

SPEECH EMOTION RECOGNITION

Project submitted to

Shri Ramdeobaba College of Engineering & Management, Nagpur

in partial fulfillment of requirement for the award of

degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

By

Ms. Pooja Upadhye

Mr. Akhilesh Mishra

Mr. Devansh Sahu

Mr. Aryan Bhangadiya

Mr. Harsh Upadhye

Guide

Prof. Aarti Karandikar



Computer Science and Engineering

Shri Ramdeobaba College of Engineering & Management, Nagpur 440013

(An Autonomous Institute affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

June 2023

SHRI RAMDEOBABA COLLEGE OF ENGINEERING & MANAGEMENT, NAGPUR

(An Autonomous Institute Affiliated to Rashtrasant Tukdoji Maharaj Nagpur University Nagpur)

Department of Computer Science and Engineering(Data Science)

CERTIFICATE

This is to certify that the Thesis on “**Speech Emotion Recognition**” is a bonafide work of

1. Ms. Pooja Upadhye
2. Mr. Akhilesh Mishra
3. Mr. Devansh Sahu
4. Mr. Harsh Upadhye
5. Mr. Aryan Bhangadiya

submitted to the Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur in partial fulfillment of the award of a Bachelor of Technology, in Computer Science and Engineering (Data Science). It has been carried out at the Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur during the academic year 2022-2023.

Date: 03-06-2023

Place: Nagpur

Prof. Aarti Karandikar

Project Guide

Dr Avinash Agarwal

H.O. D

Department of Computer
Science and Engineering

DECLARATION

I, hereby declare that the thesis titled “Speech Emotion Recognition” submitted herein, has been carried out in the Department of Computer Science and Engineering of Shri Ramdeobaba College of Engineering & Management, Nagpur. The work is original and has not been submitted earlier as a whole or part for the award of any degree / diploma at this or any other institution / University.

Date: 03-06-2023

Place: Nagpur

Ms. Pooja Upadhye

(Roll No.: 11)

Mr. Akhilesh Mishra

(Roll No.: 34)

Mr. Devansh Sahu

(Roll No.: 42)

Mr. Harsh Upadhye

(Roll No.: 73)

Mr. Aryan Bhangadiya

(Roll No.: 70)

APPROVAL SHEET

This report entitled Speech Emotion Recognition by Akhilesh Mishra, Aryan Bhangadiya, Devansh Sahu, Harsh Upadhye and Pooja Upadhye is approved for the degree of B Tech CSE Data Science.

Name & signature of Supervisor(s)

Name & signature of External Examiner(s)

Name & signature RRC Members

Name & signature of HOD

Date:

Place:

ACKNOWLEDGEMENT

It has been a pleasant journey filled with learning opportunities and experiences, where we finally arrive at the end of this wonderful project work. We wish to express gratitude to all those who have blessed and supported us whenever needed.

Special Thanks to our Guide Prof. Aarti Karandikar, who calmly aided us step by step throughout our project with their wonderful insights and unmatched experience by leading us close towards the goal. From the very initial stage to its end, we gained a great deal of knowledge related to the working and processes that can be combined to obtain the needed results.

We express our sincere gratitude to Dr. Avinash Agrawal, Head of the Department of Computer Science Department, RCOEM for his guidance, which brought a positive influence in our work and towards its success.

Each member of the group is thankful for receiving such a wonderful opportunity and we aim to reach greater heights in the future.

Ms. Pooja Upadhye

(Roll No.: 11)

Mr. Akhilesh Mishra

(Roll No.: 34)

Mr. Devansh Sahu

(Roll No.: 42)

Mr. Harsh Upadhye

(Roll No.: 73)

Mr. Aryan Bhangadiya

(Roll No.: 70)

ABSTRACT

The purpose of cognitive speech theory is to determine emotion in speech by analyzing music. Speech Recognition solutions are developed using machine learning and deep learning. adopts a new method involving multiple machine learning algorithms using ensemble learning to classify recorded messages in real time. The integration of advanced features into the training data during training of the ML model affects current theoretical thinking and leads to significant false positives. Modern models often address the problem of training large body sizes. On the other hand, today's models are more focused on the integration of various classifications; this can improve the accuracy of cognitive recognition even if the training corpus contains sorry items. "Mass Learning with High Dimensional Acoustic Features (EL-HDAF)" is a new design method that uses the evaluation of the variance of features in different classes to recommend the best features. Experimental results show that classification of needs can yield better results than a single model using the weighted majority classification method. This study presents performance measurements of individual samples for each recording, including background noise, and uses a full sample of all three recordings. Evaluation is done to select the best hyperparameter configuration for the model to evaluate the performance of the learning process by majority vote. The overall performance of the learning set reached an accuracy of 66.5%, which was 65.7% better classification accuracy than the MLP model. Artificial Neural Network (ANN) models, especially Convolutional Neural Networks (CNN) are used for transmission-based spectrograms and mel-spectrograms. The most accurate model was trained for six emotional intelligences, reaching 57. The combined rate of the four databases (CREMA-D, RAVDESS, SAVEE, TESS) is 42%. More importantly, another study showed that misclassified data between the training set and the test set can affect the test accuracy of CNN.

TABLE OF CONTENTS

COVER PAGE

THESIS CERTIFICATE----- i

DECLARATION ----- ii

APPROVAL SHEET ----- iii

ACKNOWLEDGEMENTS----- iv

ABSTRACT----- v

TABLE OF CONTENTS----- vi

LIST OF FIGURES----- vii

CHAPTER 1 : INTRODUCTION ----- 1

1.1 Problem Statement ----- 1

1.2 Aim and Objectives ----- 1

1.3 Problem Specification ----- 1

1.4 Background ----- 2

CHAPTER 2 : IMPLEMENTATION ----- 3

2.1 Recurrent Neural Network (RNN) ----- 3

2.2 Long Short-term Memory (LSTM) ----- 5

2.3 Support Vector Machine (SVM) ----- 8

2.4 Random Forest (RF) ----- 12

2.5 K-Nearest Neighbors (KNN) ----- 16

2.6 Multi Layer Perceptron (MLP) ----- 19

2.7 Ensemble Model ----- 20

Chapter 3 : Result and Discussion	24
3.1 Result	24
3.2 Advantages	27
3.3 Future Scope	28
REFERENCES	29

LIST OF FIGURES

FIGURE 1: Recurrent Neural Network Accuracy	3
FIGURE 2 : Recurrent Neural Network Confusion Matrix	4
FIGURE 3 : Recurrent Neural Network Model Performance Plot	5
FIGURE 4 : Recurrent Neural Network with Long Short-Term Memory Model Performance Plot	7
FIGURE 5 : Recurrent Neural Network with Long Short-Term Memory Confusion Matrix	8
FIGURE 6 : Support Vector Machine Accuracy	10
FIGURE 7 : Support Vector Machine Model Performance Plot	11
FIGURE 8 : Random Forest Accuracy	14
FIGURE 9 : Random Forest Model Performance Plot	15
FIGURE 10 : K-Nearest Neighbors Accuracy	17
FIGURE 11 : K-Nearest Neighbors Model Performance Plot	18
FIGURE 12 : Multi Layer Perceptron Accuracy	20
FIGURE 13 : Pipeline	21
FIGURE 14 : Ensemble Accuracy with output	23
FIGURE 15 : Ensemble Model Confusion Matrix	23
FIGURE 16: Flowchart	24

CHAPTER 1 : INTRODUCTION

In this chapter we will be discussing the technology used for speech-emotion recognition, different Machine Learning (ML) and Deep Learning (DL) models and various research and literature papers and reviews available and the machine specifications for running the project properly. Basically we will be training and testing various models which can be used for speech-emotion recognition and we will select a model which classifies the emotion more accurately than other models. Also we will be adding visual diagrams which will help in comparing different used models. After model selection we will try to enhance its accuracy. Various factors affect the accuracy of models like: pre-processing, feature engineering, feature selection(quantity, quality,etc) from audio file, model compatibility with given data, etc. so understanding these factors well and improving accuracy is our aim.

By leveraging the power of feature engineering and the capabilities of advanced machine learning techniques, SER can contribute to applications such as affective computing, human-computer interaction, and sentiment analysis in various domains.

1.1 Problem Statement

Employing ML techniques for analysis of speech emotion recognition. We will be given speech/ audio and we have to classify the emotions of audio. Audio is a time series data, to identify emotion from audio we have to extract its intrinsic properties which can be done via features of audio. Our aim is to understand feature engineering for our project.

Once the features are extracted, the next step is to select an appropriate machine

learning model for classification,selection of the best suitable model,which depends on factors such as the complexity of the data, availability of labeled training data, and computational resources, and finally improve the accuracy of selected models.

1.2 Aim and Objectives

Effective recognition of emotions by speech via machine. Identification is done with the help of various machine learning and deep learning models using python. Approach is done by extracting vocal features from speech from a dataset or database.

Objectives:

1. To make machines learn to detect emotions from speech.
2. To provide automation in emotion recognition without human intervention.
3. To improve recognition accuracy.
4. To provide a valuable output to users.

1.3 Problem Specification

Using an existing pre-built model instead of creating a whole package from scratch. This gives us a head start towards achieving our goals and saves time. And we have also increased the accuracy of pre-built models. For speech-emotion recognition we have various machine learning as well as deep learning models available for us. Out of them we have selected those which are compatible with time series data. Considered models are Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Multi Layer Perceptron (MLP) and ensemble model.

We trained them and tested our data on them, and we have selected an ensemble model (RF + SVM + KNN + MLP) for emotion recognition.

1.4 Background

Introduction to emotion recognition

Speech is one of the important features which reflects the underlying emotions of a human being. There are various factors in speech which helps in revealing a person's state. Given an audio file, our aim is to identify the emotion by extracting features from audio. Recognition is done with the help of hidden features in file, features include Zero crossing rate(ZCR), Spectral centroid(SC), Spectral Rolloff(SR), Spectral bandwidth(SB), Mel-frequency cepstrum coefficient(MFCC), Pitch, Harmony, Root Mean Square (RMS), etc. out of which we have selected 3 features for extraction and they are MFCC, Pitch and RMS.

CHAPTER 2 : IMPLEMENTATION

So we tried to implement various different Machine learning models with different features extracting from the RAVDESS dataset to see which features work better together to achieve higher accuracy.

2.1 Recurrent Neural Network (RNN)

The model defines an RNN model using the Sequential API from Keras. It extracts MFCC (Mel-frequency cepstral coefficients) features using the librosa library. It consists of three SimpleRNN layers with dropout layers in between. The output layer is a Dense layer with softmax activation. The training set consists of 80% of the data, and the test set contains the remaining 20%. The training is performed for 50 epochs with a batch size of 32. The testing accuracy came not more than 48%.

```
Epoch 47/50
58/58 [=====] - 2s 31ms/step - loss: 1.2644 - accuracy: 0.5453 - val_loss: 1.4429 - val_accuracy: 0.4794
Epoch 48/50
58/58 [=====] - 2s 31ms/step - loss: 1.2329 - accuracy: 0.5556 - val_loss: 1.4330 - val_accuracy: 0.4620
Epoch 49/50
58/58 [=====] - 2s 31ms/step - loss: 1.2158 - accuracy: 0.5578 - val_loss: 1.3778 - val_accuracy: 0.4967
Epoch 50/50
58/58 [=====] - 2s 31ms/step - loss: 1.2171 - accuracy: 0.5627 - val_loss: 1.3708 - val_accuracy: 0.5054
18/18 [=====] - 0s 10ms/step - loss: 1.4470 - accuracy: 0.4878
Test accuracy: 0.4878472089767456
```

FIGURE 1 : RNN Accuracy

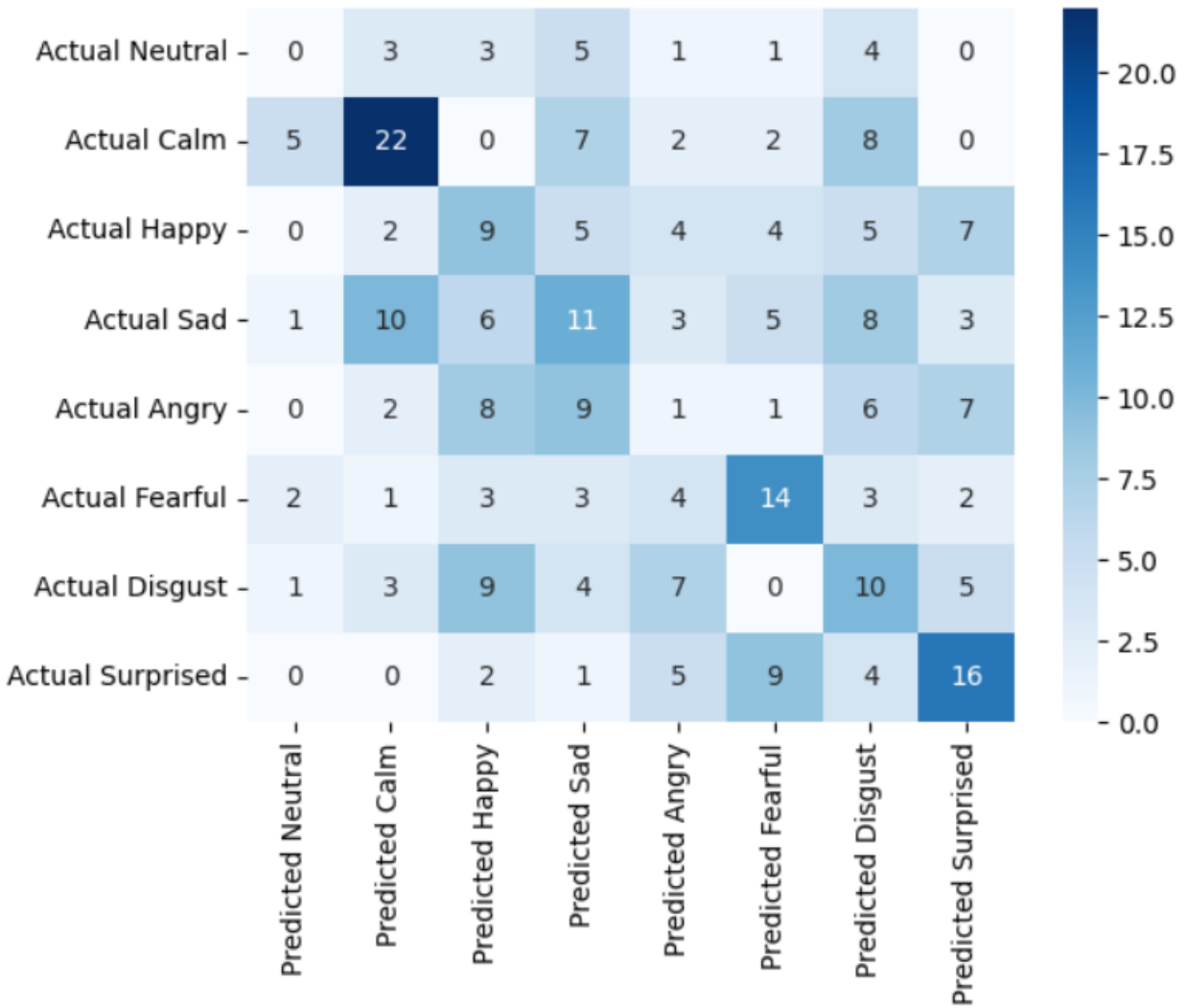


FIGURE 2 : RNN Confusion Matrix

Above is the confusion matrix for RNN model

By testing various epochs, we were able to observe that in simple RNN, the training accuracy was able to increase with the epoch but validation accuracy was not increasing.

Plot of Rnn model on graph with training and validation accuracy is as shown below :

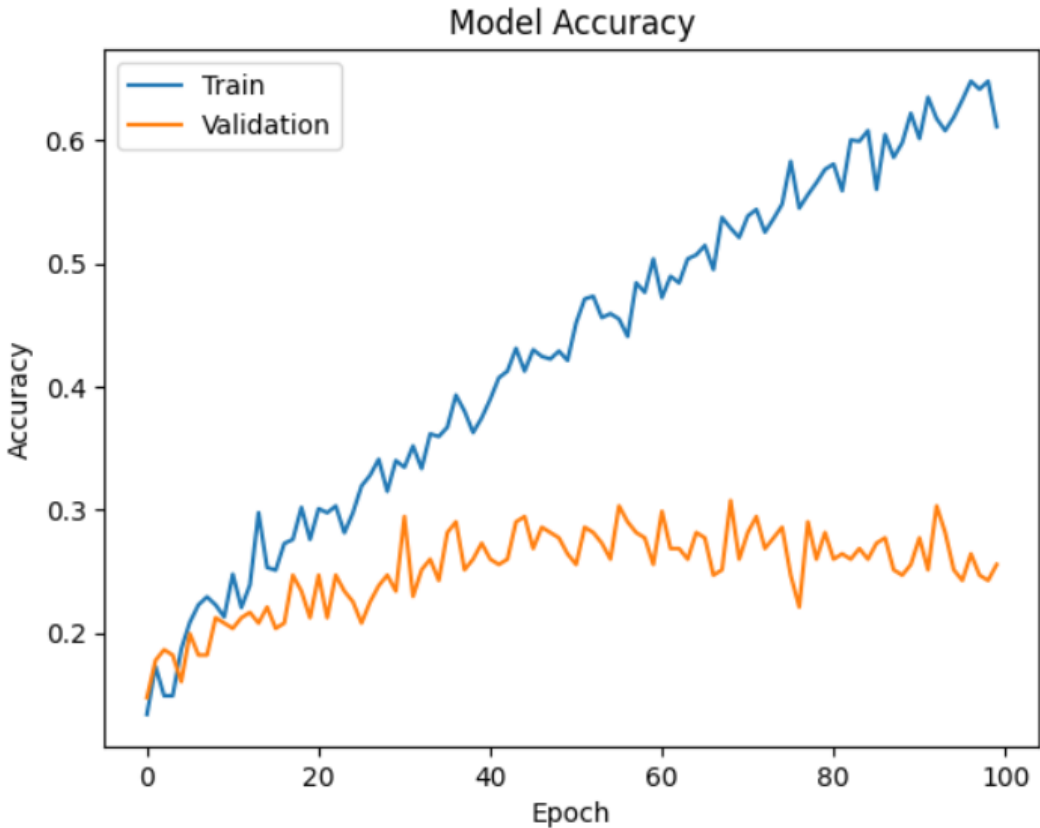


FIGURE 3 : RNN Model Performance Plot

2.2 Long Short-Term Memory (LSTM)

A hybrid model is used to check if this works well on the given dataset which was RNN-LSTM hybrid model. Deep learning or deep structured learning can be defined as a special kind of neural network composed of multiple layers. These networks are better than traditional neural networks in persisting the information from previous events. Recurrent neural network (RNN) is one such machine that has a combination of networks in a loop. The networks in the loop allow the information to persist. Each network in the loop takes input and information from the previous network, performs the specified operation and produces output along with passing the information to the next network. Some applications require only recent information while others may ask for more from the past. The common recurrent neural networks lag in learning as the gap between required previous information and the point of requirement increases to a large extent.

But fortunately Long Short Term Memory (LSTM) Networks [18], a special form of RNN, are capable of learning such scenarios. These networks are precisely designed to escape the long term dependency issue of recurrent networks. LSTMs are good at remembering information for a long time. Since more previous information may affect the accuracy of model, LSTMs become a natural choice of use

Here, some of the important libraries used were :

- 1.librosa : audio signal processing, including loading and manipulating audio files, extracting features, and transforming audio signals.
- 2.wave : reading and writing audio files in WAV format.
- 3.sklearn : library used for training and evaluating classifiers.
4. Keras : building and training neural networks.
5. RMSprop : optimize the model during training

Here, mfcc feature was extracted from the model. Firstly a dataset was loaded which included 1440 files(24 actors x 60 files each) and 8 emotions to be classified.

Data was splitted in train test as 80% for training and 20% for testing part

- Layers used were :

Dense : reduce the dimensionality of the output from the LSTM layer.

Dropout : prevent overfitting

But here, validation accuracy was very low. this could be due to several reasons like :

1. Insufficient training data: The RAVDESS dataset consists of 1440 files, which may not be enough to train a complex model like LSTM. Deep learning models typically require a large amount of data to generalize well. Increasing the size of the training set or considering data augmentation techniques could help improve the accuracy.

2. Data representation: In the given code, the MFCC features are extracted from the audio files. However, the MFCC representation may not capture all the relevant information for emotion classification.
3. Overfitting: The model may be overfitting to the training data, resulting in poor generalization to unseen data. Applying regularization techniques like dropout or early stopping can help mitigate overfitting.

Generated Graph :

As here, training accuracy is increasing as the number of epochs are set to 100 or 150 but validation accuracy is not increasing above 35.

Following is the RNN-LSTM model accuracy shown in the graph which shows training and validation accuracy plot for the number of epochs the model is trained for.

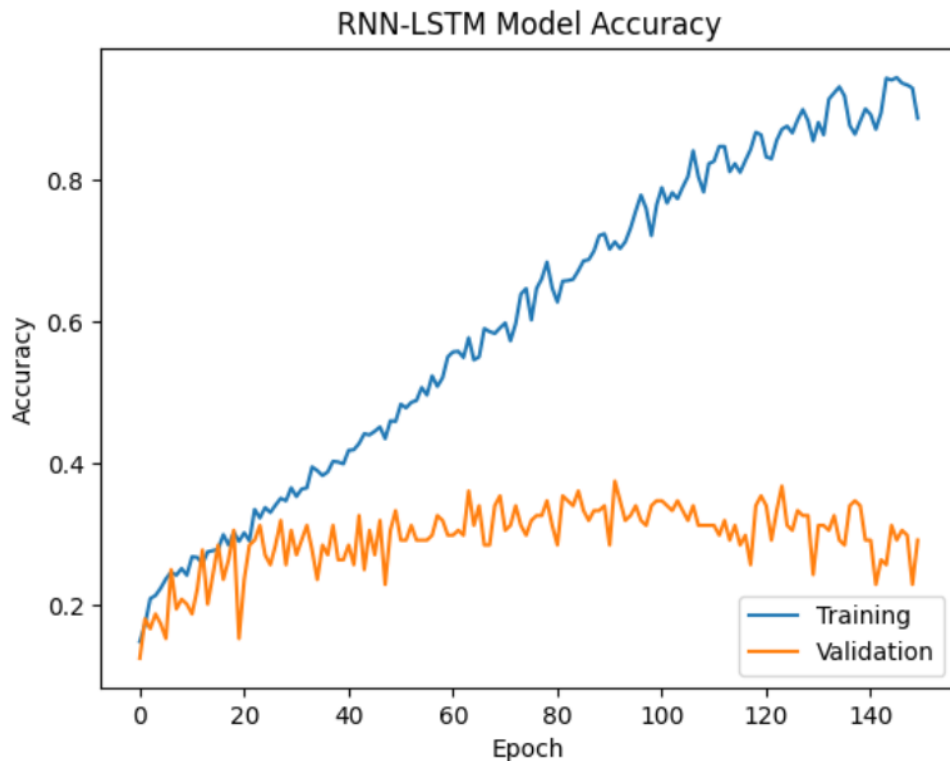


FIGURE 4 : RNN_LSTM Model Performance Plot

This is the confusion matrix of rnn-lstm model :

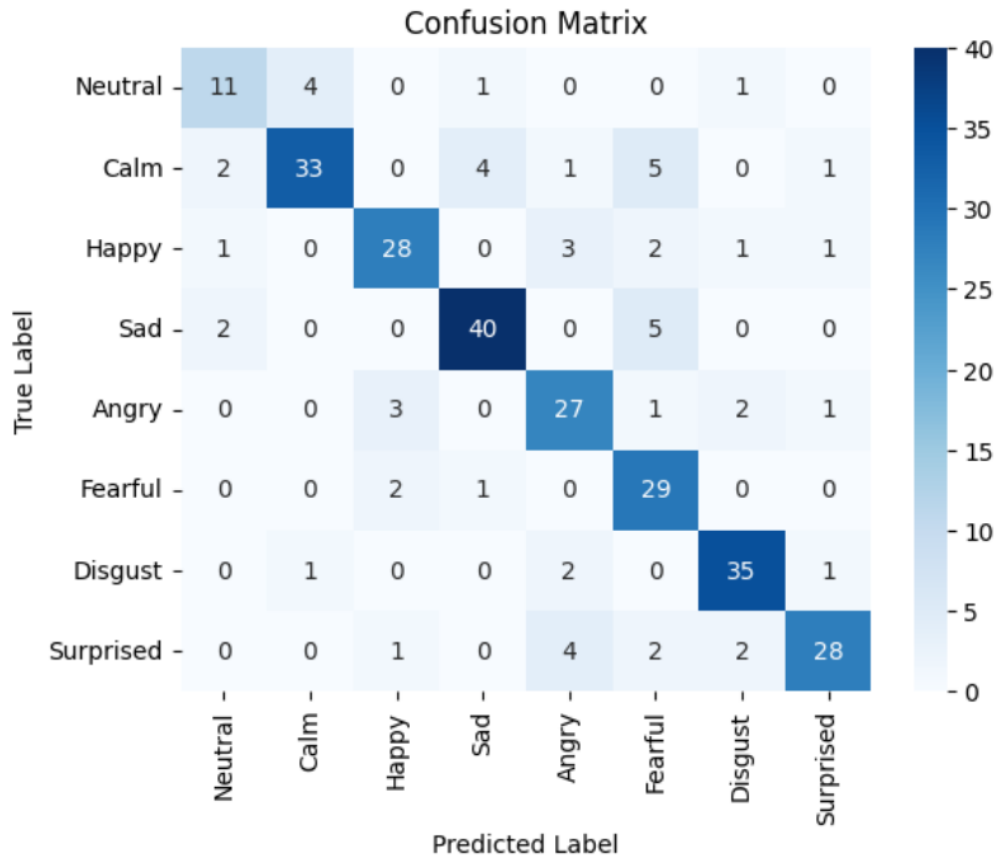


FIGURE 5 : RNN_LSTM Confusion Matrix

2.3 Support Vector Machines (SVM)

SVM is a supervised machine learning algorithm that is commonly used for classification and regression tasks. It is particularly effective in solving binary classification problems, but can also be extended to handle multi-class classification.

The main idea behind SVM is to find an optimal hyperplane that separates the data points of different classes in a high-dimensional feature space. The hyperplane is chosen in such a way that it maximizes the margin, which is the distance between the hyperplane and the nearest data points of each class. The points closest to the hyperplane are called support vectors, hence the name of the algorithm.

In the case of linearly separable data, an SVM can find a hyperplane that perfectly separates the classes. However, in many real-world scenarios, the data is not linearly separable. In such cases, SVM uses a technique called the kernel trick to map the data into a higher-dimensional space where it can be linearly separated. This allows SVM to handle non-linear decision boundaries.

The implementation of an SVM classifier using the Scikit-learn library.

1. Importing the necessary libraries: Before using the SVM classifier, you need to import the required libraries.
2. Initializing the SVM classifier: The code initializes an SVM classifier object using the SVC class from Scikit-learn. The SVC class represents the Support Vector Classifier, which is an implementation of the SVM algorithm for classification tasks.
 - The `kernel='linear'` argument specifies that a linear kernel should be used. This means that the SVM will search for a linear decision boundary to separate the classes.
 - The `probability=True` argument allows the classifier to estimate class probabilities. This is useful if you want to calculate the confidence level or probability of a particular prediction.
 - The `random_state=9` argument sets the random seed to ensure reproducibility of the results. This means that each time you run the code with the same random seed, you should obtain the same results.
3. Fitting the SVM classifier: The code fits the SVM classifier to the training data using the `fit()` method. This trains the classifier by finding the optimal hyperplane that separates the classes based on the provided training data.
 - `x_train` refers to the feature matrix or input data that is used to train the SVM classifier.
 - `y_train` refers to the target variable or the corresponding class labels for the training data.

Once the SVM classifier is fitted, it can be used to make predictions on new, unseen data using the `predict()` method. Additionally, the `probability=True` setting allows you to obtain class probabilities using the `predict_proba()` method, if needed.

1. Plotting model performance: The code uses the `matplotlib` library to create a plot showing the performance of the SVM model.

- The `plt.plot()` function is used to plot the accuracy values on the y-axis for both training and testing data, represented by the markers 'o'.
- The green dashed line represents the training accuracy, while the red dashed line represents the testing accuracy.
- The x-axis labels are set as 'Training Accuracy' and 'Testing Accuracy'.
- The y-axis is labeled as 'Score'.
- The title of the plot is set as 'SVM Model Performance'.
- The `plt.ylim()` function sets the range of the y-axis to be between 0 and 1.0 (the maximum value of accuracy).
- The `plt.legend()` function displays the legend, which explains the meaning of the different lines in the plot.
- Finally, the `plt.show()` function is called to display the plot.

The resulting plot visualizes the training and testing accuracy of the SVM model, providing insights into how well the model performs on both the training and testing datasets.



```
[ ] # Initialize and fit SVM classifier
model_SVM = SVC(kernel='linear', probability=True, random_state=9)
model_SVM.fit(x_train, y_train)

[ ] # Predict for the test set using SVM
y_pred = model_SVM.predict(x_test)

[ ] # Calculate accuracy for SVM
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy (SVM): {:.2f}%".format(accuracy * 100))

Accuracy (SVM): 65.58%
```

FIGURE 6 : SVM Accuracy

The Training and Testing Accuracy is constant throughout the evaluation of the model which is at 0.8 and 0.6 respectively as seen in the graph :

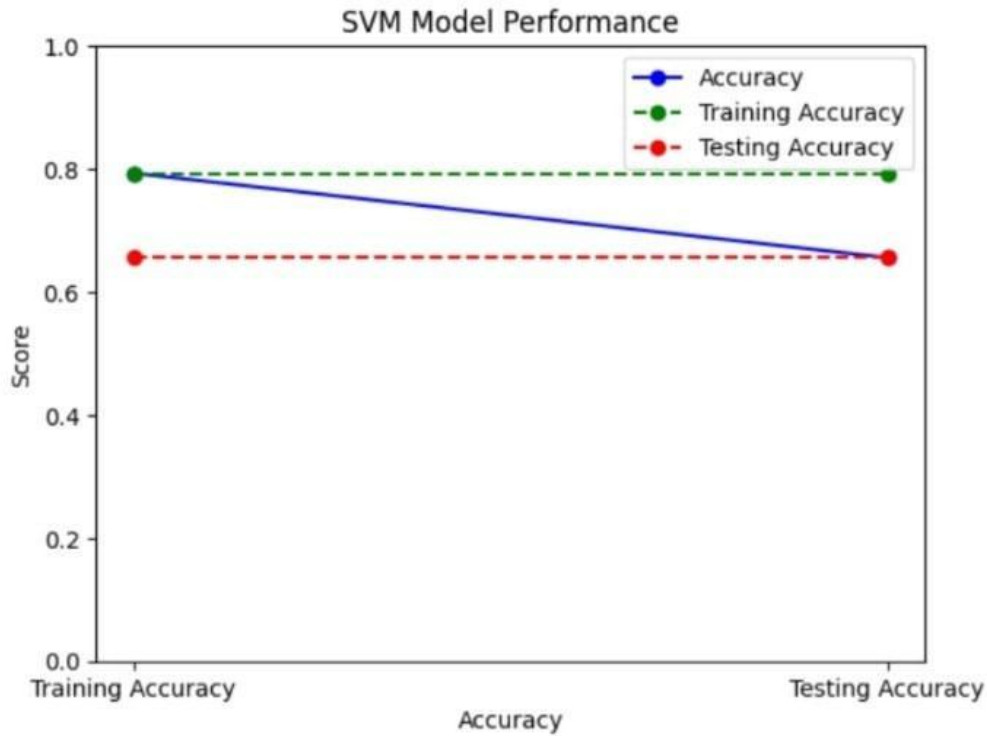


FIGURE 7 : SVM Model Performance Plot

2.4 Random Forest (RF)

Random Forest is a popular supervised machine learning algorithm used for both classification and regression tasks. It is an ensemble learning method that combines multiple decision trees to make predictions.

Here are the key points about Random Forest:

1. Ensemble of Decision Trees: Random Forest consists of a collection of decision trees.

Each tree is built using a random subset of the training data and a random subset of the features. This randomness helps to create diversity among the trees, reducing the risk of overfitting and improving the overall predictive power.

2. Bagging: Random Forest uses a technique called bagging (bootstrap aggregating).

Bagging involves creating multiple subsets of the training data by randomly sampling with replacement. Each subset is then used to train a separate decision tree. By averaging the predictions of these individual trees, Random Forest produces a final prediction that is more robust and accurate than that of a single decision tree.

3. Feature Randomness: In addition to sampling the training data, Random Forest also randomly selects a subset of features at each split of a tree. This helps to introduce more randomness and reduces the likelihood of a single feature dominating the tree's decision-making process. It also allows the model to handle high-dimensional datasets effectively.

4. Voting or Averaging: For classification tasks, Random Forest uses majority voting, where each tree in the ensemble votes for a class label, and the class with the most votes becomes the predicted class. For regression tasks, Random Forest takes the average of the predictions from all the trees as the final prediction.

Random Forest Implementation:

Initializing the Random Forest classifier: The code initializes a Random Forest classifier object using the `RandomForestClassifier` class from Scikit-learn. The `RandomForestClassifier` class represents the Random Forest algorithm for classification tasks.

- The `n_estimators=10` argument specifies the number of decision trees to be used in the Random Forest. In this case, the Random Forest will be composed of 10 decision trees.
 - The `random_state=9` argument sets the random seed to ensure reproducibility of the results. This means that each time you run the code with the same random seed, you should obtain the same results.
2. Calculating accuracy: Similar to the previous code snippet, the `score()` method of the Random Forest classifier is used to calculate the accuracy of the model on the training and testing data. The accuracy values are assigned to the variables `train_accuracy` and `test_accuracy`.
 3. Plotting model performance: The code uses the `matplotlib` library to create a plot showing the performance of the Random Forest model.
 - The `plt.plot()` function is used to plot the accuracy values on the y-axis for both training and testing data, represented by the markers 'o'. The x-axis labels are set as 'Training Accuracy' and 'Testing Accuracy'.
 - The green dashed line represents the training accuracy, while the red dashed line represents the testing accuracy.
 - The x-axis is labeled as 'Accuracy', and the y-axis is labeled as 'Score'.
 - The title of the plot is set as 'Random Forest Model Performance'.

- The `plt.ylim()` function sets the range of the y-axis to be between 0 and 1.0 (the maximum value of accuracy).
- The `plt.legend()` function displays the legend, which explains the meaning of the different lines in the plot.
- Finally, the `plt.show()` function is called to display the plot.

The resulting plot visualizes the training and testing accuracy of the Random Forest model, providing insights into how well the model performs on both the training and testing datasets.



```
[ ] # Initialize and fit Random Forest classifier
model_RF = RandomForestClassifier(n_estimators=10, random_state=9)
model_RF.fit(x_train, y_train)

RandomForestClassifier
RandomForestClassifier(n_estimators=10, random_state=9)

[ ] # Predict for the test set using Random Forest
y_pred = model_RF.predict(x_test)

[ ] # Calculate accuracy for Random Forest
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy (Random Forest): {:.2f}%".format(accuracy * 100))

Accuracy (Random Forest): 64.94%
```

FIGURE 8 : RF Accuracy

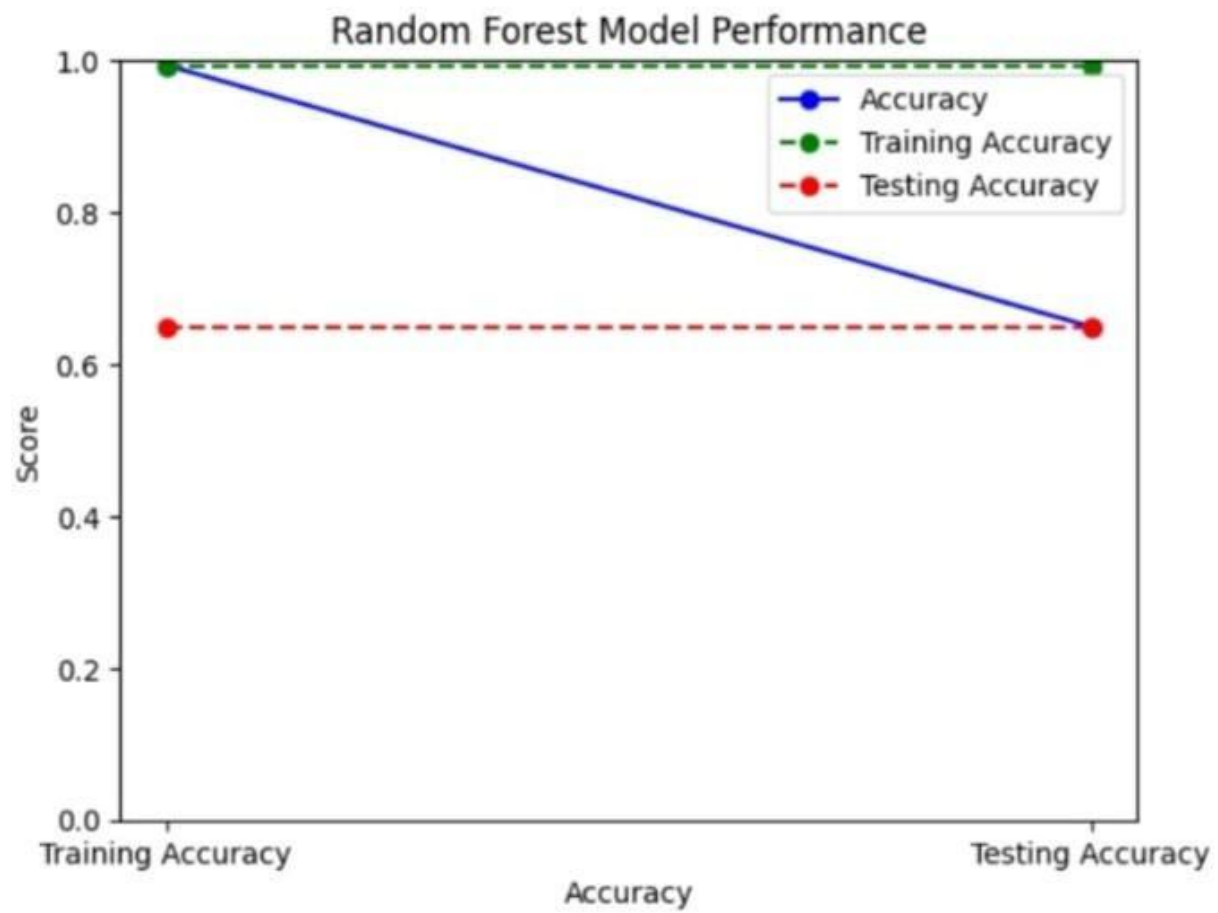


FIGURE 9 : RF Model Performance Plot

This is the Random Forest model plot showing training and testing accuracy

2.5 K-Nearest Neighbors (KNN)

KNN is a supervised machine learning algorithm used for both classification and regression tasks. It is a non-parametric algorithm that makes predictions based on the similarity of data points.

Here's how KNN works:

1. **Training Phase:** During the training phase, KNN memorizes the training data, which consists of feature vectors and corresponding labels.
2. **Prediction Phase:** When making predictions on new, unseen data, KNN finds the k nearest neighbors of the input data point in the feature space. The value of k is a hyperparameter that needs to be specified. The neighbors are determined based on a distance metric, commonly the Euclidean distance.
3. **Classification:** For classification tasks, KNN assigns a class label to the input data point based on the majority class among its k nearest neighbors. Each neighbor's vote has an equal weight.
4. **Regression:** For regression tasks, KNN predicts the output value for the input data point based on the average or median of the output values of its k nearest neighbors.

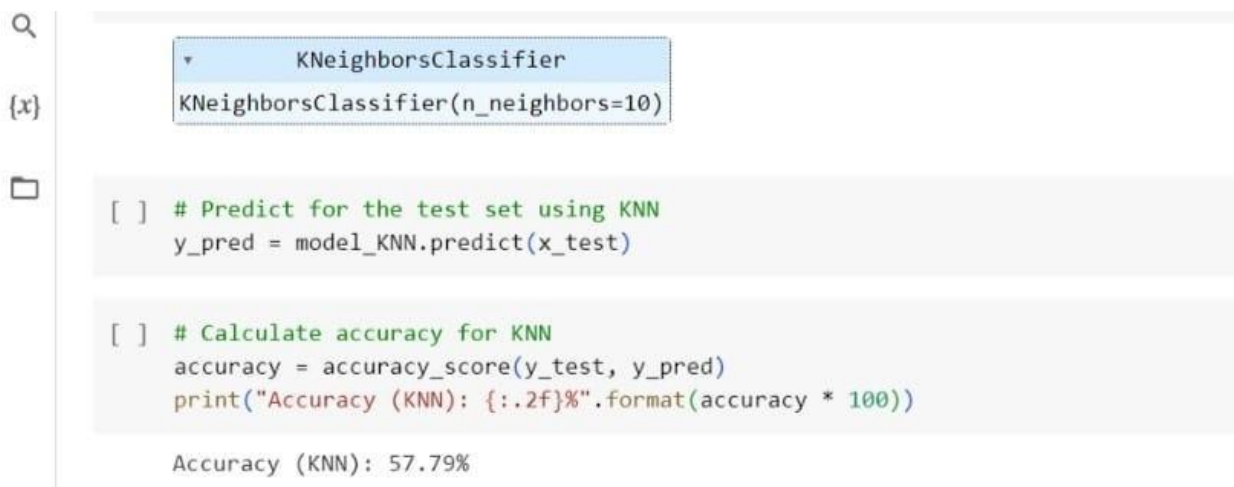
KNN Implementation:

1. **Initializing the KNN classifier:** The code initializes a KNN classifier object using the `KNeighborsClassifier` class from Scikit-learn. The `KNeighborsClassifier` class represents the KNN algorithm for classification tasks.
 - The `n_neighbors=10` argument specifies the number of neighbors to consider in the prediction. In this case, the code sets it to 10, meaning that the classifier will consider the 10 nearest neighbors of an input data point.

2. Fitting the KNN classifier: The code fits the KNN classifier to the training data using the `fit()` method. This trains the classifier by memorizing the feature vectors and labels from the provided training data.

- `x_train` refers to the feature matrix or input data that is used to train the KNN classifier.
- `y_train` refers to the target variable or the corresponding class labels for the training data.

Once the KNN classifier is fitted, it can be used to make predictions on new, unseen data using the `predict()` method. The `predict()` method takes a feature matrix as input and returns the predicted class labels.



The screenshot shows a Jupyter Notebook interface. On the left, there is a sidebar with a search icon, a variable `{x}`, and a folder icon. The main area displays a code cell with the following content:

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)

[ ] # Predict for the test set using KNN
    y_pred = model_KNN.predict(x_test)

[ ] # Calculate accuracy for KNN
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy (KNN): {:.2f}%".format(accuracy * 100))
```

Below the code cell, the output is displayed:

```
Accuracy (KNN): 57.79%
```

FIGURE 10 : KNN Accuracy

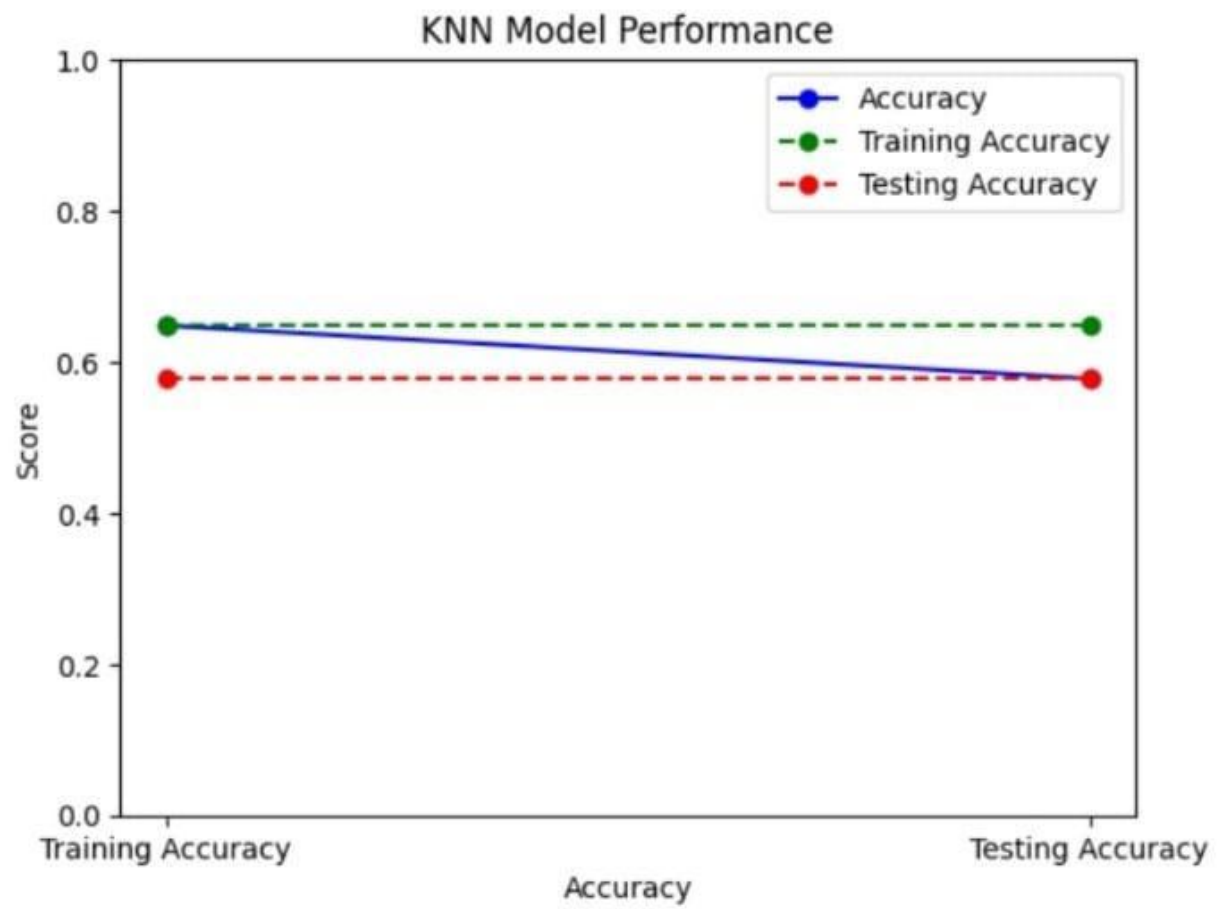


FIGURE 11 : KNN Model Performance Plot

2.6 Multi Layer Perceptron (MLP)

A multi-layer perception is a network of neurons connected with each other. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons. A multi-layer perceptron has one input layer and for each input a neuron is associated with it, it has one output layer with a single node for each output. It can have any number of layers hidden, activation, error and each layer can have more multiple neurons associated with it.

For this model training and testing parameters are as follows:

1. **alpha = 0.01** : In model this parameter controls the regularization term and it is using l2 regularization. It helps prevent overfitting by adding a penalty term to the loss function.
2. **batch_size = 256** : In this model we have used a batch gradient descent algorithm for optimization. This parameter determines the number of samples that will be propagated through the network at a time. In this case, the model will update its weights and biases after processing each batch of 256 samples.
3. **epsilon = 1e-08**: This is a small value added to the denominator to avoid division by zero errors when performing numerical computations.
4. **hidden_layer_sizes = (300,)**: This parameter defines the number of neurons in each hidden layer. In our model we have used 300 neurons in each hidden layer.
5. **learning_rate = 'adaptive'**: The learning rate determines the step size at each iteration while updating the weights and biases of the neural network. Setting it to 'adaptive' means that the learning rate will be adjusted dynamically based on the progress of the training process.
6. **max_iter = 2000**: This parameter sets the maximum number of iterations or epochs for which the model will be trained. An epoch refers to a complete pass through the entire training dataset. We have set it to max 2000 epochs.

```
[ ] #Initialise Multi Layer Perceptron Classifier
model = MLPClassifier(alpha = 0.01, batch_size = 256, epsilon = 1e-08, hidden_layer_sizes = (300,), learning_rate = 'adaptive', max_iter = 2000)

[ ] model.fit(x_train, y_train)

MLPClassifier
MLPClassifier(alpha=0.01, batch_size=256, hidden_layer_sizes=(300,),
learning_rate='adaptive', max_iter=2000)

[x] [ ] #Calculate Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy*100))

Accuracy: 68.83%

[ ]
```

FIGURE 12 : MLP Accuracy

2.7 Ensemble Model

An **ensemble model** for **Speech Emotion Recognition (SER)** using a voting classifier with a soft voting scheme is an effective approach to improve the accuracy and robustness of emotion classification. The ensemble model consists of multiple individual classifiers, each trained on different features or algorithms, and their predictions are combined through voting. In soft voting, the predicted probabilities for each class are averaged across all classifiers, and the class with the highest average probability is selected as the final prediction. This approach leverages the diversity of the individual classifiers and combines their strengths to achieve better performance. By aggregating the predictions from multiple classifiers, the ensemble model can capture a wider range of features and capture more nuanced patterns in speech signals, leading to more accurate emotion recognition results. Additionally, the ensemble model is more resistant to noise and variations in the input data, making it more robust in real-world scenarios. Overall, the ensemble model with a soft voting classifier is a powerful technique for improving SER performance and enhancing the reliability of emotion classification in speech analysis applications.

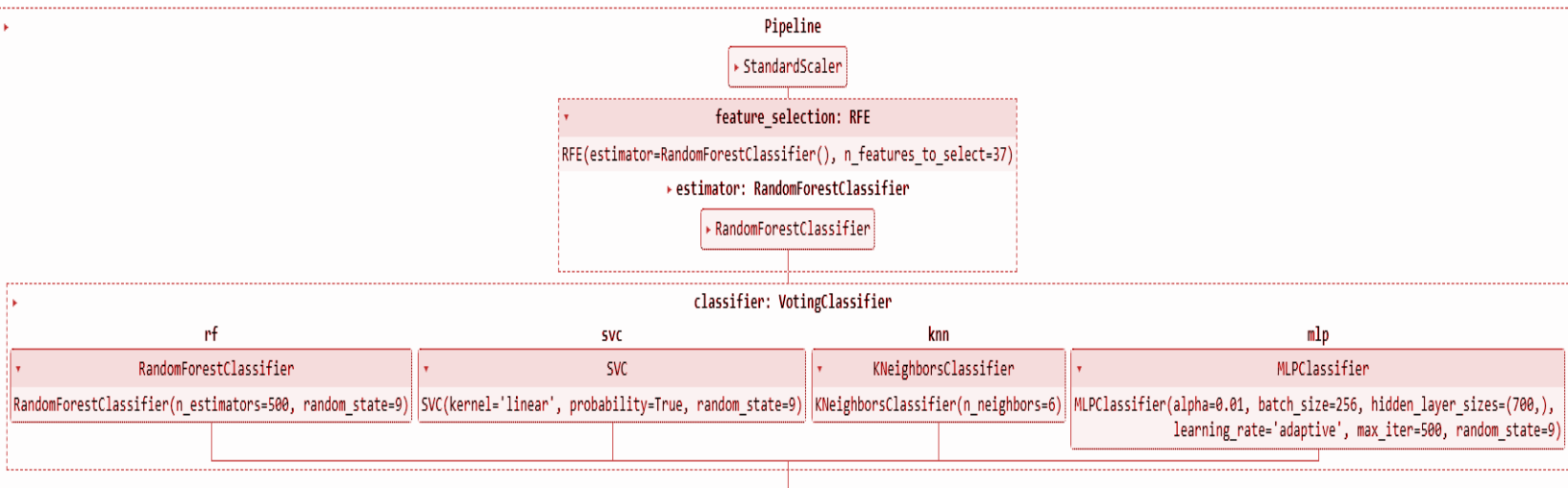


FIGURE 13 : Pipeline

The above model represents a pipeline for **Speech Emotion Recognition (SER)** using an ensemble model with a voting classifier and various classifiers within it. Let's break down the code and elaborate on each component and their parameters.

The pipeline starts with a data preprocessing step, where the `'StandardScaler()'` is used to standardize the features. Standardization is crucial in SER to ensure that different features are on the same scale, allowing classifiers to learn effectively from the data.

Next, the pipeline includes feature selection using Recursive Feature Elimination (RFE) with a Random Forest Classifier (`'RandomForestClassifier()'`). RFE helps in selecting the most relevant features for emotion recognition by recursively eliminating less important features. The `'n_features_to_select'` parameter is set to 37, indicating that the algorithm should select the top 37 features.

Moving on to the ensemble classifier, the `'VotingClassifier'` is utilized with the `'voting'` parameter set to 'soft'. This indicates that the ensemble will perform a soft voting scheme, where the predicted probabilities for each class from each individual classifier will be averaged.

The ensemble model consists of four different classifiers:

1. Random Forest Classifier (`RandomForestClassifier()`): This classifier utilizes an ensemble of decision trees and performs classification by majority voting. The `n_estimators` parameter is set to 500, specifying the number of trees in the ensemble.
2. Support Vector Classifier (`SVC(kernel='linear', probability=True, random_state=9)`): This classifier employs support vector machines (SVM) with a linear kernel for classification. The `probability` parameter is set to True, allowing the classifier to output probability estimates. The `random_state` parameter is set to 9 to ensure reproducibility.
3. K-Nearest Neighbors Classifier (`KNeighborsClassifier(n_neighbors=6)`): This classifier classifies instances based on their nearest neighbors. The `n_neighbors` parameter is set to 6, indicating that it considers the six nearest neighbors for classification.
4. Multi-Layer Perceptron Classifier (`MLPClassifier()`): This classifier represents a neural network with multiple hidden layers. It utilizes the backpropagation algorithm for training. The `alpha`, `batch_size`, `hidden_layer_sizes`, `max_iter`, and `learning_rate` parameters are set to specific values to control the learning process and network architecture. The `random_state` parameter is set to 9 for reproducibility.

By combining these diverse classifiers in the ensemble model, the pipeline benefits from their collective decision-making power. Each classifier brings its own unique perspective and strengths in identifying emotions from speech signals, leading to a more accurate and robust SER system.

Overall, this pipeline provides a comprehensive framework for Speech Emotion Recognition, encompassing data preprocessing, feature selection, and an ensemble of classifiers. It leverages the strengths of multiple algorithms and incorporates a soft voting scheme to improve the accuracy and reliability of emotion classification in speech analysis applications.

This is what accuracy with output looks like:

```

Test file: YAF_gas_angry -> Predicted Emotion: angry
Test file: YAF_gaze_angry -> Predicted Emotion: angry
Test file: YAF_hire_angry -> Predicted Emotion: angry
Test file: YAF_keen_angry -> Predicted Emotion: angry
Test file: YAF_lease_angry -> Predicted Emotion: angry
Test file: YAF_kill_angry -> Predicted Emotion: angry
Test file: YAF_jug_angry -> Predicted Emotion: angry
Test file: YAF_mode_angry -> Predicted Emotion: angry
Test file: YAF_merge_angry -> Predicted Emotion: angry
Test file: YAF_perch_angry -> Predicted Emotion: angry
Test file: YAF_pain_angry -> Predicted Emotion: fearful
Test file: YAF_mill_angry -> Predicted Emotion: angry
Test file: YAF_neat_angry -> Predicted Emotion: angry
Test file: YAF_near_angry -> Predicted Emotion: angry
Test file: YAF_rat_angry -> Predicted Emotion: angry
Test file: YAF_raid_angry -> Predicted Emotion: angry
Test file: YAF_sheep_angry -> Predicted Emotion: angry
Test file: YAF_talk_angry -> Predicted Emotion: angry
Test file: YAF_witch_angry -> Predicted Emotion: angry
Test file: YAF_voice_angry -> Predicted Emotion: fearful
Test file: YAF_youth_angry -> Predicted Emotion: angry
Accuracy: 81.67%
Number of correct predictions per class:
Angry: 28
Fearful: 22
Surprised: 19
Sad: 29

```

FIGURE 14 : Ensemble Accuracy with output

This is the generated Confusion matrix:

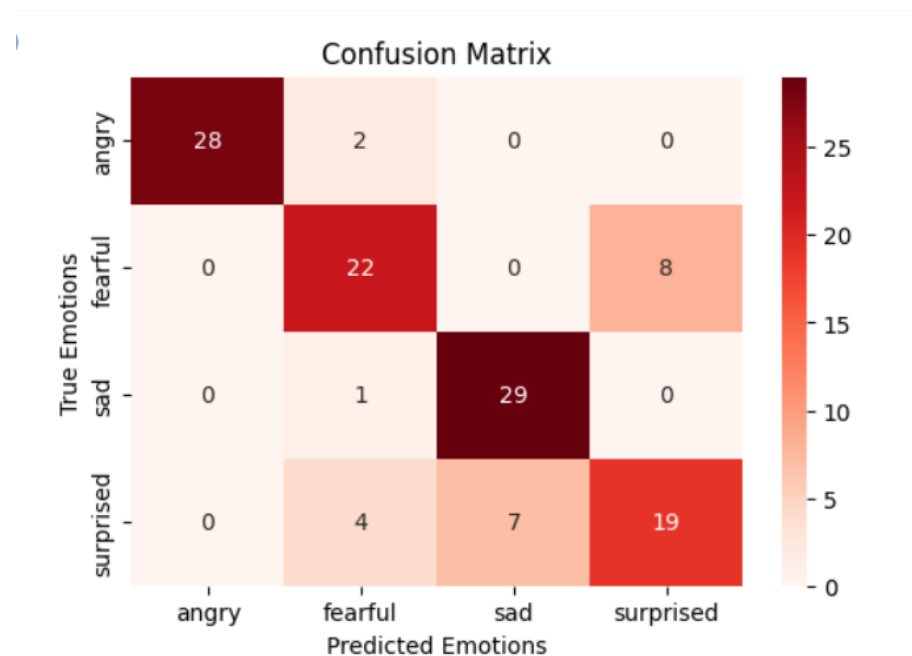


FIGURE 15 : Ensemble Model Confusion Matrix

CHAPTER 3 : RESULT AND DISCUSSION

3.1 RESULT

The whole integrated project can be clearly understood with the help of the flowchart given below. Here, we can observe the phases that take place step-by-step from data fetching to finally predicting and calculating accuracy of the model.

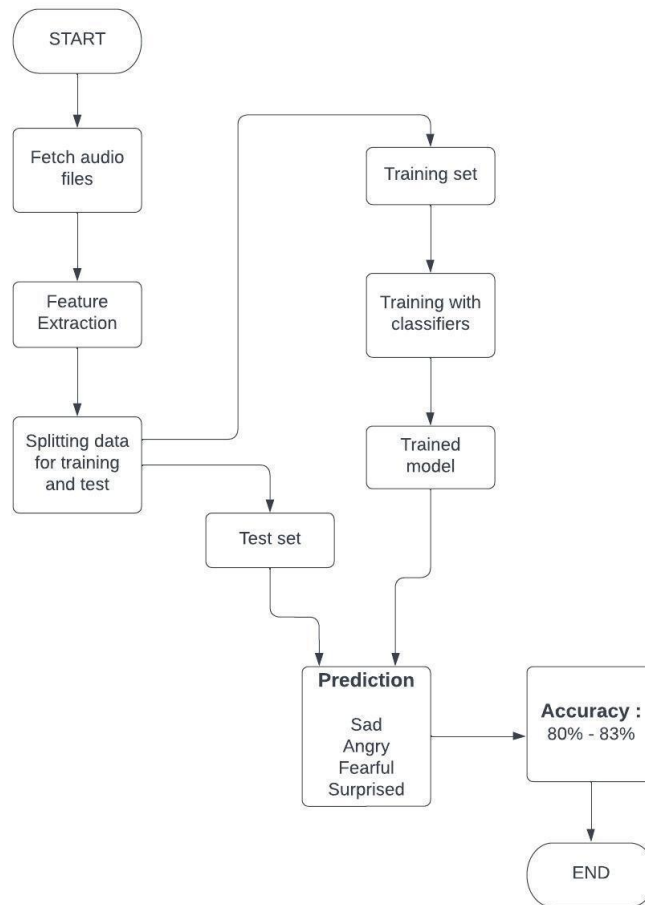
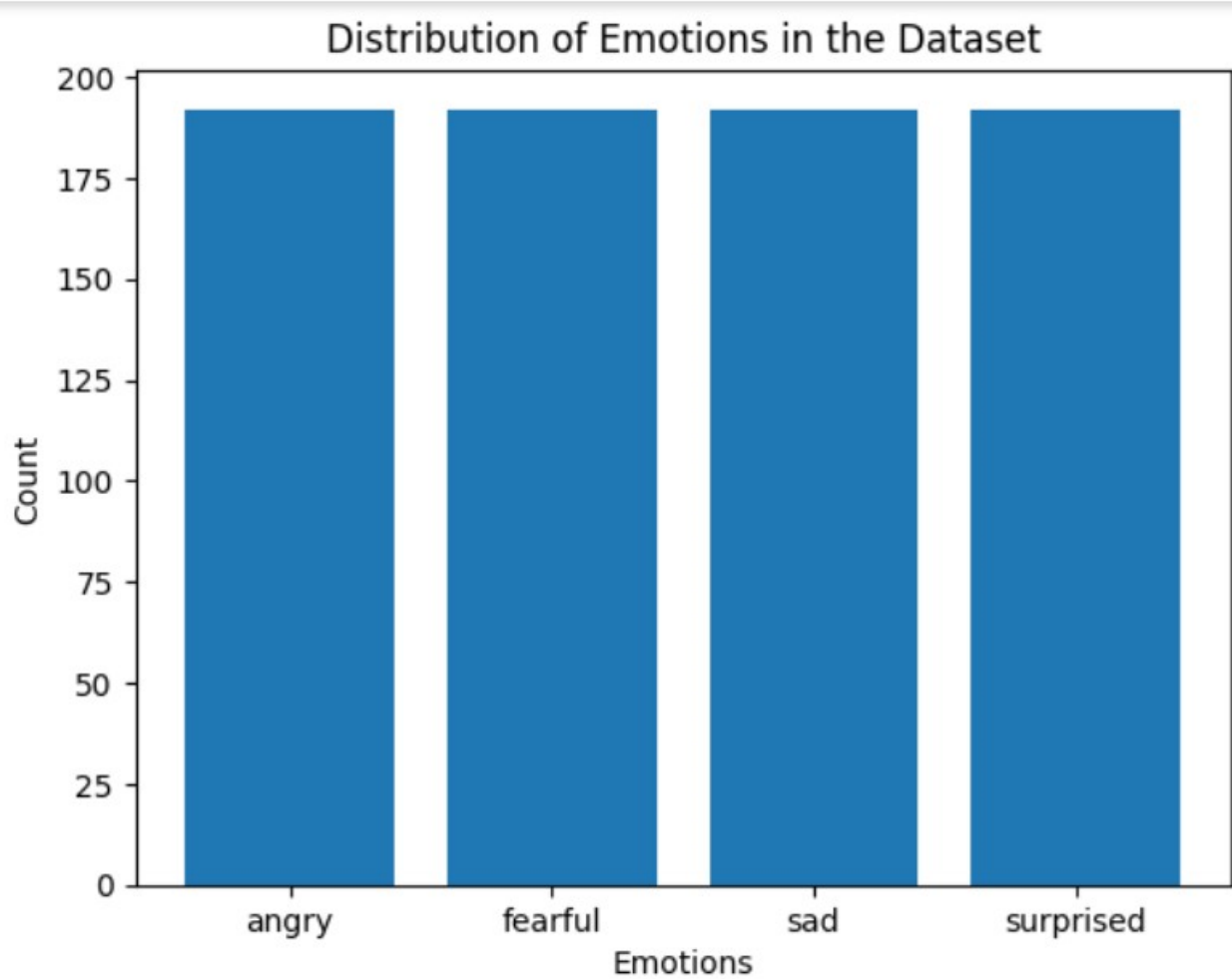


FIGURE 16 : Flowchart

We are properly able to recognize audio in four classes: sad, angry, fearful and surprised, with accuracy between 80% - 83%.

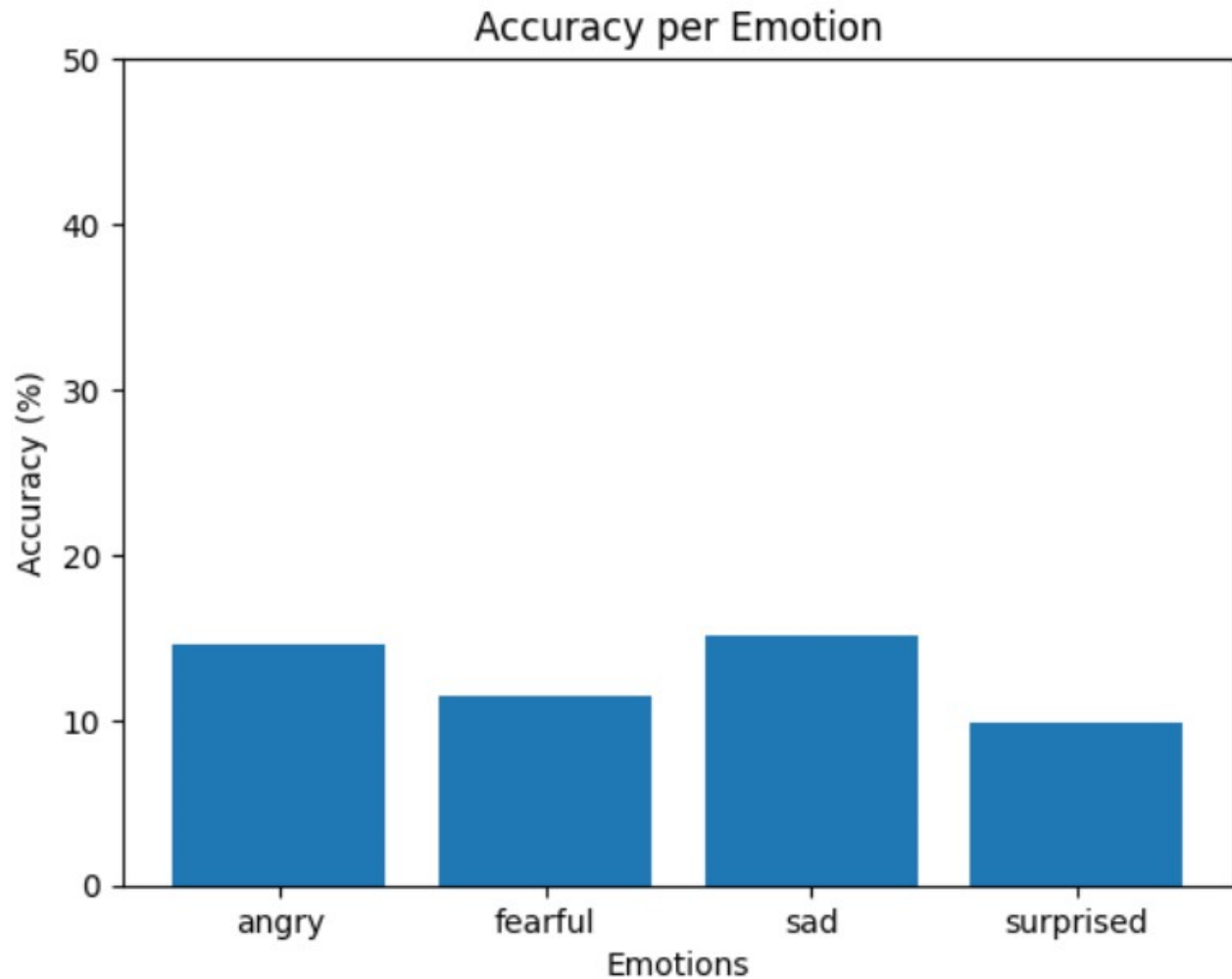
This is the initial distribution of emotions in the dataset :



These are some of the emotions that are given as input and the output emotion predicted associated with them.

```
Test file: OAF_bite_fear -> Predicted Emotion: fearful
Test file: OAF_came_fear -> Predicted Emotion: fearful
Test file: OAF_deep_fear -> Predicted Emotion: fearful
Test file: OAF_get_fear -> Predicted Emotion: fearful
Test file: OAF_gaze_fear -> Predicted Emotion: fearful
Test file: OAF_kite_fear -> Predicted Emotion: fearful
Test file: OAF_hit_fear -> Predicted Emotion: fearful
Test file: OAF_kill_fear -> Predicted Emotion: fearful
Test file: OAF_hire_fear -> Predicted Emotion: surprised
Test file: OAF_jug_fear -> Predicted Emotion: surprised
Test file: OAF_juice_fear -> Predicted Emotion: fearful
Test file: OAF_match_fear -> Predicted Emotion: fearful
Test file: OAF_live_fear -> Predicted Emotion: surprised
Test file: OAF_life_fear -> Predicted Emotion: surprised
Test file: OAF_merge_fear -> Predicted Emotion: surprised
Test file: OAF_lease_fear -> Predicted Emotion: fearful
Test file: OAF_mess_fear -> Predicted Emotion: fearful
Test file: OAF_numb_fear -> Predicted Emotion: surprised
Test file: OAF_nice_fear -> Predicted Emotion: fearful
Test file: OAF_pick_fear -> Predicted Emotion: fearful
Test file: OAF_neat_fear -> Predicted Emotion: fearful
Test file: OAF_said_fear -> Predicted Emotion: fearful
Test file: OAF_red_fear -> Predicted Emotion: fearful
Test file: OAF_read_fear -> Predicted Emotion: surprised
Test file: OAF_voice_fear -> Predicted Emotion: fearful
Test file: OAF_thought_fear -> Predicted Emotion: surprised
Test file: OAF_wash_fear -> Predicted Emotion: fearful
Test file: OAF_back_ps -> Predicted Emotion: surprised
Test file: OAF_bath_ps -> Predicted Emotion: surprised
```

At the end, accuracy calculated per emotion can be represented as :



3.2 ADVANTAGES

1. It allows us to extract and analyze emotional information from speech signals. Understanding emotions expressed in speech can provide valuable insights into human behavior, psychological state, and overall affective responses.
2. Speech is a non-intrusive modality for emotion recognition. Unlike other methods that require physical sensors or devices attached to individuals, speech can be captured remotely using microphones or recorded audio, making it a convenient and less obtrusive way to gather emotional data.
3. Speech is a natural and widely used communication medium among humans. By recognizing emotions from speech, we can gain insights into emotions expressed in various contexts, such as phone conversations, social media content, customer service interactions, and more.

3.3 FUTURE SCOPE

This project has the potential to enrich our understanding of human emotions, enhance human-computer interaction, improve mental health diagnostics, and provide valuable insights in various domains where emotions play a significant role.

Run time recognition of emotions can be added, this is particularly useful in applications where immediate feedback or response is required, such as emotion-aware virtual assistants, emotion-based human-computer interaction, or affective computing systems.

Multilingual and cross-cultural emotion recognition: Emotions can be expressed differently across languages and cultures. Future research may explore methods for cross-cultural and multilingual emotion recognition, enabling models to recognize and adapt to diverse emotional expressions.

REFERENCES

[1] Ong K, Lee C, Lim H, Lim K, “Speech Emotion Recognition with Light Gradient Boosting Decision Trees Machine,” *International Journal of Electrical and Computer Engineering* 13.4 (2023): 4020–4028.

URL: <https://www.mendeley.com/catalogue/a64d2cd2-c7b4-39be-b540-6db77acba76/>

[2] Matin, Rezwan, “Developing a speech emotion recognition solution using ensemble learning for children with autism spectrum disorder to help identify human emotions,” *Texas State University Electronic Theses and Dissertations*(2020-12)

URL: <https://digital.library.txstate.edu/handle/10877/13037>

[3] Meera Mohan, P. Dhanalakshmi, R. Satheesh Kumar, “Speech Emotion Classification using Ensemble Models with MFCC,” *Science Direct*(01-2023)

URL: <https://www.sciencedirect.com/science/article/pii/S1877050923001631>

[4] M.M. Venkata Chalapathi, M. Rudra Kumar, Neeraj Sharma, S. Shitharth, "Ensemble Learning by High-Dimensional Acoustic Features for Emotion Recognition from Speech Audio Signal," *Hindawi Security and Communication Networks*(2022).

URL: <https://www.hindawi.com/journals/scn/2022/8777026/>

[5] Sathit Prasomphan, Surinee Doungwichain, “Detecting Human Emotion via Speech Recognition by Using Ensemble Classification Model,” *Springer Link*(2018)

URL: https://link.springer.com/chapter/10.1007/978-3-319-98752-1_8

[6] Damian Valles, Rezwan Matin, "An Audio Processing Approach using Ensemble Learning for Speech-Emotion Recognition for Children with ASD," *IEEE Xplore*(2021)

URL: <https://ieeexplore.ieee.org/document/9454174>

[7] Sergey Verbitskiy, Vladimir Berikov, Viacheslav Vyshegorodtsev, “ERANNs: Efficient residual audio neural networks for audio pattern recognition” *Science Direct*(2022)

URL: <https://doi.org/10.1016%2Fj.patrec.2022.07.012>

[8] Nagaraja N Poojary, Dr. Shivakumar G S, Akshath Kumar B.H, “Speech Emotion Recognition Using MLP Classifier,” *International Journal of Scientific Research in Science and Technology*. 218-222. (2021)

URL: <https://ijsrcseit.com/paper/CSEIT217446.pdf>

[9] M. Zielonka, A. Piastowski, A. Czyżewski, P. Nadachowski, M. Operlejn, K. Kaczor, "Recognition of Emotions in Speech Using Convolutional Neural Networks on Different Datasets," Multidisciplinary Digital Publishing Institute(2022)

URL: <https://www.mdpi.com/2079-9292/11/22/3831>

[10] A. Yadav and D. K. Vishwakarma, "A Multilingual Framework of CNN and Bi-LSTM for Emotion Classification," *IEEE Xplore*(2020).

URL: <https://ieeexplore.ieee.org/document/9225614>

[11] Nasifa Tanjin Ira, Mohammad Osiur Rahman, "An Efficient Speech Emotion Recognition Using Ensemble Method of Supervised Classifiers," *IEEE Xplore*(2020)

URL: <https://ieeexplore.ieee.org/document/9350913>

[12] Zhouyu Fu, Guojun Lu, K. M. Ting, D. Zhang, "A Survey of Audio-Based Music Classification and Annotation," *IEEE Transactions on Multimedia*, vol. 13, no. 2, pp. 303-319, (April 2011).

URL: <https://ieeexplore.ieee.org/document/5664796>

[13] Ashish B. Ingale, D. S. Chaudhari, "Speech Emotion Recognition," International Journal of Soft Computing and Engineering (IJSCE) (March 2012)

URL: <https://www.ijscce.org/wp-content/uploads/papers/v2i1/A0425022112.pdf>

[14] Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) Dataset

URL: <https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio>

[15] Feature Extraction

URL: <https://devopedia.org/audio-feature-extraction#:~:text=Audio%20feature%20extraction%20is%20a,converting%20digital%20and%20analog%20signals.>

[16] Machine Learning models

URL: <https://scikit-learn.org/stable/>

[17] Ensemble Model

URL: <https://www.geeksforgeeks.org/ensemble-classifier-data-mining/>

[18] Feature selection by Recursive Feature Elimination method

URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[19] Implementation of Pipeline

URL: <https://www.turing.com/kb/building-ml-pipeline-in-python-with-scikit-learn>

[20] Models performance Visualization

URL: <https://coderzcolumn.com/tutorials/machine-learning/scikit-plot-visualizing-machine-learning-algorithm-results-and-performance>

[21] Toronto emotional speech set (TESS) Dataset

URL: <https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess>