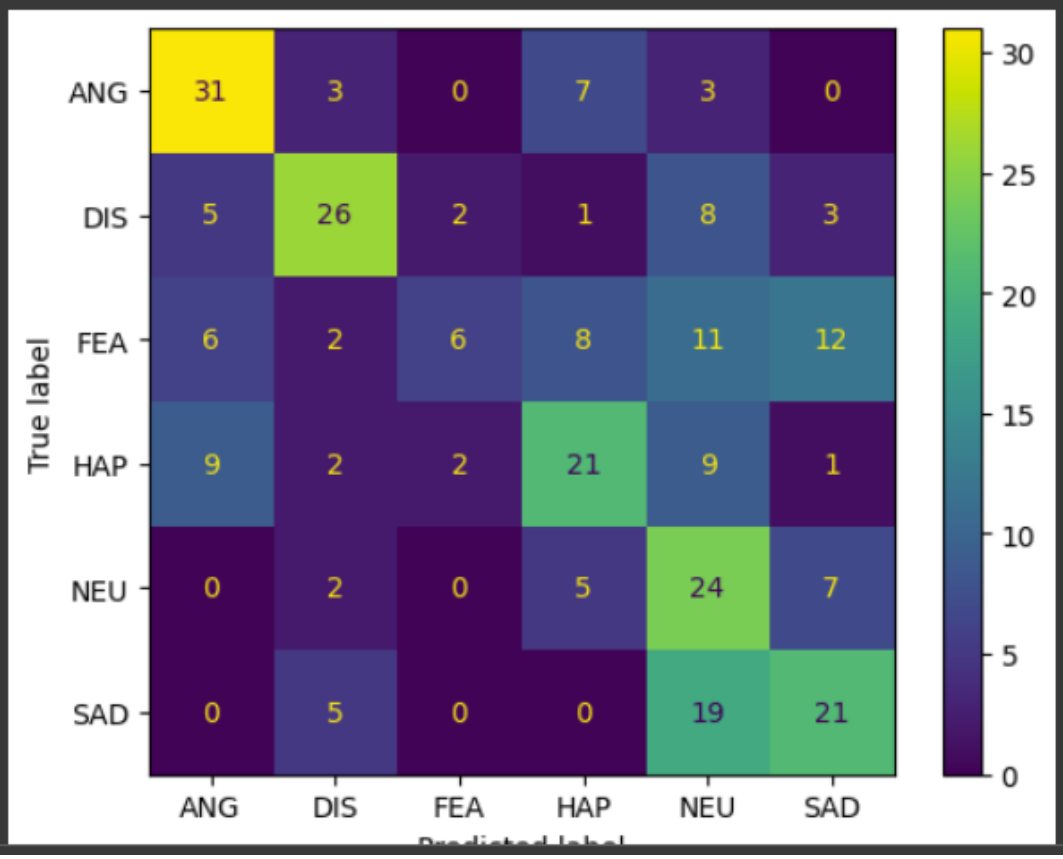
```
test_model(modeltry_1D,x_val,y_val)

9/9 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

         ANG       0.60      0.64      0.62        44
         DIS       0.51      0.47      0.49        45
         FEA       0.40      0.27      0.32        45
         HAP       0.50      0.41      0.45        44
         NEU       0.43      0.61      0.50        38
         SAD       0.47      0.56      0.51        45

    accuracy                           0.49       261
   macro avg       0.48      0.49      0.48       261
weighted avg       0.49      0.49      0.48       261
```



Third Model

In this model, most confusing class is **NEUTRAL** as its precision equals **0.40**. The accuracy of this model is **0.49**.

```
test_model(modeltry2_1D, x_val, y_val)

9/9 [==============================] - 0s 32ms/step
              precision    recall  f1-score   support

         ANG       0.61      0.70      0.65        44
         DIS       0.65      0.58      0.61        45
         FEA       0.60      0.13      0.22        45
         HAP       0.50      0.48      0.49        44
         NEU       0.32      0.63      0.43        38
         SAD       0.48      0.47      0.47        45

    accuracy                           0.49       261
   macro avg       0.53      0.50      0.48       261
weighted avg       0.53      0.49      0.48       261
```

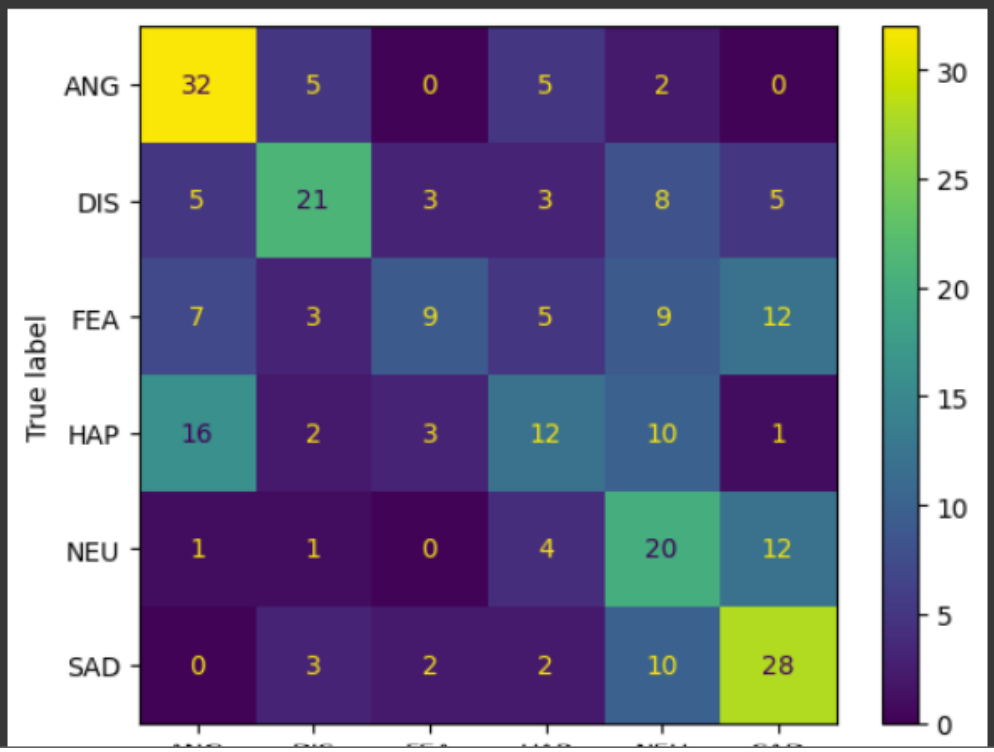


Fourth Model

In this model, the most confusing class is **NEUTRAL** as its precision equals **0.34**. The accuracy of this model is **0.47**.

```
test_model(model3_1D, x_val, y_val)
```

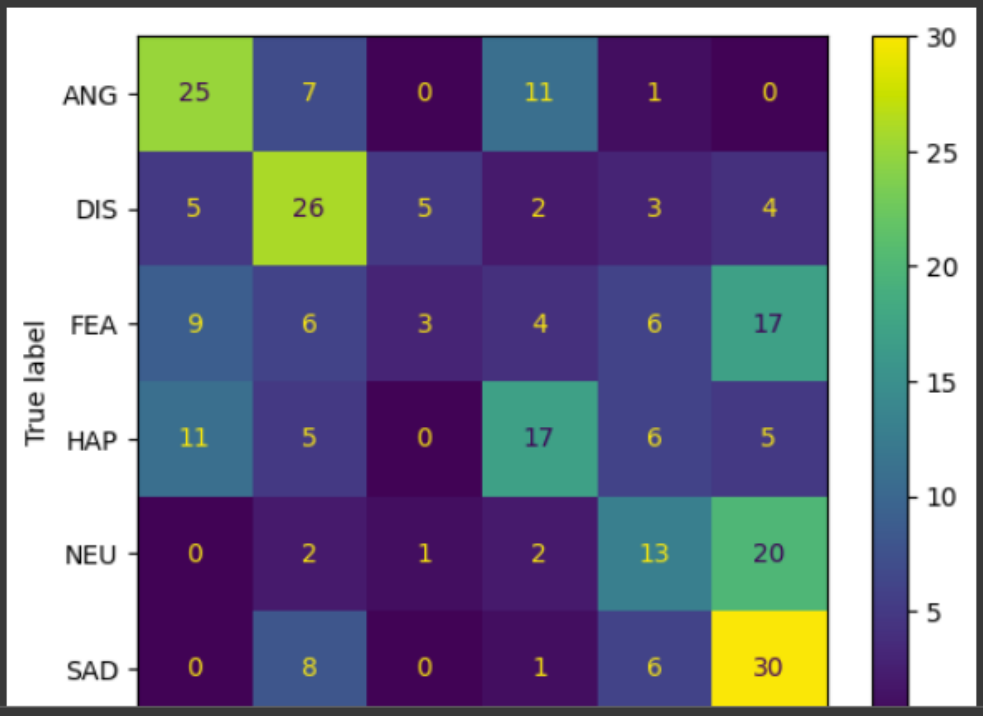
```
9/9 [==============================] - 0s 6ms/step
              precision    recall  f1-score   support

         ANG       0.52      0.73      0.61        44
         DIS       0.60      0.47      0.52        45
         FEA       0.53      0.20      0.29        45
         HAP       0.39      0.27      0.32        44
         NEU       0.34      0.53      0.41        38
         SAD       0.48      0.62      0.54        45

    accuracy                           0.47       261
   macro avg       0.48      0.47      0.45       261
weighted avg       0.48      0.47      0.45       261
```



Fifth Model

In this model, the most confusing class is **FEAR** as its precision is **0.33** with an accuracy of **0.44**.

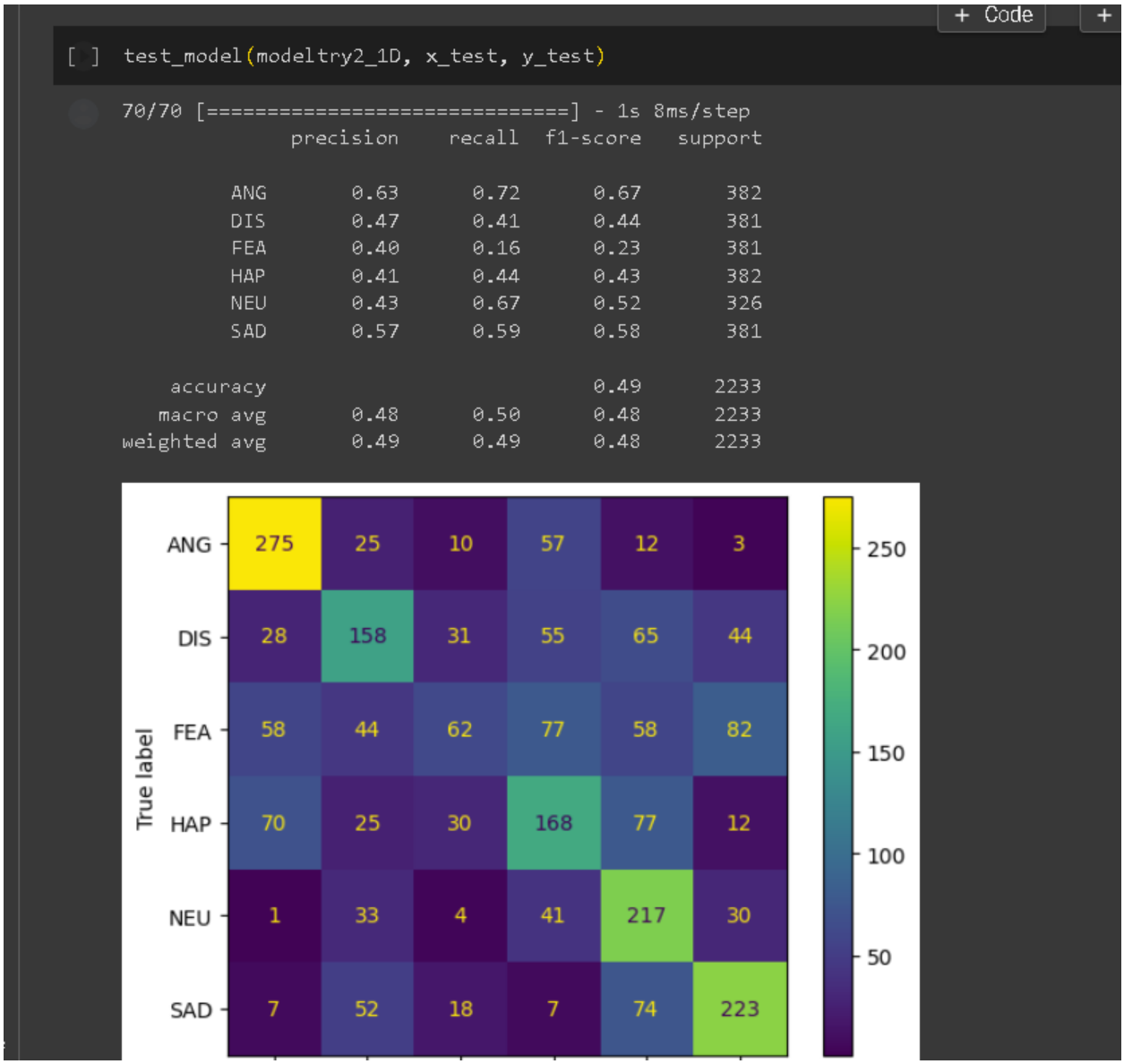
```
test_model(modell, x_val, y_val)

9/9 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

         ANG       0.50      0.57      0.53        44
         DIS       0.48      0.58      0.53        45
         FEA       0.33      0.07      0.11        45
         HAP       0.46      0.39      0.42        44
         NEU       0.37      0.34      0.36        38
         SAD       0.39      0.67      0.50        45

    accuracy                           0.44       261
   macro avg       0.42      0.43      0.41       261
weighted avg       0.42      0.44      0.41       261
```



Conclusion

So it appears that the model **(modeltry2_1D)** has the best accuracy in the validation so we apply this model on testing.

Accuracy: **0.49**.



2D Models

Model #	Most Confusing Class	Most Confusing Class Precision	Accuracy
1	DISSECTED	0.35	0.46
2	DISSECTED	0.44	0.51
3	ANGRY	0.17	0.17

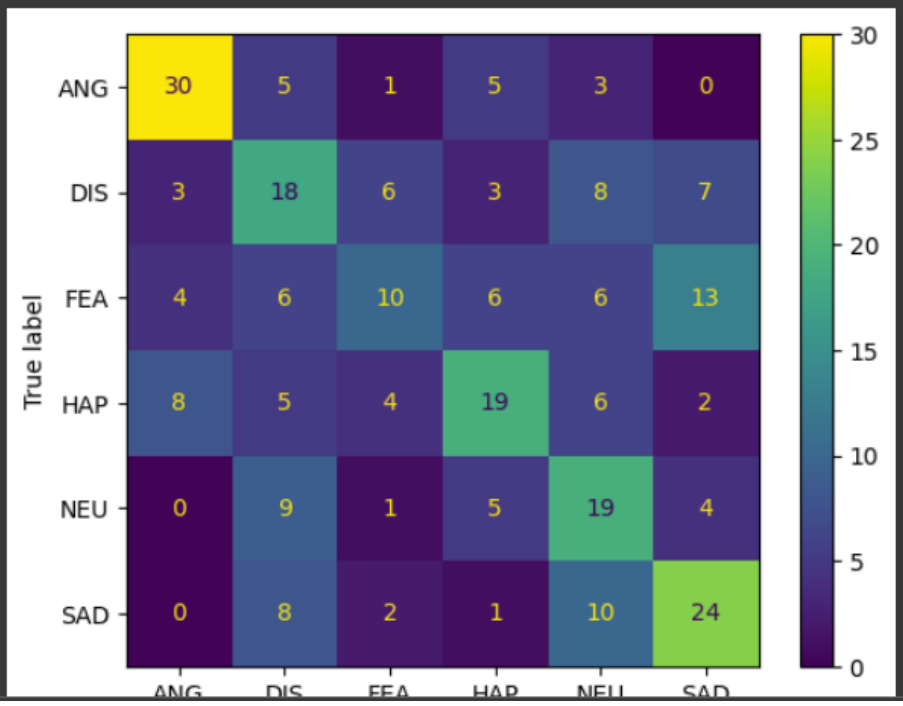
Best accuracy → 0.51 in model #2.

First Model

This model get accuracy **0.46** and the most confusing classes here is **DISSECTED** as it appears in the matrix.

```
test_model(model2_2D,x_val2,y_val2)
```

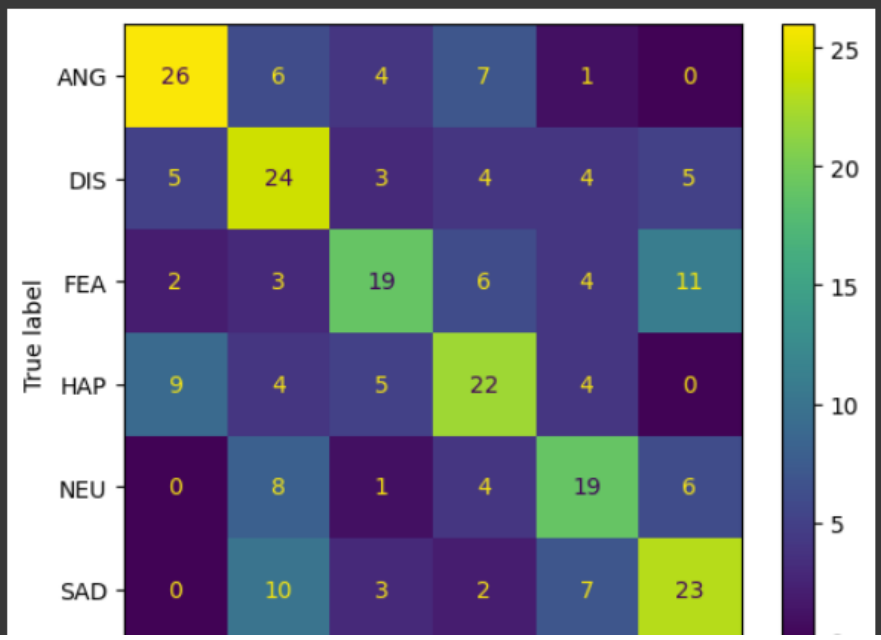
```
9/9 [==============================] - 0s 6ms/step
              precision    recall  f1-score   support

         ANG       0.67      0.68      0.67        44
         DIS       0.35      0.40      0.38        45
         FEA       0.42      0.22      0.29        45
         HAP       0.49      0.43      0.46        44
         NEU       0.37      0.50      0.42        38
         SAD       0.48      0.53      0.51        45

    accuracy                           0.46       261
   macro avg       0.46      0.46      0.45       261
weighted avg       0.46      0.46      0.45       261
```



Second Model

This model4_2D accuracy is **0.51 (2D)**, and the most confusing class is **DISSECTED** as it appears in the matrix.

```
[45] test_model(model4_2D ,x_val2,y_val2)

9/9 [==============================] - 1s 8ms/step
              precision    recall  f1-score   support

         ANG       0.62      0.59      0.60        44
         DIS       0.44      0.53      0.48        45
         FEA       0.54      0.42      0.48        45
         HAP       0.49      0.50      0.49        44
         NEU       0.49      0.50      0.49        38
         SAD       0.51      0.51      0.51        45

    accuracy                           0.51       261
   macro avg       0.51      0.51      0.51       261
weighted avg       0.51      0.51      0.51       261
```

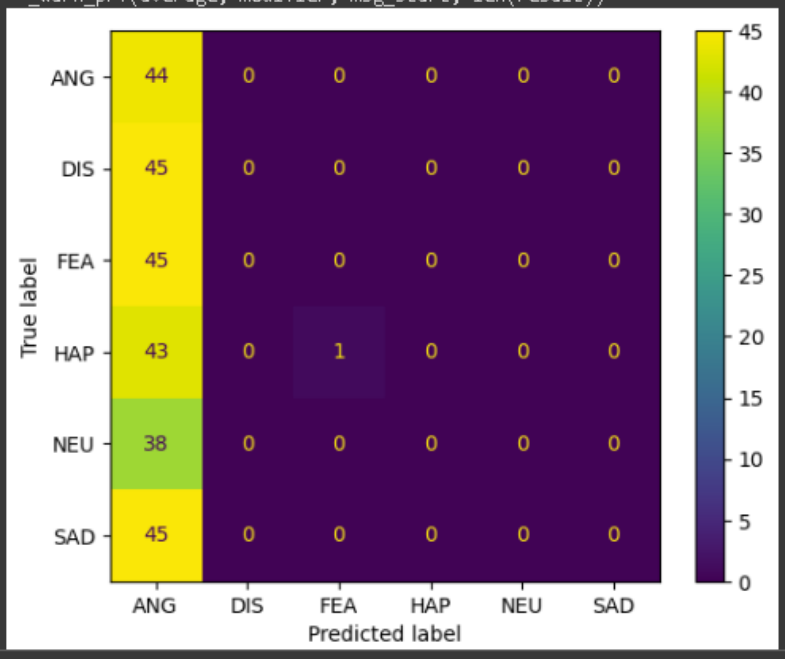


Third Model

This model accuracy is **0.17** in which it appear it detect all sounds as an angry voice.

```
test_model(model_2D,x_val2,y_val2)
```

```
9/9 [==============================] - 1s 21ms/step
              precision    recall  f1-score   support

         ANG       0.17      1.00      0.29        44
         DIS       0.00      0.00      0.00        45
         FEA       0.00      0.00      0.00        45
         HAP       0.00      0.00      0.00        44
         NEU       0.00      0.00      0.00        38
         SAD       0.00      0.00      0.00        45

    accuracy                           0.17       261
   macro avg       0.03      0.17      0.05       261
weighted avg       0.03      0.17      0.05       261

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: P
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: P
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: P
  _warn_prf(average, modifier, msg_start, len(result))
```



Conclusion

So it appears that the best one for the 2d validation is model4_2D with an accuracy that equals **0.51**, So we apply testing to it and it appears that the accuracy is **0.49** and the most confusing matrix is **DIS**.

```
test_model(model2_2D,x_test2,y_test2)
```

```
70/70 [==============================] - 0s 5ms/step
              precision    recall  f1-score   support

         ANG       0.61      0.63      0.62       382
         DIS       0.40      0.36      0.38       381
         FEA       0.41      0.25      0.31       381
         HAP       0.44      0.47      0.46       382
         NEU       0.51      0.61      0.56       326
         SAD       0.51      0.64      0.57       381

    accuracy                           0.49      2233
   macro avg       0.48      0.49      0.48      2233
weighted avg       0.48      0.49      0.48      2233
```