SPEECH EMOTION RECOGNITION

Data Importing and processing and EDA for data

First we have imported the data and performed the necessary actions on the data like extraction of the emotions from the audio file name.

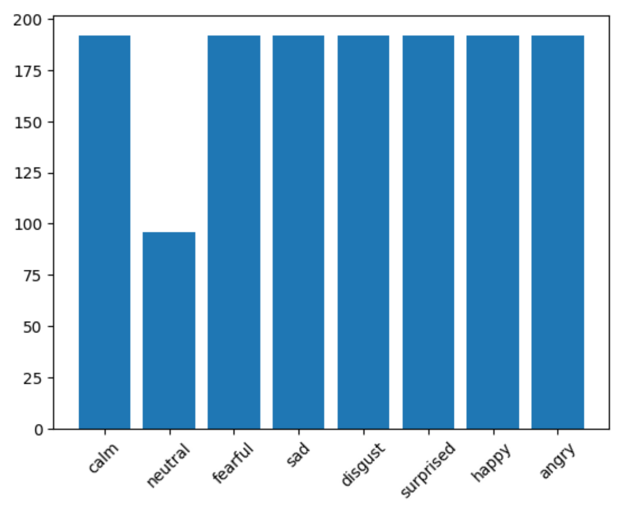
The feature extraction techniques are :

MFCC: MFCCs are a feature extraction technique used to represent the spectral characteristics of an audio signal. The technique involves taking the log of the power spectrum, followed by a Discrete Cosine Transform (DCT) to produce a set of coefficients that represent the spectral envelope of the signal. MFCCs are commonly used in speech recognition applications because they capture important features of human speech, such as formants and harmonics.

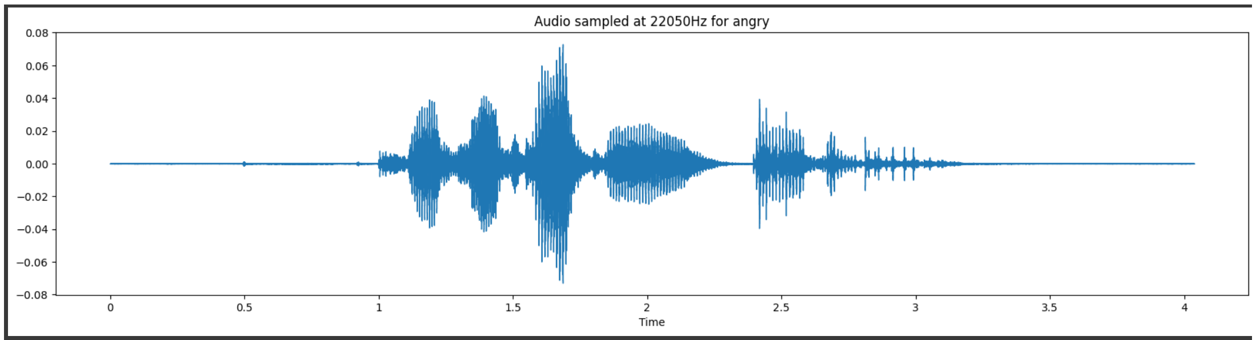
Mel spectrogram: Mel spectrograms are a variation of the traditional spectrogram that use a non-linear frequency scale that better approximates the human auditory system. The spectrogram is divided into frequency bands using a Mel filterbank, which weights the frequency bins according to their perceived loudness. Mel spectrograms are commonly used in music analysis applications because they provide a visual representation of the spectral content of a signal.

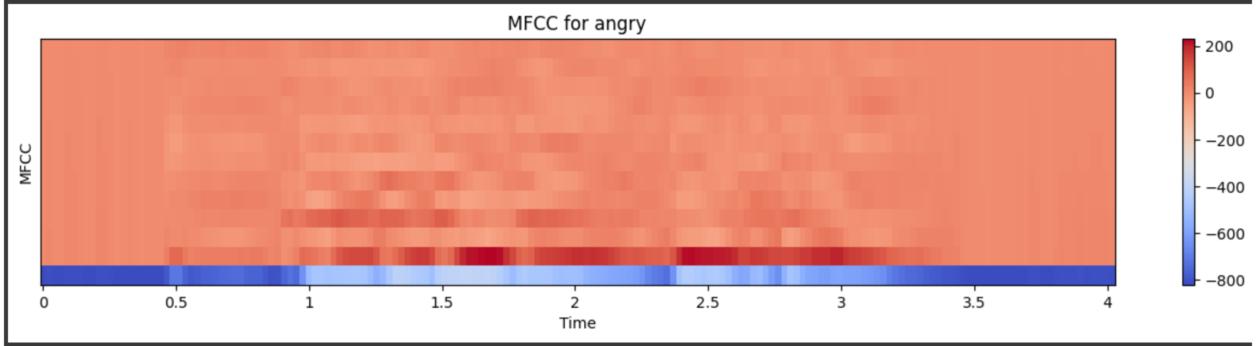
Chroma: Chroma features are a set of features that represent the pitch content of an audio signal. The chroma representation involves dividing the frequency spectrum into 12 bins corresponding to the 12 notes in the Western musical scale, and computing the energy in each bin. Chroma features are commonly used in music analysis applications, such as chord recognition and genre classification.

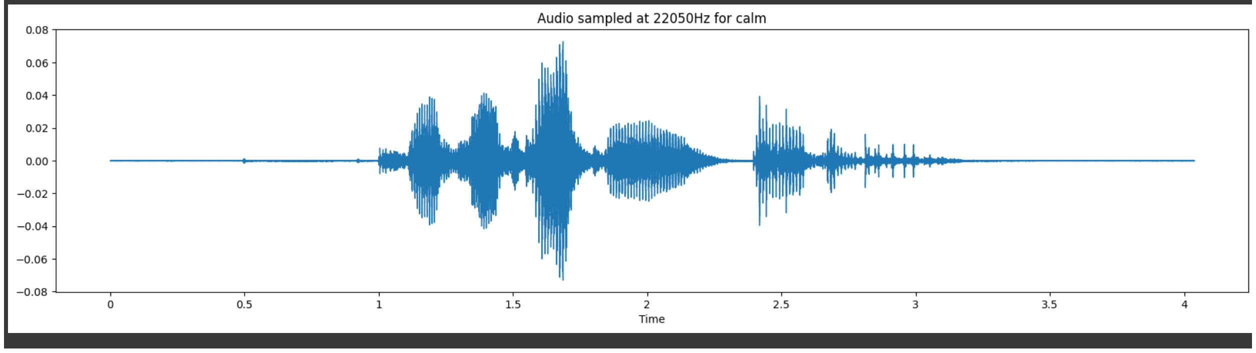
Distribution of classes (emotions) in the dataset given :

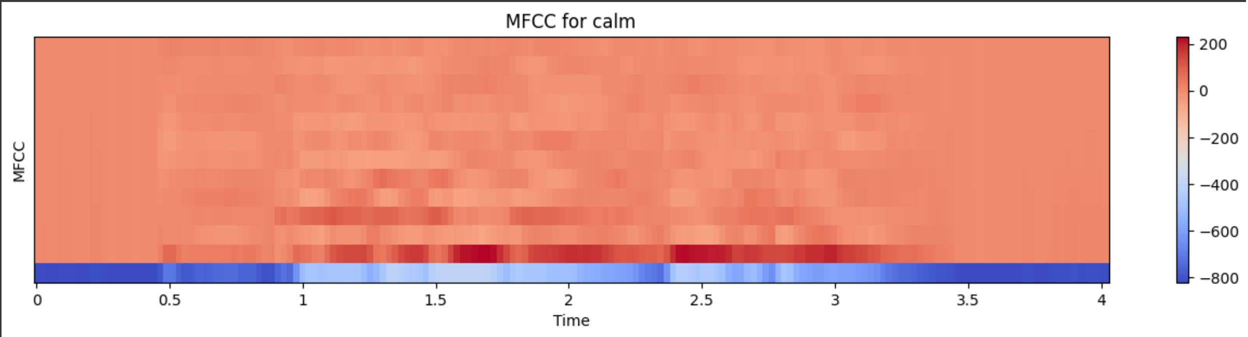


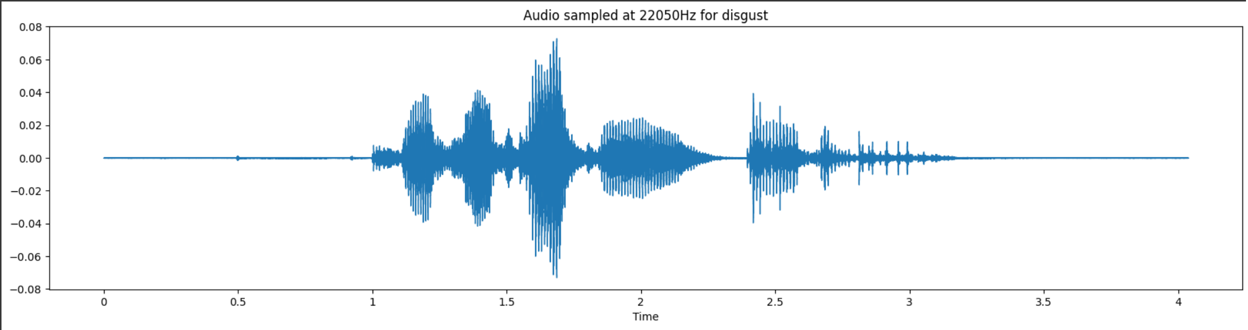
Plotting the mfcc features (one for each of the emotions) :

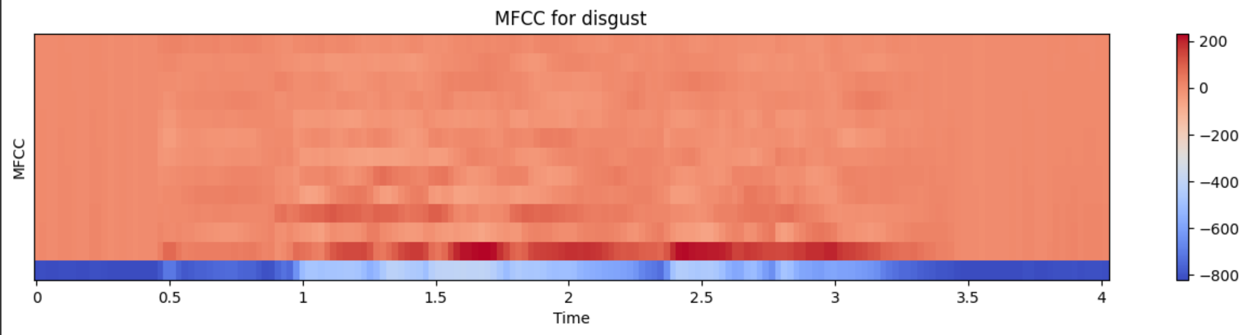


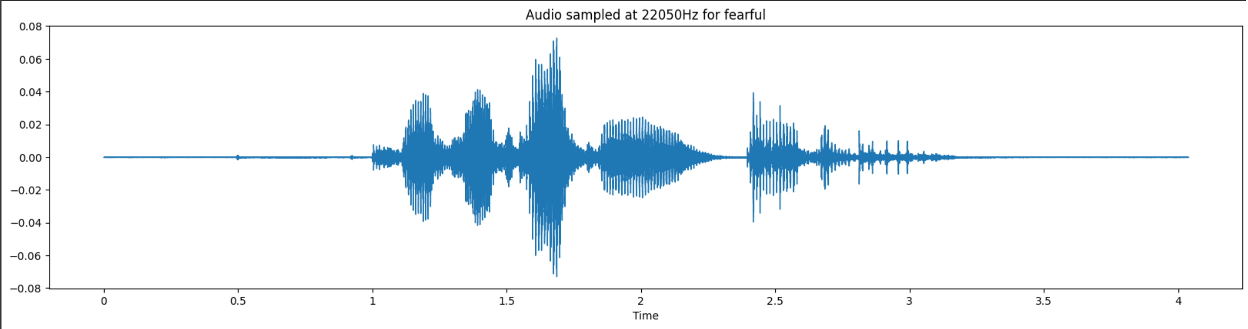


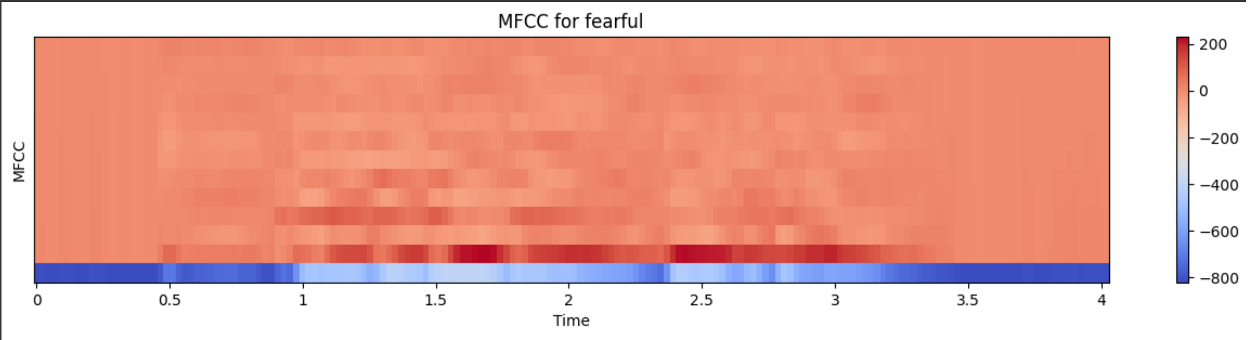


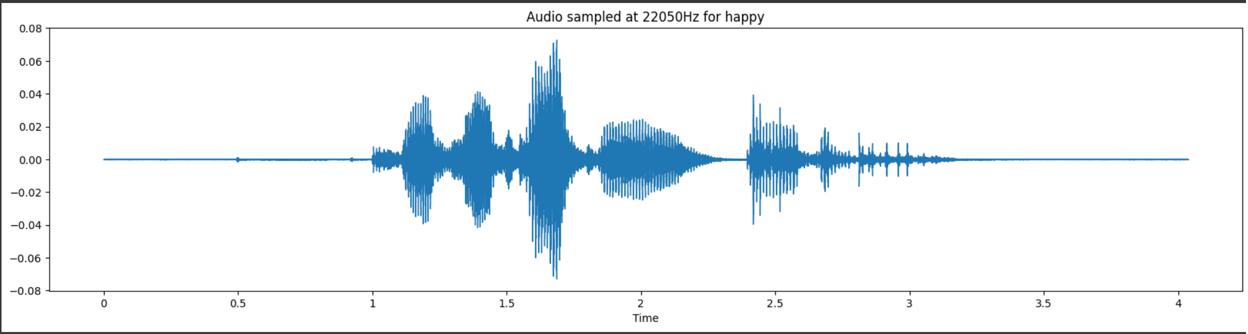

MFCC for calm
MFCC
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
200
0
−200
−400
−600
−800

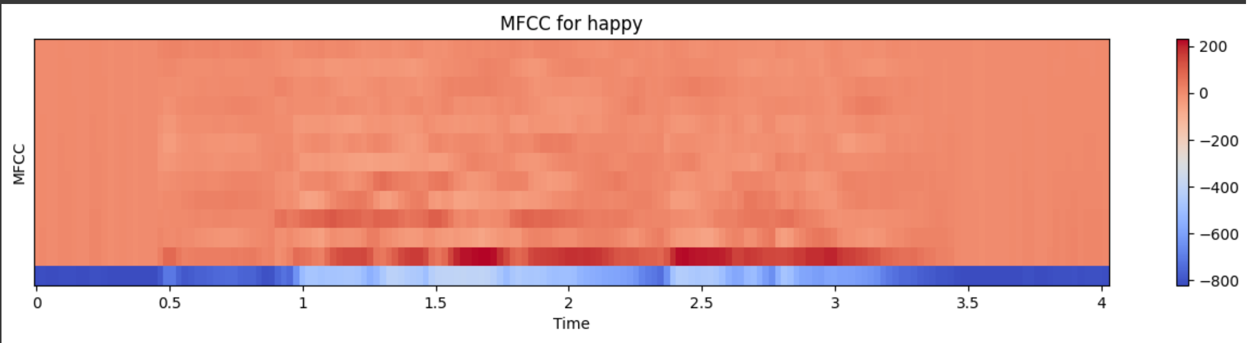

Audio sampled at 22050Hz for disgust
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
0.5
1
1.5
2
2.5
3
3.5
4
Time

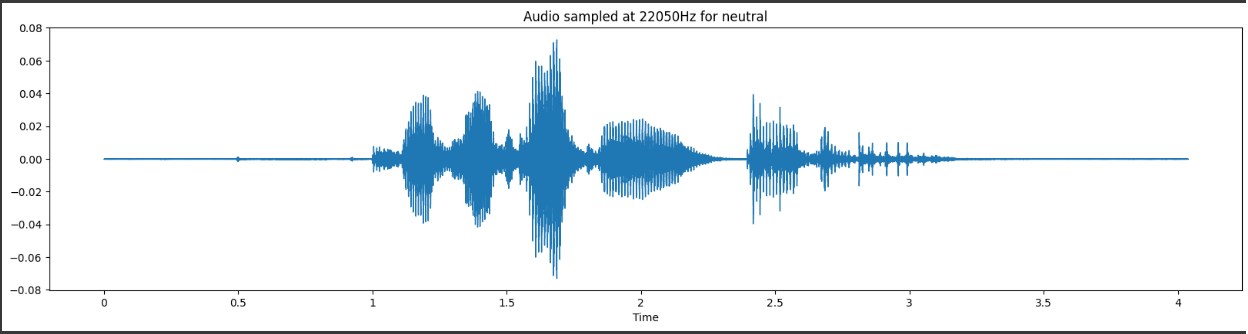

MFCC for disgust
MFCC
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
200
0
−200
−400
−600
−800

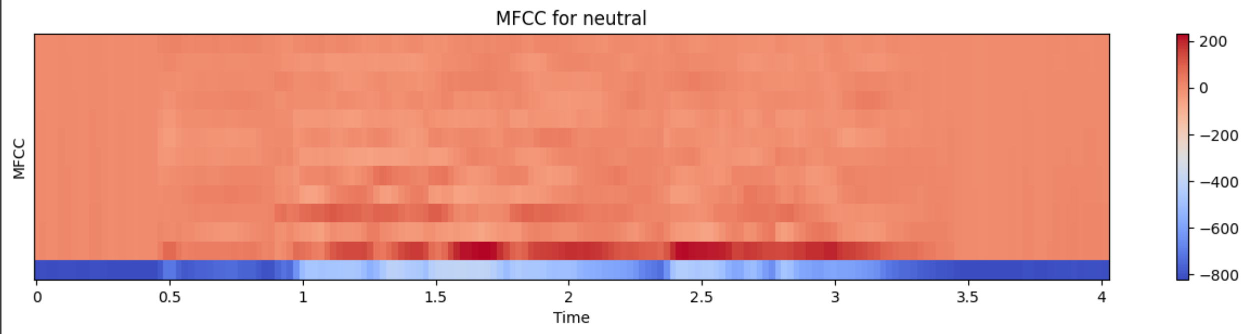

Audio sampled at 22050Hz for fearful
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
0.5
1
1.5
2
2.5
3
3.5
4
Time

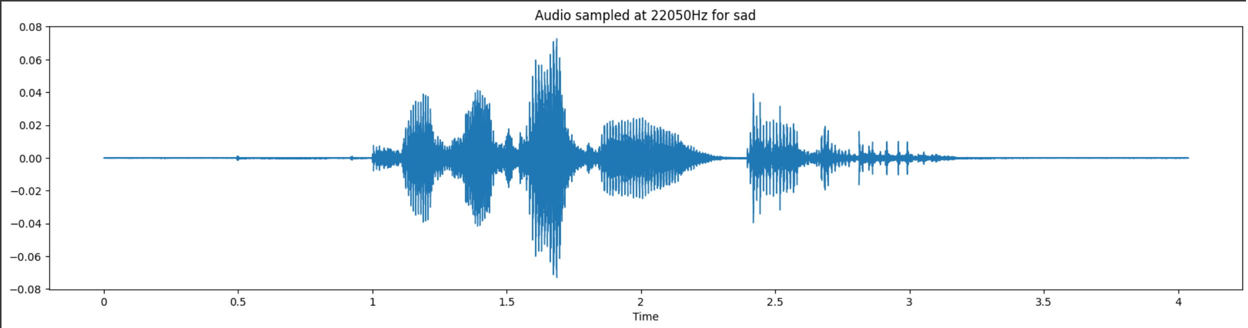

MFCC for fearful
MFCC
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
200
0
−200
−400
−600
−800

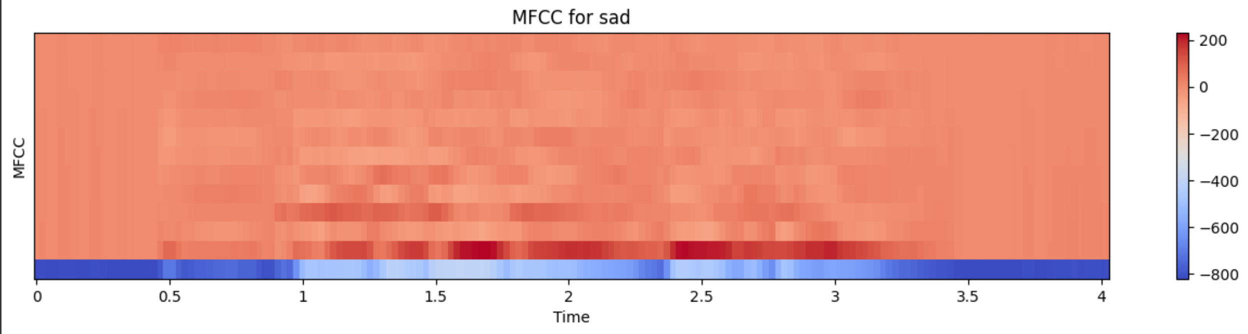

Audio sampled at 22050Hz for happy
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
0.5
1
1.5
2
2.5
3
3.5
4
Time

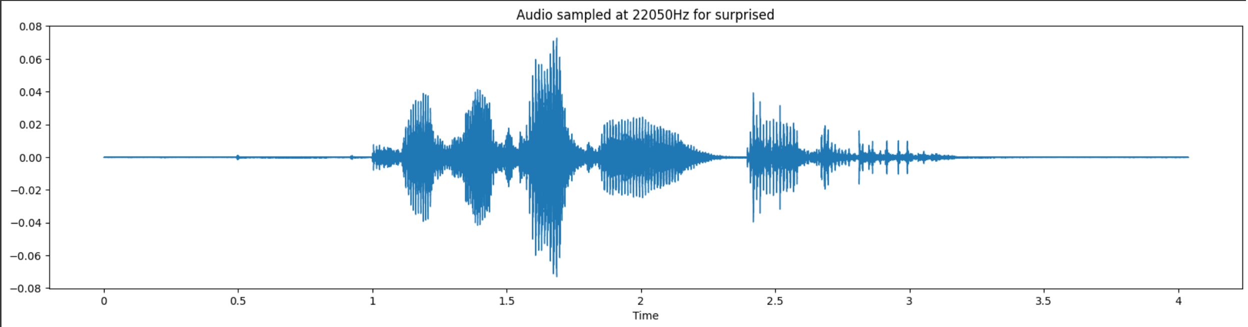

MFCC for happy
MFCC
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
200
0
−200
−400
−600
−800

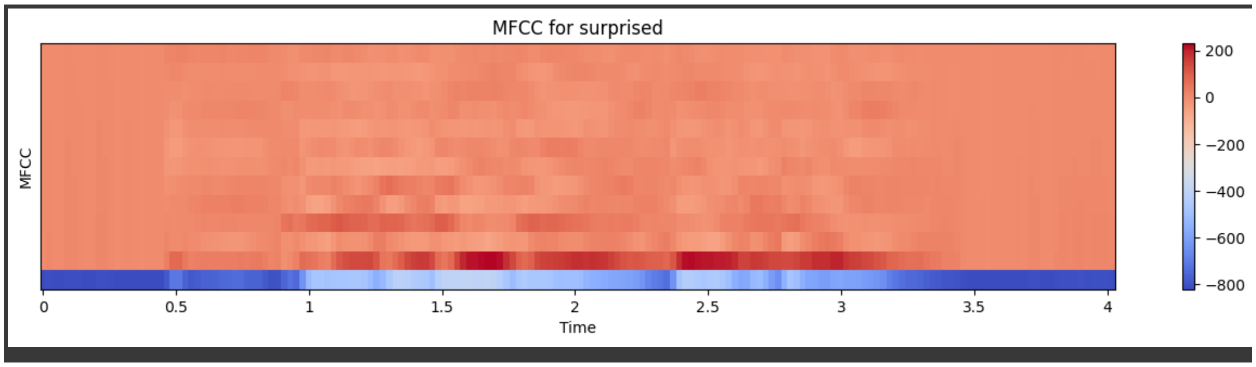

Audio sampled at 22050Hz for neutral
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
0.5
1
1.5
2
2.5
3
3.5
4
Time


MFCC for neutral
MFCC
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
200
0
−200
−400
−600
−800


Audio sampled at 22050Hz for sad
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
0.5
1
1.5
2
2.5
3
3.5
4
Time


MFCC for sad
MFCC
Time
0
0.5
1
1.5
2
2.5
3
3.5
4
200
0
−200
−400
−600
−800


Audio sampled at 22050Hz for surprised
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0
0.5
1
1.5
2
2.5
3
3.5
4
Time



Explanation of codes for each of the models trained :

ANN (ARTIFICIAL NEURAL NETWORKS) :

For training the ann :

The build_model function defines a neural network model with tunable hyperparameters, including the number of hidden layers, the number of units in each layer, the activation function, the dropout rate, and the optimizer. The RandomSearch tuner is used to randomly search the hyperparameter space to find the best set of hyperparameters that optimize the validation accuracy.

After the tuner has finished searching for the best hyperparameters, the best_hps variable contains the best set of hyperparameters found during the search. These hyperparameters are used to build the final neural network model, which is then trained on the full dataset with early stopping to prevent overfitting. Finally, the accuracy of the trained model is evaluated on the test set.

SVM (Support Vector Machine) :

The code imports the necessary libraries: GridSearchCV and SVC from scikit-learn, load_iris from scikit-learn.datasets, train_test_split and accuracy_score from scikit-learn.metrics.

It loads the iris dataset using the load_iris function and splits it into training and test sets using the train_test_split function.

It defines an SVM model with the default hyperparameters, which will be used as a base model for the grid search.

It defines a dictionary of hyperparameters to explore, which includes different values for the regularization parameter C, the kernel parameter gamma, and the kernel function itself (linear, polynomial, radial basis function, and sigmoid).
It creates a GridSearchCV object with the SVM model, the hyperparameter space to explore, and the number of folds for cross-validation (cv=5).
It performs the grid search on the training data using the fit method of the GridSearchCV object. This will try all the combinations of hyperparameters defined in the param_grid dictionary and select the best model based on the cross-validation score.
It gets the best hyperparameters found by the grid search using the best_params_ attribute of the GridSearchCV object.
It gets the best model found by the grid search using the best_estimator_ attribute of the GridSearchCV object.
It evaluates the best model on the training set by making predictions with the predict method and calculating the accuracy using the accuracy_score function.
It evaluates the best model on the test set by making predictions with the predict method and calculating the accuracy using the accuracy_score function.

Decision Tree Classifier , Random Forest Classififer , Bagging , Classifier , KNN

In all these cases the code is as similar to svm the only difference is that the hyper parameters are for corresponding models .

Feature Extraction and Training :

We have used five different methods for feature extraction (using librosa library) :

1) Feature extraction using MFCC

2) Feature extraction using MEL SPECTROGRAM

3) Feature extraction using CHROMA

4) A combination of all the above features

5) Adding some noise , shifting etc to the features obtained by mfcc

For each model we have first performed the hyperparameter tuning using grid search to find the best hyperparameters for training.

1) Feature extraction using MFCC

Mel Frequency Cepstral Coefficients (MFCC) are a set of features commonly used in speech processing and analysis, including emotion recognition. These features are extracted from the audio signal and represent the spectral characteristics of the speech signal. These features are powerful for emotion recognition.

TRAINING USING ANN (ARTIFICIAL NEURAL NETWORKS)

Here we have performed hyperparameter tuning for finding the best hyper parameters using keras_tuner and the best hyper parameters obtained are :

{'units1': 32, 'activation1': 'tanh', 'dropout 1': 0.2, 'units2': 512, 'activation2': 'relu', 'dropout 2': 0.4, 'learning_rate': 0.01, 'num_layers': 1, 'optimizer': 'adam'}

units1: Number of units in the first layer of the neural network. This hyperparameter controls the complexity of the model and the amount of information it can learn.

activation1: Activation function used in the first layer of the neural network. This hyperparameter controls how the input signal is transformed by the first layer.

dropout1: Dropout rate in the first layer of the neural network. This hyperparameter controls the amount of regularization applied to the first layer.

units2: Number of units in the second layer of the neural network. This hyperparameter controls the complexity of the model and the amount of information it can learn.

activation2: Activation function used in the second layer of the neural network. This hyperparameter controls how the output of the first layer is transformed by the second layer.

dropout2: Dropout rate in the second layer of the neural network. This hyperparameter controls the amount of regularization applied to the second layer.

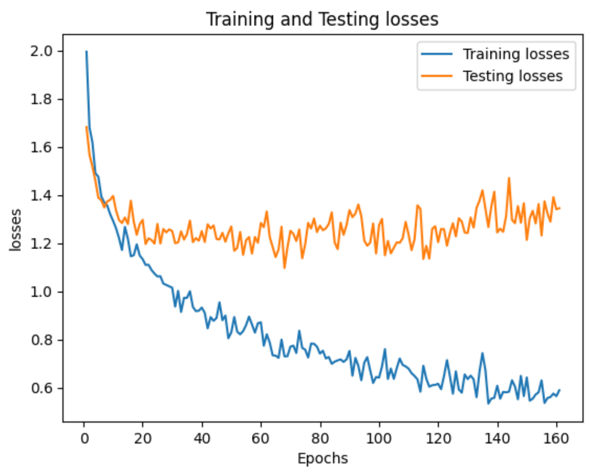
learning_rate: Learning rate of the optimizer used in the neural network. This hyperparameter controls how quickly the model adapts to the training data.

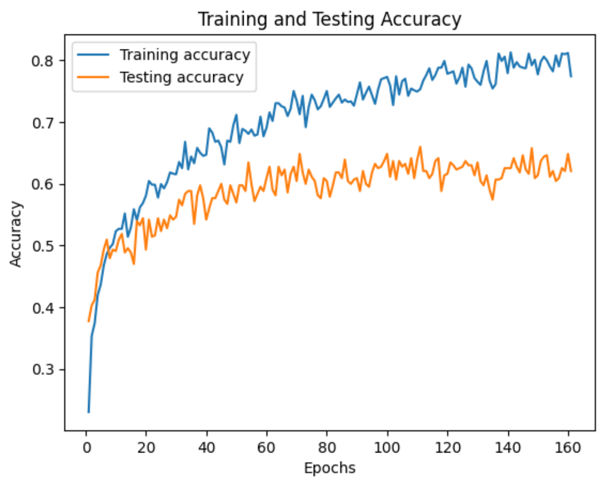
num_layers: Number of layers in the neural network. This hyperparameter controls the depth of the model and the amount of information it can learn.

optimizer: Optimizer used in the neural network. This hyperparameter controls how the model updates its parameters during training.

```
Training accuracy is :  0.9394841194152832
Testing accuracy is :  0.6203703880310059
```

After training the model using ANN we have got the following results :





Classification Report :

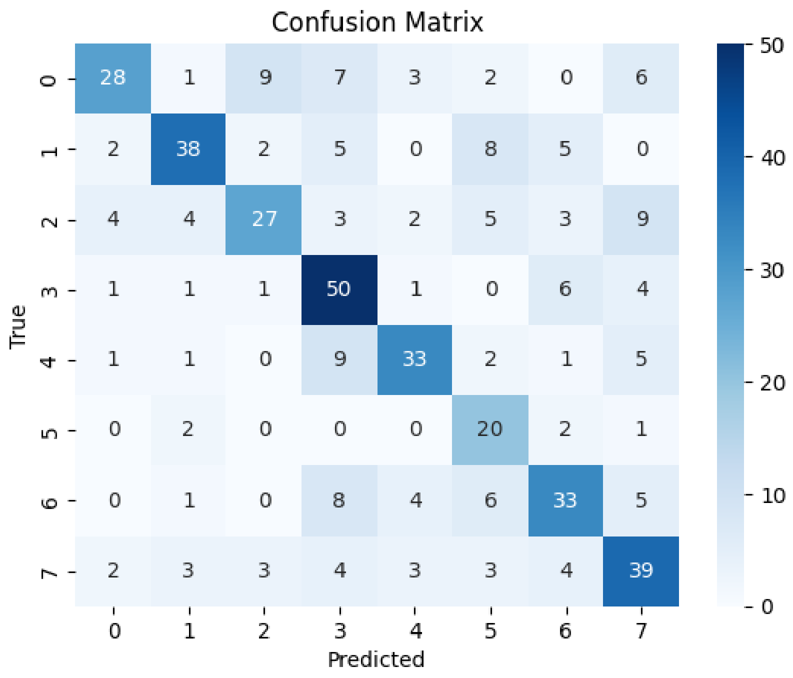
```
              precision    recall  f1-score   support
```

```
         0       0.74      0.50      0.60        56
         1       0.75      0.63      0.68        60
         2       0.64      0.47      0.55        57
         3       0.58      0.78      0.67        64
         4       0.72      0.63      0.67        52
         5       0.43      0.80      0.56        25
         6       0.61      0.58      0.59        57
         7       0.57      0.64      0.60        61

  accuracy                           0.62       432
 macro avg       0.63      0.63      0.62       432
weighted avg       0.64      0.62      0.62       432
```

From the above classification report :

The precision, recall, and F1 score for each class indicates how well model is able to distinguish between the different emotions in the dataset. For example, the precision for class 0 is 0.74, which means that out of all instances predicted as class 0, 74% were actually class 0. The recall for class 0 is 0.5, which means that out of all actual instances of class 0, only 50% were correctly predicted as class 0. The F1 score for class 0 is 0.6, which is the harmonic mean of precision and recall.

Confusion Matrix using ANN :



	0	1	2	3	4	5	6	7
0	28	1	9	7	3	2	0	6
1	2	38	2	5	0	8	5	0
2	4	4	27	3	2	5	3	9
3	1	1	1	50	1	0	6	4
4	1	1	0	9	33	2	1	5
5	0	2	0	0	0	20	2	1
6	0	1	0	8	4	6	33	5
7	2	3	3	4	3	3	4	39

As here we can see that we are overfitting using ANN for this dataset as training accuracy is very high as compared to the testing accuracy .

TRAINING USING SVM (SUPPORT VECTOR MACHINE)

Here first we performed hyper parameter tuning using grid search cv for the following parameters :

```
param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'gamma': [0.0001, 0.001, 0.01, 0.1],
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
}
```

The results obtained are as following :

Classification Report:

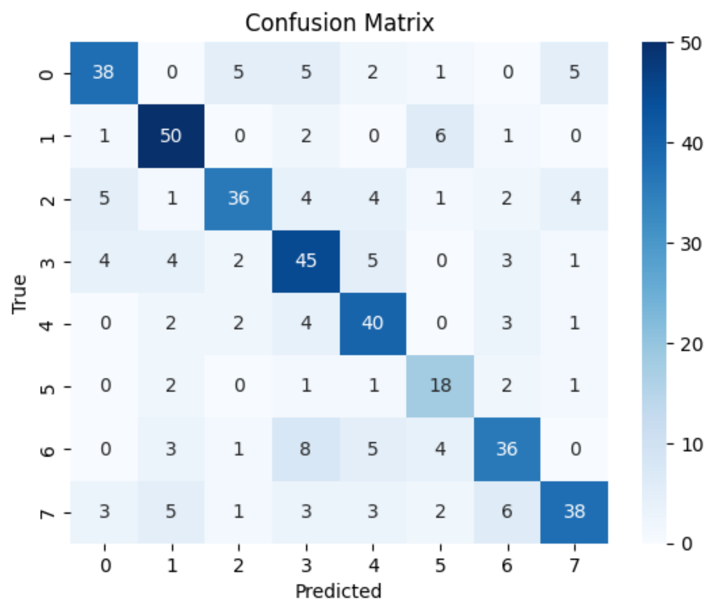
```
Classification Report:
              precision    recall  f1-score   support
```

```
           0       0.75      0.68      0.71        56
           1       0.75      0.83      0.79        60
           2       0.77      0.63      0.69        57
           3       0.62      0.70      0.66        64
           4       0.67      0.77      0.71        52
           5       0.56      0.72      0.63        25
           6       0.68      0.63      0.65        57
           7       0.76      0.62      0.68        61

    accuracy                           0.70       432
   macro avg       0.69      0.70      0.69       432
weighted avg       0.70      0.70      0.70       432
```

The report shows various evaluation metrics such as precision, recall, and F1-score, which are calculated for each class as well as an overall score.

For example, in this report, we can see that for class 0, the precision is 0.75, the recall is 0.68, and the F1-score is 0.71. This means that out of all the instances classified as class 0, 75% were actually class 0 (precision), 68% of all actual class 0 instances were correctly classified (recall), and the harmonic mean of precision and recall (F1-score) is 0.71.

Overall, this report gives us a good idea of how well the model is performing on the dataset. In this case, we can see that the model has an accuracy of 0.70, which means that it correctly predicts the class of 70% of the instances in the testing dataset.



Results obtained from training svm are :

Testing accuracy: 0.6967592592592593

Training accuracy: 1.0

Best hyperparameters: {'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}

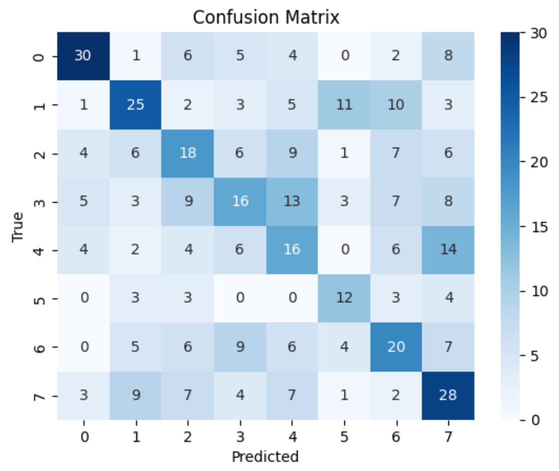
TRAINING USING DECISION TREE CLASSIFIER

Performing hyper parameter tuning we get the following results :

```
Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.54      0.58        56
           1       0.46      0.42      0.44        60
           2       0.33      0.32      0.32        57
           3       0.33      0.25      0.28        64
           4       0.27      0.31      0.29        52
           5       0.38      0.48      0.42        25
           6       0.35      0.35      0.35        57
           7       0.36      0.46      0.40        61
```

```
    accuracy                           0.38       432
   macro avg       0.39      0.39      0.39       432
weighted avg       0.39      0.38      0.38       432
```

Confusion Matrix is :



Results obtained are :

```
Training accuracy is :  0.9583333333333334
Testing accuracy is :  0.3819444444444444
```

The classification report shows the precision, recall, and F1-score for each class along with their support (number of samples) in the test set. From the report, we can see that the model is not performing well and is not able to predict the classes accurately. The testing accuracy is also very low, which further confirms this.

Moreover, the training accuracy is very high (close to 1), indicating that the model has overfit the training data and is not able to generalize well to new unseen data.

TRAINING USING KNN

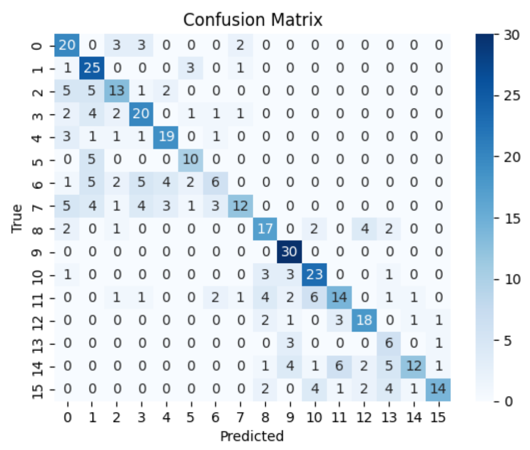
After hyper parameter tuning we have the following results :

```
Testing accuracy: 0.6111111111111112
Training accuracy:  0.8154761904761905
Best hyperparameters:  {'n_neighbors': 2, 'p': 2}
```

Classification Report :

```
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.71      0.59        28
           1       0.51      0.83      0.63        30
           2       0.54      0.50      0.52        26
           3       0.57      0.65      0.61        31
           4       0.68      0.73      0.70        26
           5       0.59      0.67      0.62        15
           6       0.46      0.24      0.32        25
           7       0.71      0.36      0.48        33
           8       0.59      0.61      0.60        28
           9       0.70      1.00      0.82        30
          10       0.64      0.74      0.69        31
          11       0.58      0.42      0.49        33
          12       0.69      0.69      0.69        26
          13       0.32      0.60      0.41        10
          14       0.80      0.38      0.51        32
          15       0.82      0.50      0.62        28

    accuracy                           0.60       432
   macro avg       0.61      0.60      0.58       432
weighted avg       0.62      0.60      0.59       432
```

Confusion matrix :

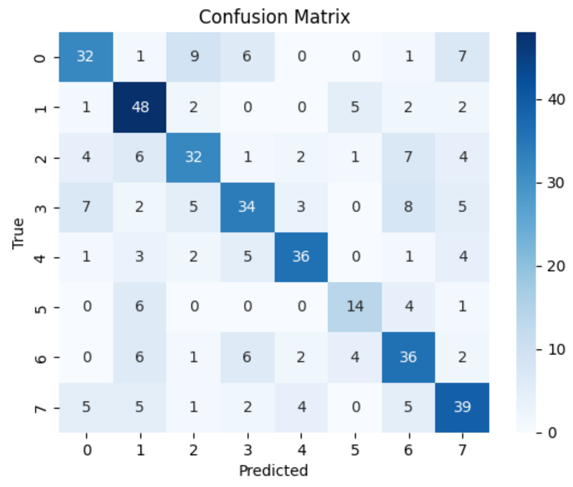


Training using Random Forest Classifier :

We have following results –

```
Testing accuracy: 0.6273148148148148
Training accuracy:  1.0
Best hyperparameters:  {'criterion': 'gini', 'max_depth': 20, 'max_features': 'auto', 'n_estimators': 500}
Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.57      0.60        56
           1       0.62      0.80      0.70        60
           2       0.62      0.56      0.59        57
           3       0.63      0.53      0.58        64
           4       0.77      0.69      0.73        52
           5       0.58      0.56      0.57        25
           6       0.56      0.63      0.60        57
           7       0.61      0.64      0.62        61

    accuracy                           0.63       432
   macro avg       0.63      0.62      0.62       432
weighted avg       0.63      0.63      0.63       432
```

Confusion Matrix :



Training with Bagging Classifier :

The following results are obtained here :

```
Testing accuracy: 0.5694444444444444
Training accuracy:  1.0
Best hyperparameters:  {'base_estimator__max_depth': 15, 'max_features': 0.5, 'max_samples': 0.9,
'n_estimators': 200}
```

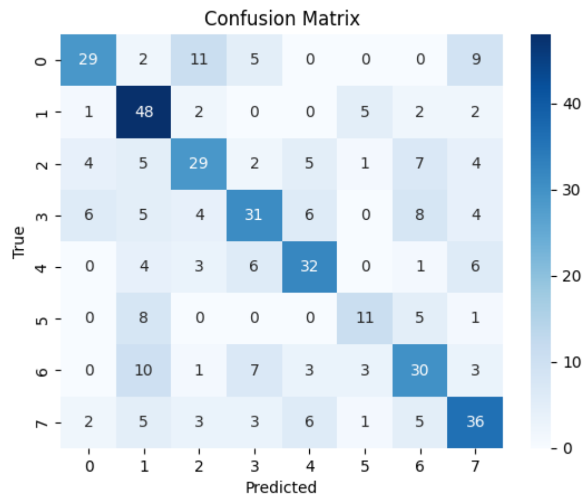
Classification report:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.52      0.59        56
           1       0.55      0.80      0.65        60
           2       0.55      0.51      0.53        57
           3       0.57      0.48      0.53        64
           4       0.62      0.62      0.62        52
           5       0.52      0.44      0.48        25
```

```
       6      0.52      0.53      0.52        57

       7      0.55      0.59      0.57        61
```

```
  accuracy                        0.57       432

 macro avg      0.57      0.56      0.56       432

weighted avg      0.57      0.57      0.57       432
```

<u>Confusion Matrix :</u>



True \ Predicted	0	1	2	3	4	5	6	7
0	29	2	11	5	0	0	0	9
1	1	48	2	0	0	5	2	2
2	4	5	29	2	5	1	7	4
3	6	5	4	31	6	0	8	4
4	0	4	3	6	32	0	1	6
5	0	8	0	0	0	11	5	1
6	0	10	1	7	3	3	30	3
7	2	5	3	3	6	1	5	36

Here from this method of feature selection we have got the maximum accuracy as :

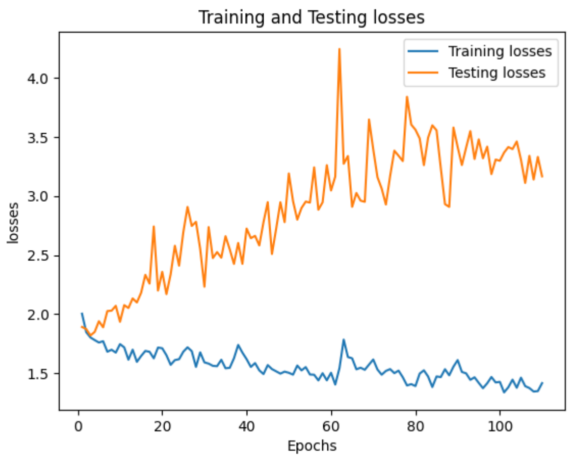
Model	Training Accuracy	Testing Accuracy
ANN	0.9394841194152832	0.6203703880310059
SVM	1.0	0.6967592592592593
Decision Tree	0.9583333333333334	0.3819444444444444
Random Forest	1.0	0.6273148148148148

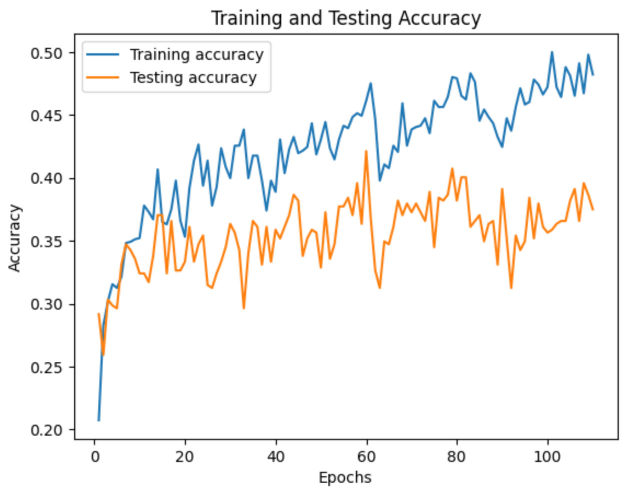
Bagging	1.0	0.5694444444444444
KNN	0.8154761904761905	0.6111111111111112

2) Feature Extraction using Mel Spectrogram

ANN :
Best hyper parameters are :

```
{'units1': 32, 'activation1': 'tanh', 'dropout1': 0.2, 'units2': 512, 'activation2': 'relu',
'dropout2': 0.4, 'learning_rate': 0.01, 'num_layers': 1, 'optimizer': 'adam'}
```





```
Classification Report:
              precision    recall  f1-score   support

           0       0.57      0.52      0.54        56
           1       0.35      0.60      0.44        60
           2       0.28      0.37      0.32        57
           3       0.58      0.33      0.42        64
           4       0.54      0.25      0.34        52
           5       0.00      0.00      0.00        25
           6       0.22      0.18      0.19        57
           7       0.33      0.52      0.41        61

    accuracy                           0.38       432
   macro avg       0.36      0.35      0.33       432
```

```
weighted avg       0.39      0.38      0.36       432
```

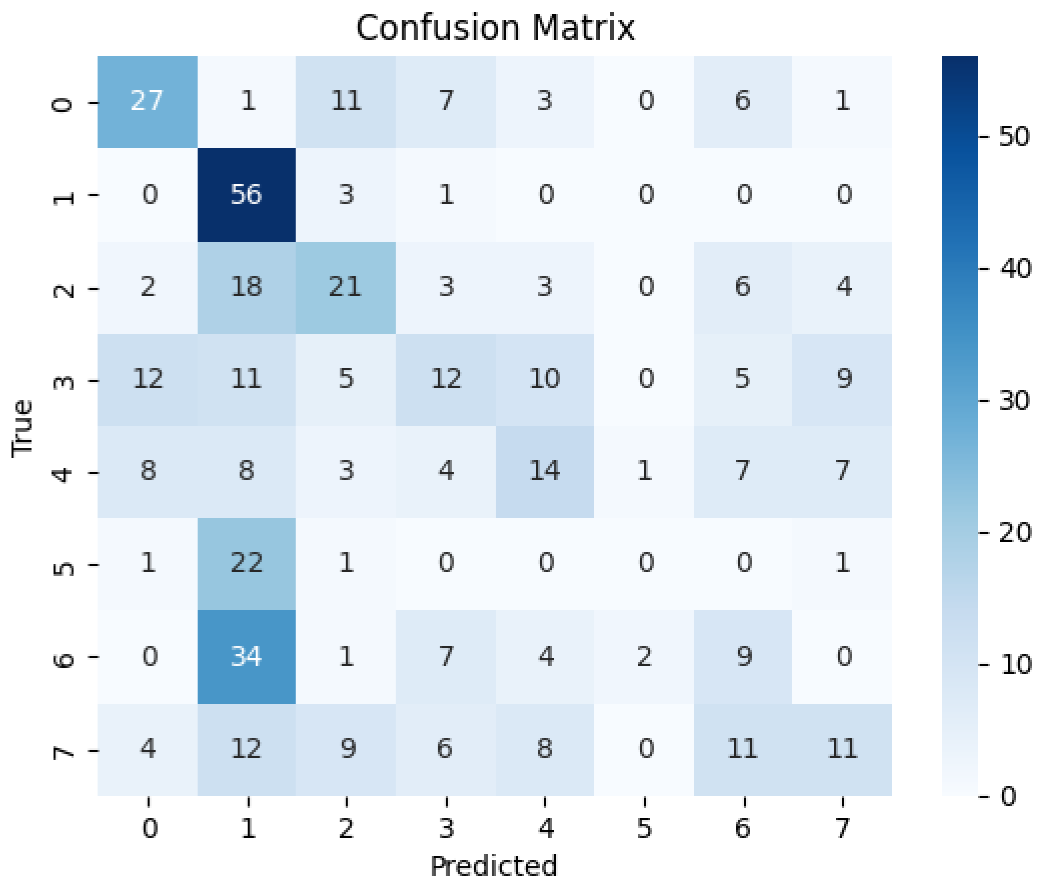
Thus we have following :

```
Training accuracy is :  0.5466269850730896
Testing accuracy is :  0.375
```

SVM :

```
Testing accuracy: 0.3472222222222222
Training accuracy:  0.6001984126984127
Best hyperparameters:  {'C': 10, 'gamma': 0.0001, 'kernel': 'linear'}
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.50      0.48      0.49        56
           1       0.35      0.93      0.50        60
           2       0.39      0.37      0.38        57
           3       0.30      0.19      0.23        64
           4       0.33      0.27      0.30        52
           5       0.00      0.00      0.00        25
           6       0.20      0.16      0.18        57
           7       0.33      0.18      0.23        61

    accuracy                           0.35       432
   macro avg       0.30      0.32      0.29       432
weighted avg       0.32      0.35      0.31       432
```



Here only some of the model and their details are shown , they can be checked into colab file attached with this file.

Model	**Training Accuracy**	**Testing Accuracy**
ANN	0.5466269850730896	0.375
SVM	0.6001984126984127	0.3472222222222222
Decision Tree	0.8690476190476191	0.28703703703703703
Random Forest	1.0	0.5138888888888888
KNN	0.7261904761904762	0.4050925925925926

3) TRAINING THE MODEL USING CHROMA FEATURES :

The results obtained are :

Model	**Training Accuracy**	**Testing Accuracy**
ANN	0.3789682686328888	0.18518517911434174
SVM	0.9037698412698413	0.19212962962962962
Decision Tree	0.2619047619047619	0.19675925925925927
KNN	0.42857142857142855	0.1736111111111111

4) Combined Features

Model	**Training Accuracy**	**Testing Accuracy**
ANN	0.8859127163887024	0.5208333333333334
SVM	1.0	0.6481481481481481
Decision Tree	0.7738095238095238	0.3101851851851852
KNN	0.8412698412698413	0.5787037037037037

5) PERFORMING AUDIO AUGMENTATION

Here we are adding extra inputs to data by adding noise , shift and pitch .
Noise augmentation: This involves adding noise to an audio signal to simulate real-world noise sources that may be present during model deployment.
Pitch augmentation: This involves altering the pitch of an audio signal, while maintaining the same tempo and duration. This can help the model learn to recognize the same sound at different pitches, which may occur due to changes in the speaker's age or gender, or due to different recording devices.

Shift augmentation: This involves altering the timing of an audio signal, by shifting it forward or backward in time. This can help the model learn to recognize the same sound at different time points, which may occur due to changes in the speaker's cadence, or due to variations in the recording environment.
Overall now we have our data more diverse and more information about the dataset.

The results obtained from this data after transforming :

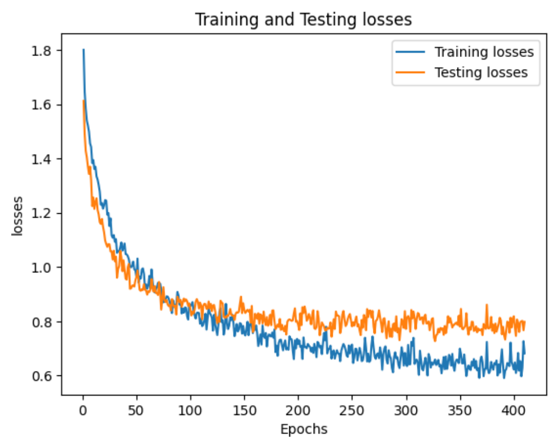
Model	**Training Accuracy**	**Testing Accuracy**
ANN	0.932539701461792	0.7353395223617554
SVM	0.9821428571428571	0.7175925925925926
Decision Tree	0.9203042328042328	0.3904320987654321
KNN	0.8373015873015873	0.6003086419753086
ANN with PCA	0.5794	0.6273148059844971

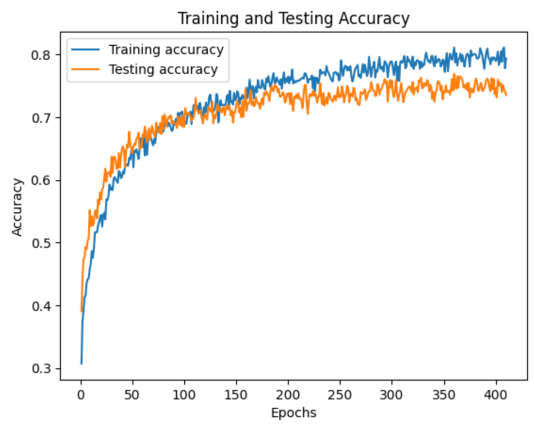
Analyzing all the models trained, we can see that we have got the highest accuracy for ANN with feature extraction of audio augmentation.

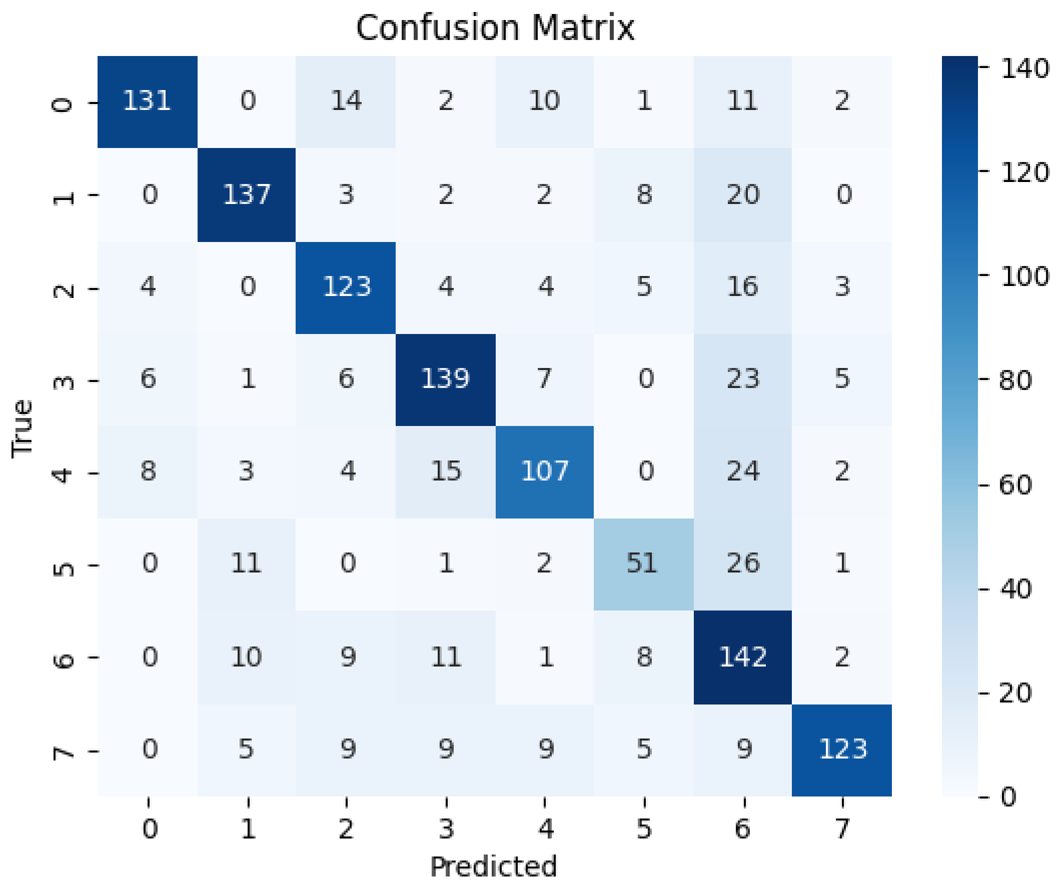
For the best model :

Best hyper parameters are :

```
{'units1': 256,
 'activation1': 'relu',
 'dropout1': 0.4,
 'units2': 64,
 'activation2': 'relu',
 'dropout2': 0.30000000000000004,
 'learning_rate': 0.01,
 'num_layers': 1,
 'optimizer': 'adam'}
```







True \ Predicted	0	1	2	3	4	5	6	7
0	131	0	14	2	10	1	11	2
1	0	137	3	2	2	8	20	0
2	4	0	123	4	4	5	16	3
3	6	1	6	139	7	0	23	5
4	8	3	4	15	107	0	24	2
5	0	11	0	1	2	51	26	1
6	0	10	9	11	1	8	142	2
7	0	5	9	9	9	5	9	123

```
Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.77      0.82       171
           1       0.82      0.80      0.81       172
           2       0.73      0.77      0.75       159
           3       0.76      0.74      0.75       187
           4       0.75      0.66      0.70       163
           5       0.65      0.55      0.60        92
           6       0.52      0.78      0.63       183
           7       0.89      0.73      0.80       169

    accuracy                           0.74      1296
   macro avg       0.75      0.72      0.73      1296
weighted avg       0.76      0.74      0.74      1296
```

Conclusions:

1. MFCC features worked best for this data for speech recognition while others performed poor.
2. MFCC with some extra noise , shifting and pitch augmentation worked better than that of originally MFCC because in this case now the data has more diversity and can handle unseen data better so having higher testing accuracy than that with MFCC features originally.
3. Overall, among all the trained models, the artificial neural network (ANN) performed the best.
4. Moreover , SVM gave an accuracy approximately equal to a small difference but in case of SVM there is overfitting with 100% training accuracy and testing accuracy around 70% .
5. Decision Tree is not working good which may be due to decision tree easily overfits or as we have only a subset of data which was not able to give much information to form a better decision tree or may be lack of linearity (Decision tree works better on linearly separable dataset) .
6. Random Forest and Bagging Classifier are also working better here in this case only for training data (causes overfitting) and they took too much time to train so all the accuracies are not available. But there is overfitting with these and also low testing accuracy.
7. Hence we have got the best accuracy for ANN with data added with some noise , pitch and shift.