MACHİNE LEARNİNG FOR SİGNAL PROCESSİNG – EHB-328E

PROJECT FINAL REPORT
SPEECH EMOTİON RECOGNITION

Prof. Dr. Ender Mete Ekşioğlu

Osman Kaan Kurtça

040160090

Aziz Gündoğdu

040160112

1.Data Preprocessing

1.1 Normalization

In the last part of the progress report, we extract 180 features for each sound file. Now, we will normalize our dataset and apply dimensionality reduction if it's needed.

First, we edited the load_data and extract_feature functions that I wrote in the first report. In progress report, we created our train and test set, but we did not normalize the dataset. So that, we normalize dataset using Standard Scaler from scikit-learn package.

The Standard Scaler assumes your data is normally distributed within each feature and will scale them such that the distribution is now centered around 0, with a standard deviation of 1. The mean and standard deviation are calculated for the feature and then the feature is scaled based on:

$$\frac{x_i - mean(x)}{stdev(x)}$$

	0	1	2	3	4	5
0	-709.03351	55.76609	2.69818	16.39175	3.37237	-1.10114
1	-695.37878	61.31311	-0.60217	14.27753	4.69573	-2.75294
2	-687.33856	57.97822	0.12055	13.90119	1.86231	1.50366
3	-684.72296	62.30627	-0.77971	15.71041	2.55225	1.15999
4	-717.24048	63.73140	2.22593	15.64493	4.03384	-2.02896
5	-678.53870	59.68554	-2.25319	13.61240	1.60572	-2.37622
6	-686.34979	60.72581	2.91972	10.66652	4.11751	0.55427
7	-668.66919	55.60154	-3.68078	15.51894	2.26318	0.15237
8	-630.32843	55.67267	-4.81232	14.59022	-2.40618	-4.58026
9	-610.24481	53.79882	-5.08196	17.26598	0.76053	-5.32776
10	-642.02765	54.75199	-1.59563	12.08410	-1.71814	-2.60358

Figure.1 – Before normalization for 5 feature

	0	1	2	3	4	5
0	-1.28632	0.77248	0.75438	0.80234	0.97625	0.47297
1	-1.15431	1.11405	0.49147	0.56361	1.12442	0.23963
2	-1.07658	0.90869	0.54905	0.52111	0.80718	0.84093
3	-1.05130	1.17520	0.47733	0.72540	0.88443	0.79238
4	-1.36566	1.26296	0.71676	0.71801	1.05031	0.34190
5	-0.99151	1.01383	0.35995	0.48850	0.77846	0.29285
6	-1.06702	1.07788	0.77203	0.15586	1.05968	0.70682
7	-0.89610	0.76234	0.24623	0.70378	0.85207	0.65004
8	-0.52544	0.76672	0.15610	0.59891	0.32928	-0.01850
9	-0.33129	0.65134	0.13462	0.90105	0.68383	-0.12410
10	-0.63854	0.71003	0.41234	0.31593	0.40631	0.26073

Figure.2 – After normalization for 5 feature

1.2 Dimensionality Reduction

We applied principal component analysis (PCA) to reduce the dimension of our dataset. At first, we chose 32 as the principal component number. We can change this number according to the train and test result of our model or we may not use this method at all.

```
pca_32 = PCA(n_components=32, random_state=42)
X_scaled = pca_32.fit_transform(x_scaled)
```

Figure.3 – PCA method application

```
Shape of dataset: (1440, 180)
Shape of dataset after epplying PCA: (1440, 32)
```

Figure.4 – Change of dataset size

Note: Instead of applying PCA, according to the test results we will get, we can reduce the dimension of our data manually by examining the dataset.

1.3 Determining the emotions to be predicted

The RAVDESS dataset contains audio files of 8 different emotions. To get a better predictive model, we thought of removing some emotions that would be similar and rare.

```
deleted_index = np.where(y == "neutral")
deleted_index = np.concatenate([deleted_index,np.where(y == "fearful")],axis=1)
deleted_index = np.concatenate([deleted_index,np.where(y == "disgust")],axis=1)
x=np.delete(x, deleted_index, 0)
y=np.delete(y, deleted_index, 0)
```

Figure.5 – Emotions extracted from the dataset.

In this case, there are 5 emotions we can predict: Calm, Happy, Sad, Angry, Surprised.

1.4 Label Encoder and One-Hot Encoder

Since our labels types are string, we should encode these as a numerical value. We will use either one according to our model

1.4.1 Label Encoder

It is used to digitize the data exactly. That is, it assigns a numerical value to each categorical data.

1.4.2 One-Hot Encoder

This allows us to realize categorical partition binarization. It gives a "1" to the current value and a "0" to those that do not exist.

	0
0	4.00000
1	4.00000
2	3.00000
3	1.00000
4	0.00000
5	2.00000
6	1.00000
7	1.00000
8	0.00000
9	4.00000
10	2.00000

Figure.6 – Label Encoding

	0	1	2	3	4
0	0.00000	0.00000	0.00000	0.00000	1.00000
1	0.00000	0.00000	0.00000	0.00000	1.00000
2	0.00000	0.00000	0.00000	1.00000	0.00000
3	0.00000	1.00000	0.00000	0.00000	0.00000
4	1.00000	0.00000	0.00000	0.00000	0.00000
5	0.00000	0.00000	1.00000	0.00000	0.00000
6	0.00000	1.00000	0.00000	0.00000	0.00000
7	0.00000	1.00000	0.00000	0.00000	0.00000
8	1.00000	0.00000	0.00000	0.00000	0.00000
9	0.00000	0.00000	0.00000	0.00000	1.00000
10	0.00000	0.00000	1.00000	0.00000	0.00000

Figure.7 – One-Hot Encoding

2. Application of ML&DL Models and Examination of Test Results

2.1 Naive Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Figure.7 – Bayes Theorem

2.1.1 Test Results for Naive Bayes

```
 Naive Bayes: 127 of 240 data were classified correctly.
Accuracy: %52.916666666666664
```

Figure.8 – Accuracy of Naïve Bayes Classifier for test set

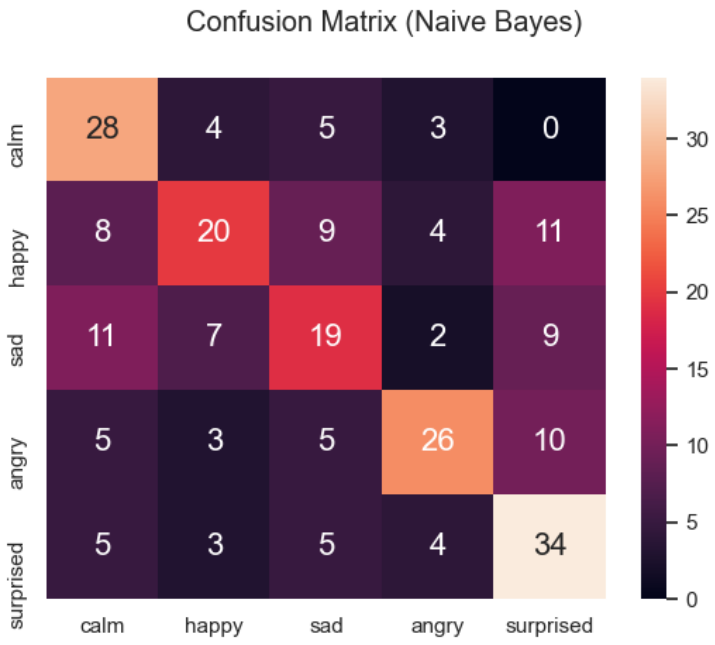

Figure.8 – Confusion Matrix for Naïve Bayes Classifier

We got an accuracy rate of just over %50, and we observed that some emotions were likened to each other ('calm' vs. 'sad') and wrong predictions were made. It's hard to say it's a good estimate rate.

In Confusion Matrix, while column names refer to the actual classes, row names refer to the predicted classes.

2.2 Decision Tree Classifier

Decision tree learning is one of the predictive modelling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

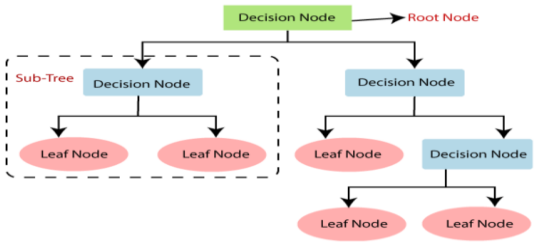

Figure.9 – Decision Tree Example

2.2.1 Test Results for Decision Tree Classifier

```
 Decision Tree: 131 of 240 data were classified correctly.
Accuracy: %54.58333333333333
```

Figure.10 - Accuracy of Decision Tree Classifier for test set

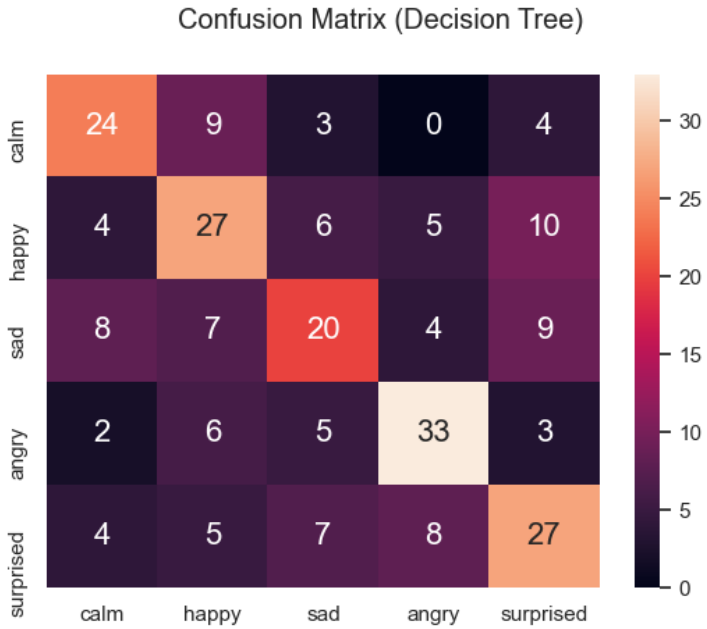

Figure.11 - Confusion Matrix for Decision Tree Classifier

We see that our accuracy rate has increased and there are more accurate predictions in the error matrix than the naïve bayes classifier but our accuracy rate is still just over %50.

2.3 Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean/average prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees.

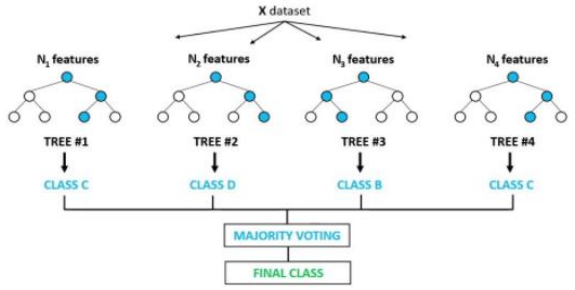

Figure.12 – Random Forest Classifier Example

2.3.1 Test Results for Random Forest Classifier

```
Random Forest: 159 of 240 data were classified correctly.
Accuracy: %66.25
```

Figure.13 - Accuracy of Random Forest Classifier for test set

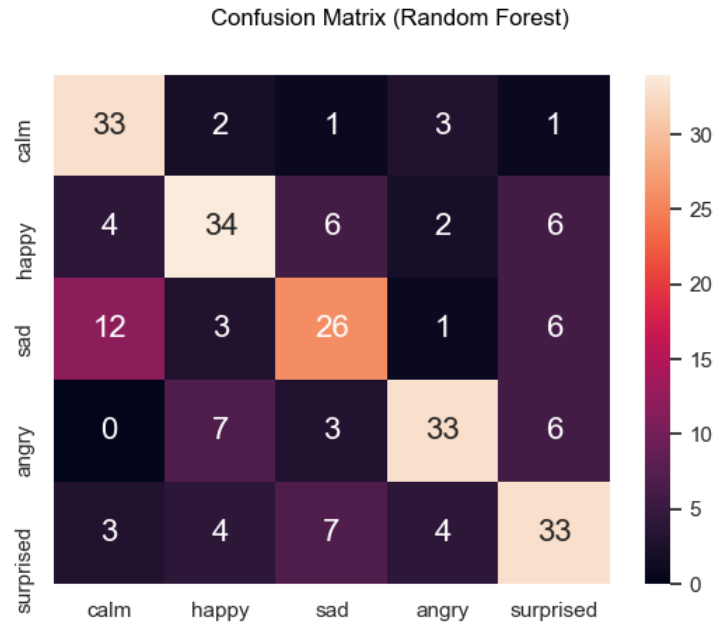

Figure.14 – Confusion Matrix for Random Forest Classifier

We see that we have significantly increased our accuracy rate but we can say that the model still does not predict ‘sad’ and ‘calm’ emotions well.

2.4 Support Vector Machines

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

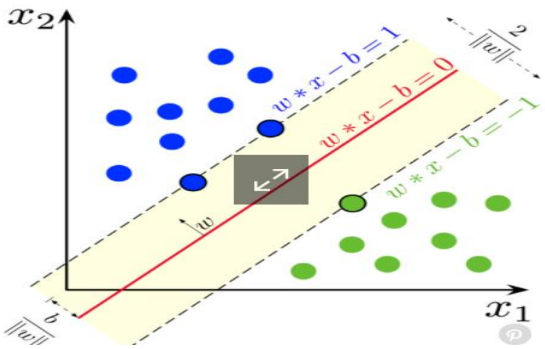

Figure.15 - Maximum-margin hyperplane and margins for an SVM

2.4.1 Test Results for SVM

```
Support Vector Machines: 163 of 240 data were classified correctly.
Accuracy: %67.91666666666667
```

Figure.16 - Accuracy of SVM Classifier for test set

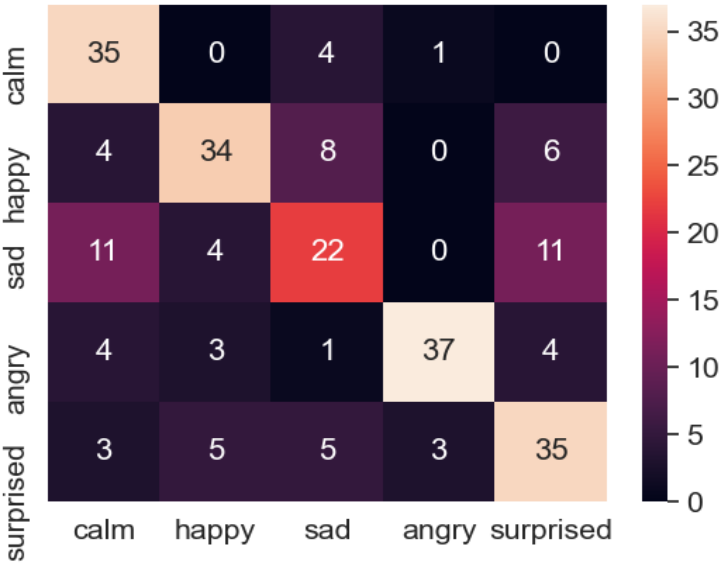

Figure.17 – Confusion Matrix for SVM Classifier

We got a slightly better result than the Random Forest Classifier but when we examine the error matrix, we observe the same problem here. . Now, we will use deep learning model for better results.

2.5 Multilayer Perceptron

A multilayer perceptron (MLP) is a class of feedforward artificial neural network (ANN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

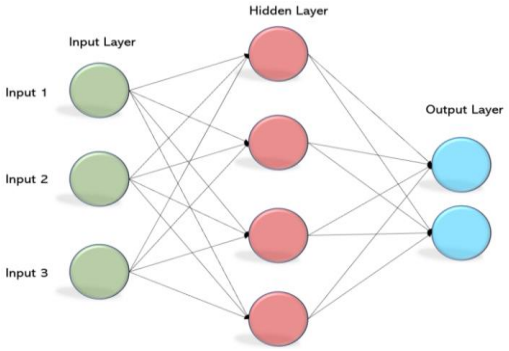

Figure.18 – MLP example include one single hidden layer.

We used 3 different MLP model and got different results:

1- MLP including single hidden layer created using scikit-learn library (Python)
2- MLP including 2 hidden layer created from scratch using numpy library (Python)
3- MLP including 2 hidden layer created using Keras library (Python)

2.5.1 MLP including single hidden layer created using scikit-learn library

```
MLP(single hidden layer): 179 of 240 data were classified correctly.
Accuracy: %74.58333333333333
```

Figure.19 - Accuracy of MLP Classifier (one hidden layer) for test set

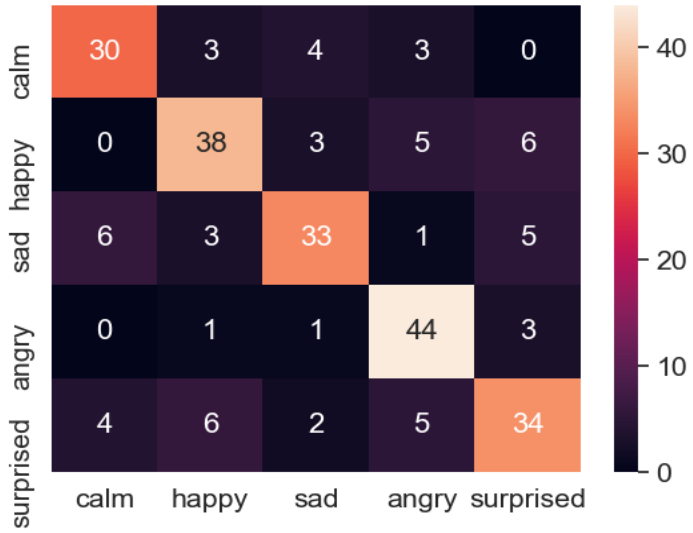

Figure.20 - Confusion Matrix for MLP Classifier (one hidden layer)

2.5.2 MLP including 2 hidden layer created from scratch using numpy library

Learning rate is determined as 0.1 and each hidden layer have 120 neurons. Sigmoid is used as an activation function and PCA is applied for dimensionality reduction. Epoch number is determined as 1000.

```
MLP(2 hidden layer): 183 of 240 data were classified correctly.
Accuracy: %76.25
```

Figure.21 - Accuracy of MLP Classifier (2 hidden layer) for test set

Figure.22 - Confusion Matrix for MLP Classifier (2 hidden layer)

We observe that our accuracy rate increases when we have 2 hidden layers instead single hidden layer. However, this model has a significant drawback: training time. Since we update the weights of all data in our dataset one by one, the training time takes much longer than other models. Now, we will import Keras library and use Keras MLP Class for shorter training time.

2.5.3 MLP including 2 hidden layer created using Keras

This model gives us 3 important advantages:

1- We can determine batch size. Batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration. It leads to faster training.
2- We can create validation set. Validation datasets can be used for regularization by early stopping (stopping training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset).

3- We can add Dropout layer to model. Dropout is a regularization technique for reducing overfitting in artificial neural networks by preventing complex co-adaptations on training data.

As in the previous model, the number of neurons in the hidden layer was determined to be 120. As the activation function, "reLu" is used in hidden layers and "sigmoid" in output layer. Since the dataset size does not affect our training time and results, pca was not applied to the dataset.

```
 MLP(2 hidden layer, Keras): 192 of 240 data were classified correctly.
Accuracy: %80.0
```

Figure.23 - Accuracy of MLP Classifier (2 hidden layer, Keras) for test set

Figure.24 - Confusion Matrix for MLP Classifier (2 hidden layer, Keras)

Our accuracy score is 0.80. This is not a perfect accuracy rate, but we can say the this is best result compared to other models we use. We are able to predict 4 out of every 5 test data correctly.

3. Conclusion

RAVDESS dataset was used to prepare the dataset and create the model. While creating frequency domain features (mfcc, chroma, mel) from the audio file dataset, a study made on a website specified in the bibliography was used. After that, Data Preprocessing is applied (Normalization, Standardization, Dimensionality Reduction, Label and One-Hot Encoding). 4 different ML&DL models were used for classification:

1-Naïve Bayes Classifier: 0.528 Accuracy

2-Decision Tree Classifier: 0.545 Accuracy

3-Random Forest Classifier: 0.662 Accuracy

4-Support Vector Machines: 0.691 Accuracy

5-Multilayer Perceptron: 0.74, 0.76, 0.80 Accuracy

As a result, we obtained an accuracy rate better than the accuracy of the study taken as the source (0.71) and we classified for more emotion. 5 emotions were used as label instead of 4.

4. Bonus

After determining the best model (Keras MLP), we observed the test results for different situations.

4.1 Change the number of data in the test set

We updated the rate of the test set from 0.25 to 0.1. Then, We observed that our accuracy rate increased.

```
 MLP(2 hidden layer, Keras): 80 of 96 data were classified correctly.
Accuracy: %83.33333333333334
```

Figure.25 – Accuracy rate for 96 test data.

Figure.26 – Confusion matrix for 96 test data.

4.2. Change in the number of labels (emotions) to be predicted.

4 Emotions: Calm, Happy, Sad, Angry

```
 MLP(2 hidden layer, Keras): 162 of 192 data were classified correctly.
Accuracy: %84.375
```

Figure.27 – Accuracy rate for 4 emotions.

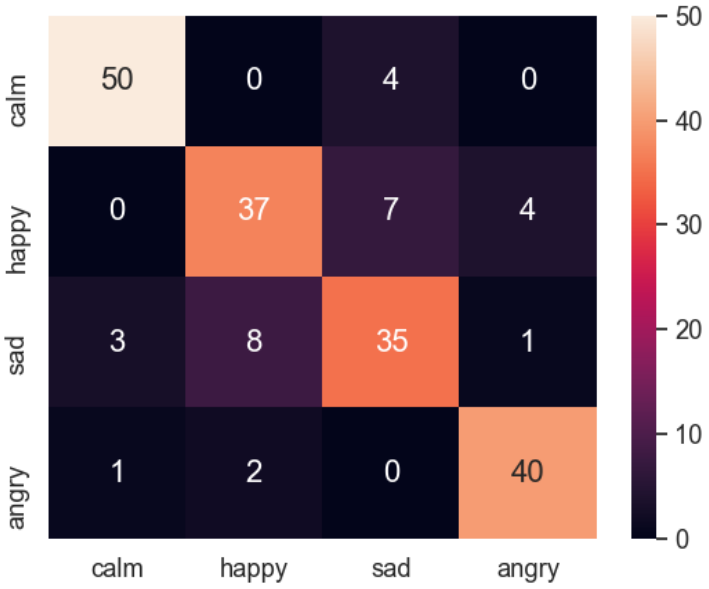

Figure.28 – Confusion matrix for 4 emotions.

8 Emotions: We use all emotions in RAVDESS Dataset

```
 MLP(2 hidden layer, Keras): 244 of 360 data were classified correctly.
Accuracy: %67.77777777777779
```

Figure.29 – Accuracy rate for all emotions.

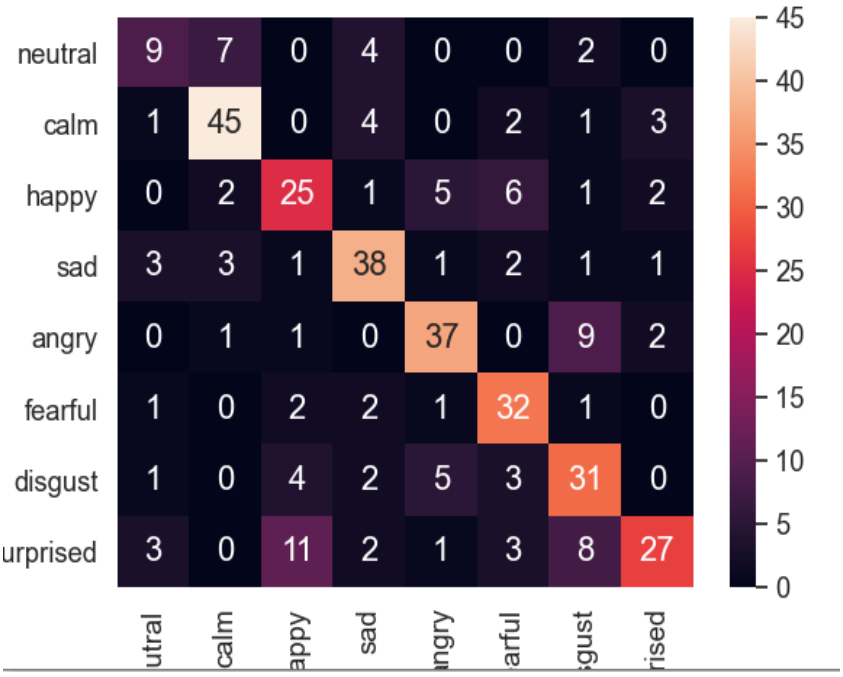

Figure.30 – Confusion matrix for all emotions.

5. References

- https://data-flair.training/blogs/python-mini-project-speech-emotion-recognition/
- https://www.geeksforgeeks.org/naive-bayes-classifiers/
- https://en.wikipedia.org/wiki/Decision_tree
- https://towardsdatascience.com/understanding-random-forest-58381e0602d2
- https://en.wikipedia.org/wiki/Training,_validation,_and_test_sets
- https://en.wikipedia.org/wiki/Multilayer_perceptron
- https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
- https://www.wikiwand.com/en/Support-vector_machine