By Krishnaveni

PROJECT REPORT ON
Emotions Recognition from EEG Signals using Neural networks

Abstract:

This Project report gives a detailed insights of Recognizing the emotions from EEG signals using the neural networks. The use of emotion recognition (ER) based on electroencephalograms (EEG) in brain-computer interfaces (BCI) has grown a lot during the past ten years. BMIs can either be invasive or non-invasive. Invasive brain mapping includes surgically implanting electrodes under the scalp to more accurately transmit brain impulses. On the other hand, non-invasive BMIs do not require surgery and are positioned over the head to monitor brain activity. Several decoding techniques are available to achieve BMI. One such technique is Electroencephalograph (EEG) signals which are noninvasive. In this we are utilizing the EEG signals for Emotion recognition which involve pre-processing and feature extraction, followed by classification. Deep learning has recently been utilized to categorize emotions in BCI systems, with improved results when compared to traditional classification algorithms. The primary goal of this project is to categorize emotions from electroencephalogram data using different recurrent neural network designs and PCA techniques. This study employs different architectures for emotion identification utilizing EEG signals: DNN(deep neural network), LSTM (long short-term memory network), GRU (gated recurrent unit), PCA(Principal component analysis) with ensemble techniques and ANN(artificial Neural network). Experiment results validated the effectiveness of these networks in terms of performance indicators. The algorithms that best performed this task is DNN.

Introduction:

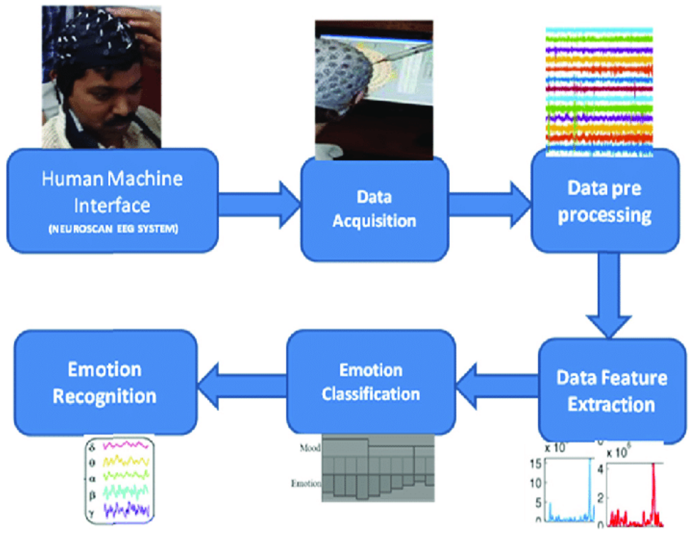


Figure 1: Block diagram for emotion recognition using EEG signals..

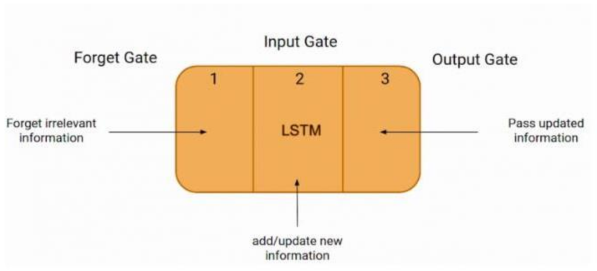
The Brain-Computer Interface (BCI) directly integrates human (or animal) brain activity with artificial signaling pathways, creating an interactive channel between the human brain and external equipment for a variety of applications. Such an interaction begins with the recording of brain activity followed by signal processing and analysis to determine the users' intentions. Emotion recognition is one of the most alluring research areas because of its potential uses in multiple circumstances. For this project we have used EEG brain wave dataset. EEG stands for electroencephalography, is a method to record the electrical activity of the brain using electrophysiological monitoring. This is done through non-invasive (in most cases) electrodes placed along the scalp that records the brain's spontaneous electrical activity over a period. Monitoring this activity is used for diagnosing or treating various disorders like brain tumors, stroke, sleep disorders, etc. Main purpose of this is to build a emotion recognition model with less rate of error and high classification rate. Various techniques were employed for this as follows.

Deep Learning Frameworks

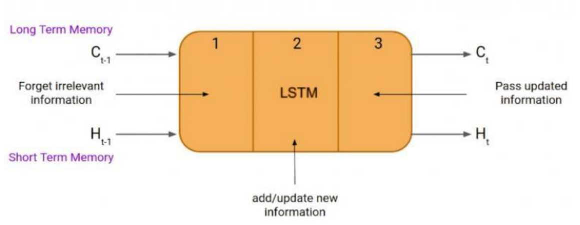
The fundamental ideas and important network components are presented. The novel model is also described in detail.

Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory Networks (LSTM) is a sophisticated RNN that can retain data for extended periods of time. It can deal with the vanishing gradient problem that RNNs confront. Depending on the data, the network may or may not preserve the memory. The network's long-term dependencies are maintained by its gating mechanisms. Based on the gating mechanism, memory may be released or held on demand in the network. The fundamental three components of an LSTM cell are gates. The forget gate is the first section, the input gate is the second, and the output gate is the third.



LSTM cell with gates

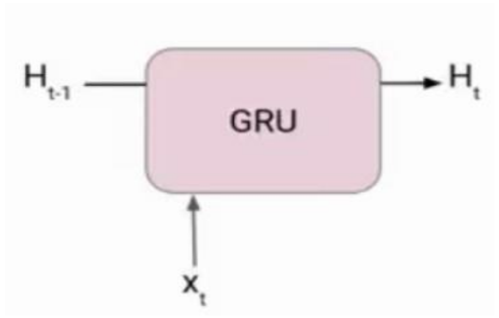


LSTM cell with hidden and cell states

Here, LSTM hidden state refers to the short-term memory and the cell state refers to the long memory. The input gate decides how much of the input node's value needs to be added to the internal state of the memory cell currently in use. The forget gate controls whether to maintain or flush the memory's current value. And the output gate controls whether the memory cell influences the output at the current time step.

Gated Recurrent Network (GRU)

Gated Recurrent Network are RNN design variations that use gating methods to regulate information flow between neural network cells. GRUs and LSTMs work on the same principle. They are quite fresh as compared to LSTM. Therefore, GRUs outperform LSTMs and have a more straightforward architecture. In GRUs, there is just one concealed state that is carried over from one time step to the next. Because of the gating mechanisms and calculations that the hidden state and incoming data go through, it may sustain both long- and short-term dependence.



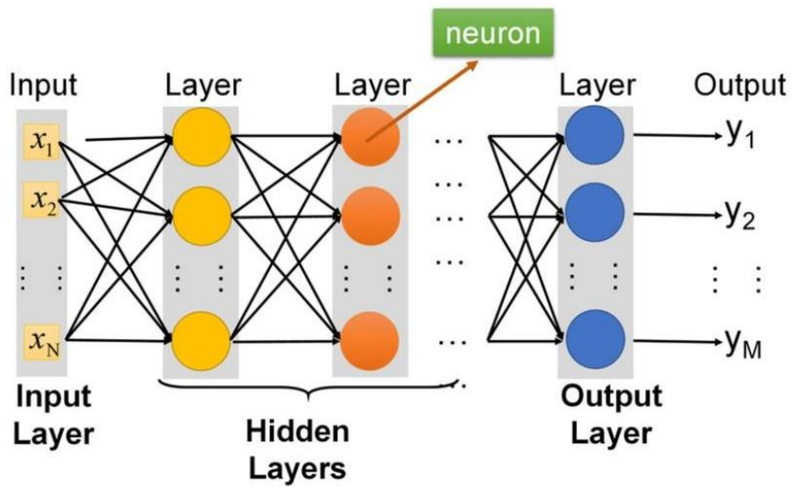
GRU cell

GRU uses include speech recognition, stock price prediction, machine translation, and sentiment analysis, among others. GRU addresses the classic RNN problem by combining two gates: the update gate and the reset gate. The update gate works similarly to the input and forget gate of an LSTM. It determines whether information should be destroyed or kept. The reset gate determines how much old data must be discarded; in other words, it determines whether or not the previous cell state is important.

By Krishnaveni

Deep Neural Network (DNN)

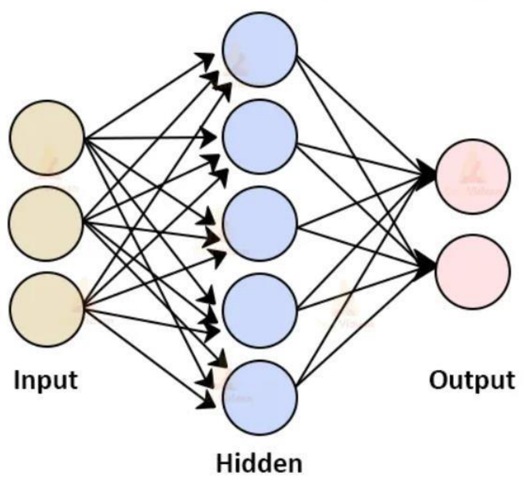
Deep Neural Networks are ANNs with more than one hidden layer in their architecture. With the use of mathematical modeling, these networks handle complicated data. Deep Neural Networks have an input layer, an output layer, and a few hidden layers in between. Based on the input received, the neurons transmit the signal to other neurons. If the signal value is larger than the threshold value, the output is passed; otherwise, it is ignored. As you can see, the data is transmitted to the input layer, which produces output for the next layer, and so on until it reaches the output layer, which predicts yes or no based on likelihood. A layer is made up of several neurons, and each neuron has a specific purpose.



DNN architecture

Artificial Neural Network (ANN)

A neural network has three layers. The first layer is the input layer. It contains the input neurons that send data to the hidden layer. The hidden layer computes on incoming data and sends the results to the output layer. Weight, activation function, and cost function are all included. The link between neurons is known as weight, which is a numerical number. The weight between neurons influences the neural network's learning ability. The weight between neurons fluctuates during artificial neural network learning. ANN learns from sample data sets is a significant advantage. The most typical application of ANN is for random function.



ANN architecture

Procedure:

The project starts with installing and importing all the Pandas, NumPy libraries and other required dependencies. The first step is to load the EEG Brainwave dataset. I have performed EDA (Exploratory Data Analysis) for better understanding of the available data and checked for existence of missing values. During data pre-processing, label encoding is employed to convert the labels into a numeric form to convert them into the machine-readable form. Here our target variable is Label which is categorized into three emotions like ‘Positive’, ‘Negative’ and ‘Neutral’. In Data preprocessing, feature engineering step involves feature scaling, which is one of the most important tasks. It brings all the features to a common scale. After data processing, data is splitted, performed Dataaugmentation and all the models are built. Now that the data has been preprocessed, it is ready for model construction. A preprocessed dataset, neural networks, and machine learning methods are necessary for model construction.

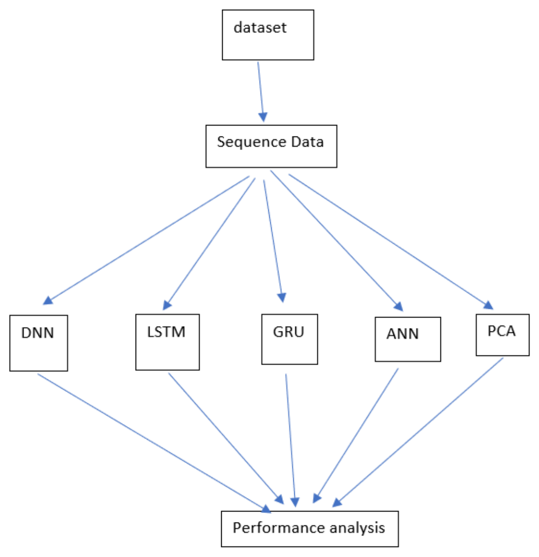
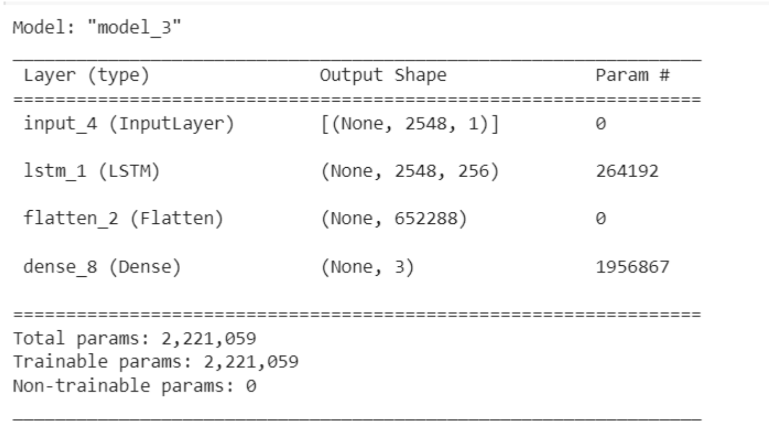


Fig 2: shows flow chart

The above is the flow chart that shows the methodology of the project. After developing different models, there performance analysis is compared. The comparison of the models yields the best model in terms of accuracy metrics.

Architecture implemented

1. LSTM Architecture:

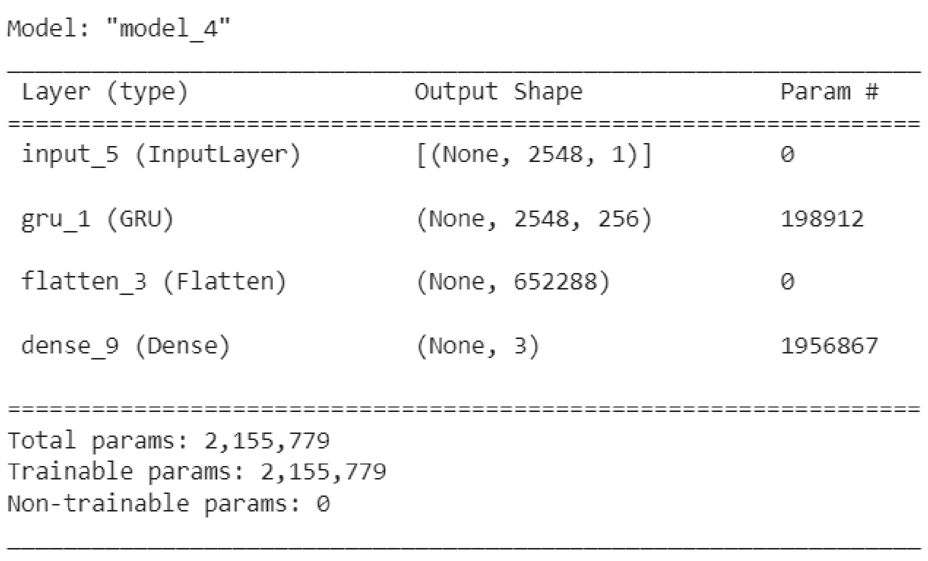


```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 2548, 1)]         0

 lstm_1 (LSTM)               (None, 2548, 256)         264192

 flatten_2 (Flatten)         (None, 652288)            0

 dense_8 (Dense)             (None, 3)                 1956867

=================================================================
Total params: 2,221,059
Trainable params: 2,221,059
Non-trainable params: 0
_________________________________________________________________
```

Kera's implementation of the LSTM model. The LSTM algorithm used in this work consists of an LSTM layer with 256 units, a flatten layer, and finally, a dense layer with softmax activation. Learning features are determined using the LSTM layers, and a dense layer is used to classify them into emotions from raw EEG signals.

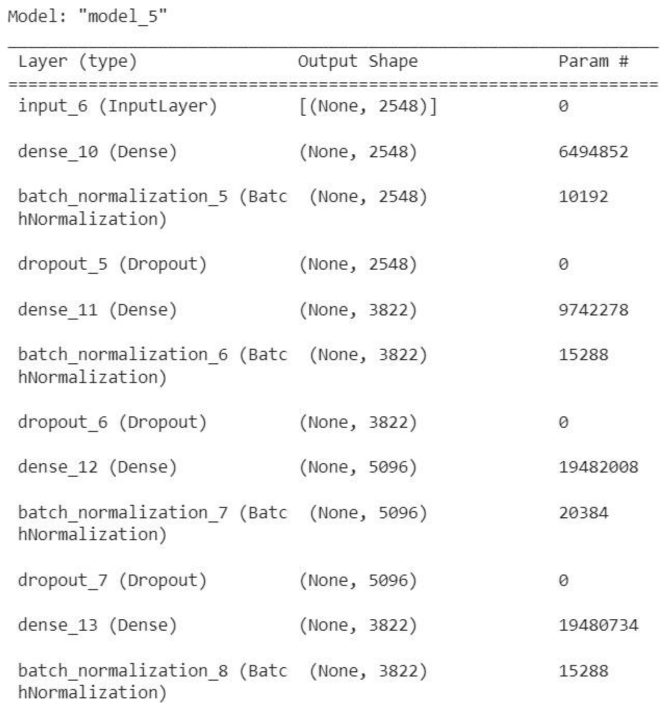
2. GRU Architecture

The GRU method utilized in this work is composed of a 256 unit GRU layer, a flatten layer, and a dense layer with softmax activation. The GRU layers are used to determine learning characteristics, while a dense layer is utilized to categorize EEG data into emotions. GRUs outperform LSTMs with less training data, particularly when used to language modeling applications. GRUs are easier to change and hence simpler with less code when more network inputs are required. The GRUs also have fewer parameters than the LSTM, as seen by the model summary. The LSTM model has 2,21,059 parameters with 256 internal units, whereas the GRU model has 2,155,779.



```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_5 (InputLayer)        [(None, 2548, 1)]         0

 gru_1 (GRU)                 (None, 2548, 256)         198912

 flatten_3 (Flatten)         (None, 652288)            0

 dense_9 (Dense)             (None, 3)                 1956867

=================================================================
Total params: 2,155,779
Trainable params: 2,155,779
Non-trainable params: 0
_________________________________________________________________
```

3. DNN Architecture



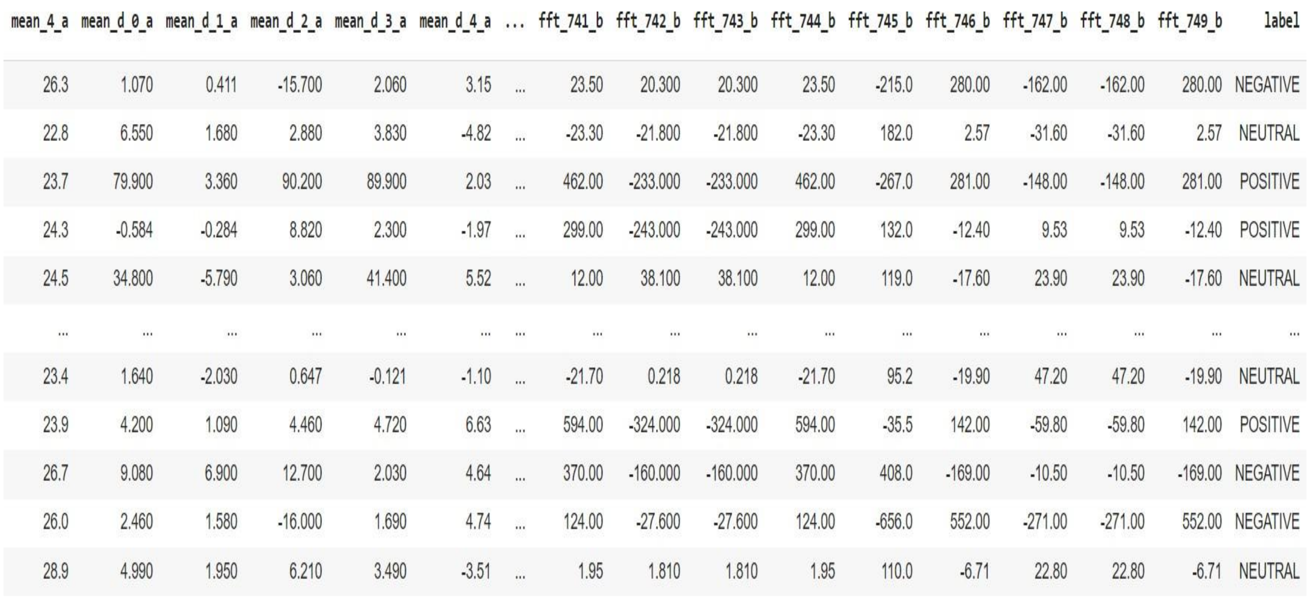
```
Model: "model_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_6 (InputLayer)        [(None, 2548)]            0

 dense_10 (Dense)            (None, 2548)              6494852

 batch_normalization_5 (Batc  (None, 2548)             10192
 hNormalization)

 dropout_5 (Dropout)         (None, 2548)              0

 dense_11 (Dense)            (None, 3822)              9742278

 batch_normalization_6 (Batc  (None, 3822)             15288
 hNormalization)

 dropout_6 (Dropout)         (None, 3822)              0

 dense_12 (Dense)            (None, 5096)              19482008

 batch_normalization_7 (Batc  (None, 5096)             20384
 hNormalization)

 dropout_7 (Dropout)         (None, 5096)              0

 dense_13 (Dense)            (None, 3822)              19480734

 batch_normalization_8 (Batc  (None, 3822)             15288
 hNormalization)
```

Data set description: The data was collected from two people (1 male, 1 female) for 3 minutes per state - positive, neutral, negative. They used a Muse EEG headband which recorded the TP9, AF7, AF8 and TP10 EEG placements via dry electrodes. Six minutes of resting neutral data is also recorded, the stimuli used to evoke the emotions are

1. Marley and Me - Negative (Twentieth Century Fox)-Death Scene
2. Up - Negative (Walt Disney Pictures)-Opening Death Scene
3. My Girl - Negative (Imagine Entertainment)-Funeral Scene
4. La La Land - Positive (Summit Entertainment)-Opening musical number
5. Slow Life - Positive (BioQuest Studios)-Nature timelapse
6. Funny Dogs - Positive (MashupZone)

To achieve better outcomes, data preprocessing is used to balance the data.

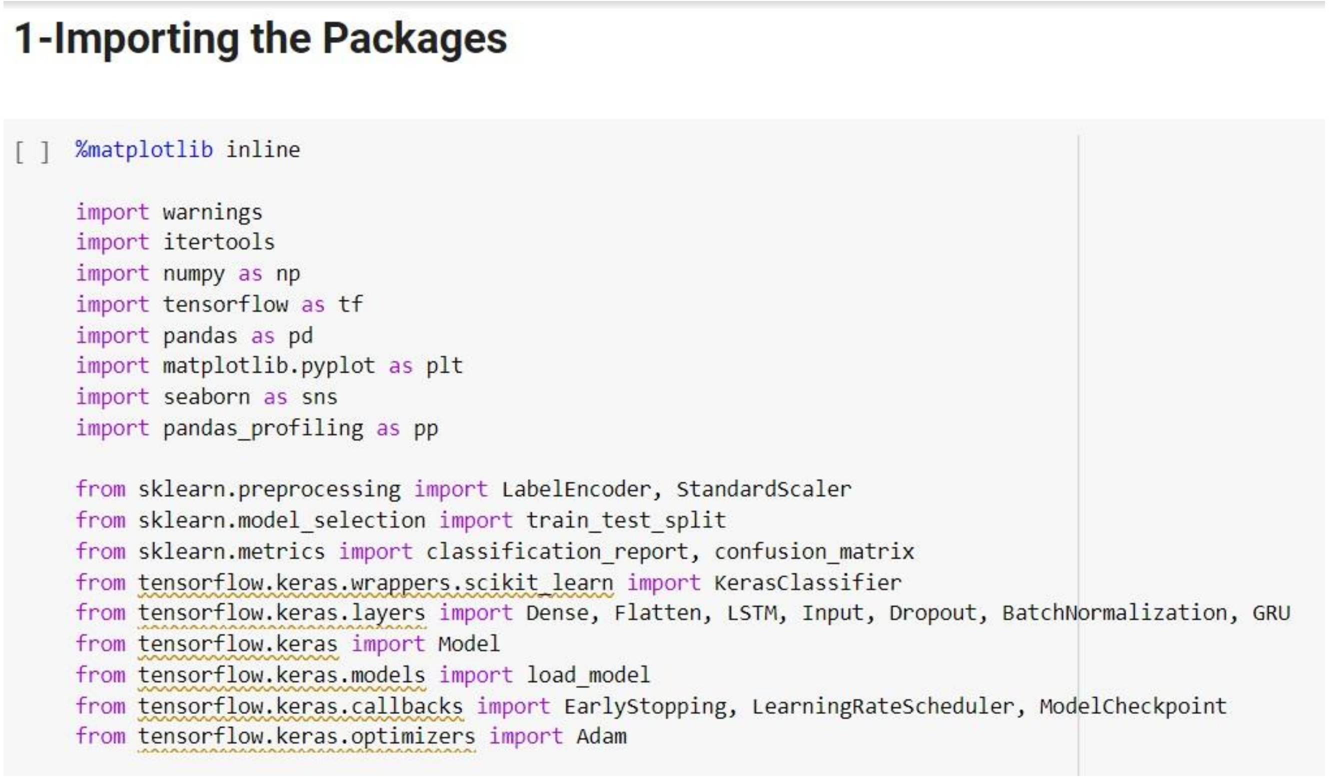
This database has a total of 2132 samples for three emotions: positive (708 samples), negative (708 samples), and neutral (716 samples).



mean_4_a	mean_d_0_a	mean_d_1_a	mean_d_2_a	mean_d_3_a	mean_d_4_a	...	fft_741_b	fft_742_b	fft_743_b	fft_744_b	fft_745_b	fft_746_b	fft_747_b	fft_748_b	fft_749_b	label
26.3	1.070	0.411	-15.700	2.060	3.15	...	23.50	20.300	20.300	23.50	-215.0	280.00	-162.00	-162.00	280.00	NEGATIVE
22.8	6.550	1.680	2.880	3.830	-4.82	...	-23.30	-21.800	-21.800	-23.30	182.0	2.57	-31.60	-31.60	2.57	NEUTRAL
23.7	79.900	3.360	90.200	89.900	2.03	...	462.00	-233.000	-233.000	462.00	-267.0	281.00	-148.00	-148.00	281.00	POSITIVE
24.3	-0.584	-0.284	8.820	2.300	-1.97	...	299.00	-243.000	-243.000	299.00	132.0	-12.40	9.53	9.53	-12.40	POSITIVE
24.5	34.800	-5.790	3.060	41.400	5.52	...	12.00	38.100	38.100	12.00	119.0	-17.60	23.90	23.90	-17.60	NEUTRAL
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
23.4	1.640	-2.030	0.647	-0.121	-1.10	...	-21.70	0.218	0.218	-21.70	95.2	-19.90	47.20	47.20	-19.90	NEUTRAL
23.9	4.200	1.090	4.460	4.720	6.63	...	594.00	-324.000	-324.000	594.00	-35.5	142.00	-59.80	-59.80	142.00	POSITIVE
26.7	9.080	6.900	12.700	2.030	4.64	...	370.00	-160.000	-160.000	370.00	408.0	-169.00	-10.50	-10.50	-169.00	NEGATIVE
26.0	2.460	1.580	-16.000	1.690	4.74	...	124.00	-27.600	-27.600	124.00	-656.0	552.00	-271.00	-271.00	552.00	NEGATIVE
28.9	4.990	1.950	6.210	3.490	-3.51	...	1.95	1.810	1.810	1.95	110.0	-6.71	22.80	22.80	-6.71	NEUTRAL

Fig 3: Dataset

Analysis:
Import all the required libraries like Numpy, Pandas, Keras and so on.

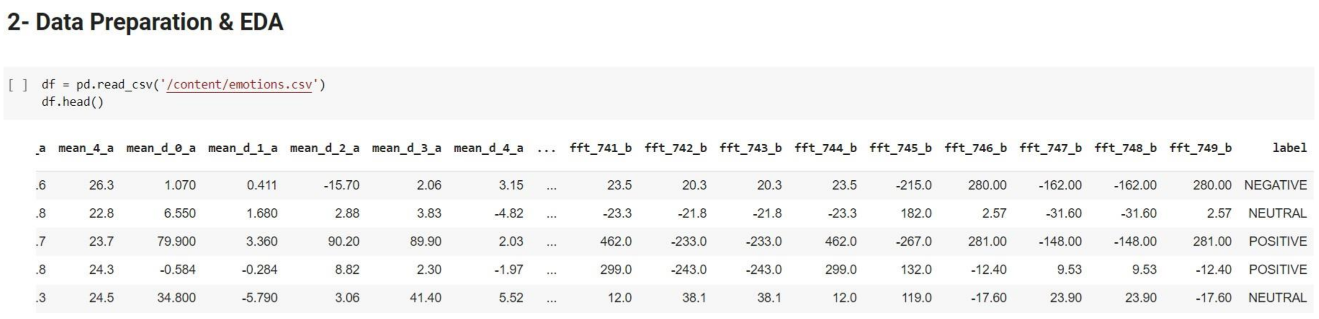


1-Importing the Packages

```
[ ] %matplotlib inline

    import warnings
    import itertools
    import numpy as np
    import tensorflow as tf
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    import pandas_profiling as pp

    from sklearn.preprocessing import LabelEncoder, StandardScaler
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report, confusion_matrix
    from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
    from tensorflow.keras.layers import Dense, Flatten, LSTM, Input, Dropout, BatchNormalization, GRU
    from tensorflow.keras import Model
    from tensorflow.keras.models import load_model
    from tensorflow.keras.callbacks import EarlyStopping, LearningRateScheduler, ModelCheckpoint
    from tensorflow.keras.optimizers import Adam
```

Fig 4: shows importing of required libraries

Loaded the EEG brainwave dataset



2- Data Preparation & EDA

```
[ ] df = pd.read_csv('/content/emotions.csv')
    df.head()
```

```
_a  mean_4_a  mean_d_0_a  mean_d_1_a  mean_d_2_a  mean_d_3_a  mean_d_4_a  ...  fft_741_b  fft_742_b  fft_743_b  fft_744_b  fft_745_b  fft_746_b  fft_747_b  fft_748_b  fft_749_b     label
.6      26.3       1.070       0.411      -15.70        2.06        3.15  ...       23.5       20.3       20.3       23.5     -215.0     280.00    -162.00    -162.00     280.00  NEGATIVE
.8      22.8       6.550       1.680        2.88        3.83       -4.82  ...      -23.3      -21.8      -21.8      -23.3      182.0       2.57     -31.60     -31.60       2.57   NEUTRAL
.7      23.7      79.900       3.360       90.20       89.90        2.03  ...      462.0     -233.0     -233.0      462.0     -267.0     281.00    -148.00    -148.00     281.00  POSITIVE
.8      24.3      -0.584      -0.284        8.82        2.30       -1.97  ...      299.0     -243.0     -243.0      299.0      132.0     -12.40       9.53       9.53     -12.40  POSITIVE
.3      24.5      34.800      -5.790        3.06       41.40        5.52  ...       12.0       38.1       38.1       12.0      119.0     -17.60      23.90      23.90     -17.60   NEUTRAL
```

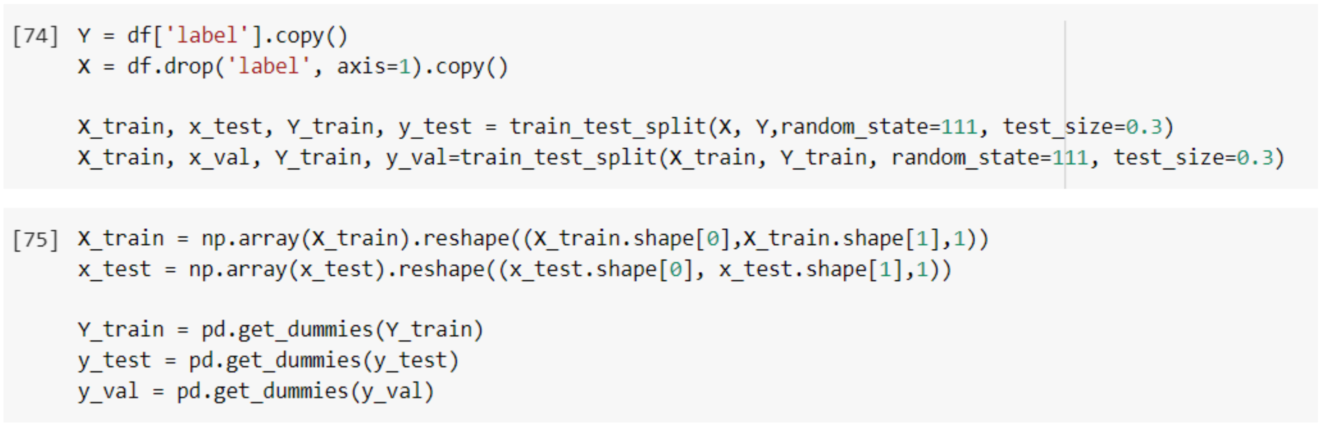
Fig 5: dataset is loaded

The dataset has no missing values, but our target variable label has categorical values which has to be converted to numeric values.

EDA (Exploratory Data Analysis): EDA is the process of studying a dataset in order to uncover patterns and anomalies (outliers) and to create hypotheses based on our understanding ofthe dataset. EDA include generating summary statistics for numerical data in a dataset as well as constructing various graphical representations to better comprehend the data.

The describe () function in Pandas is quite useful for obtaining different summary statistics. This function returns the data's count, mean, standard deviation, minimum and maximum values, and quantiles.

DATA SPLITTING: We have splitted the data as test set is 30% and train set is 70%



```
[74] Y = df['label'].copy()
     X = df.drop('label', axis=1).copy()

     X_train, x_test, Y_train, y_test = train_test_split(X, Y,random_state=111, test_size=0.3)
     X_train, x_val, Y_train, y_val=train_test_split(X_train, Y_train, random_state=111, test_size=0.3)
```

```
[75] X_train = np.array(X_train).reshape((X_train.shape[0],X_train.shape[1],1))
     x_test = np.array(x_test).reshape((x_test.shape[0], x_test.shape[1],1))

     Y_train = pd.get_dummies(Y_train)
     y_test = pd.get_dummies(y_test)
     y_val = pd.get_dummies(y_val)
```

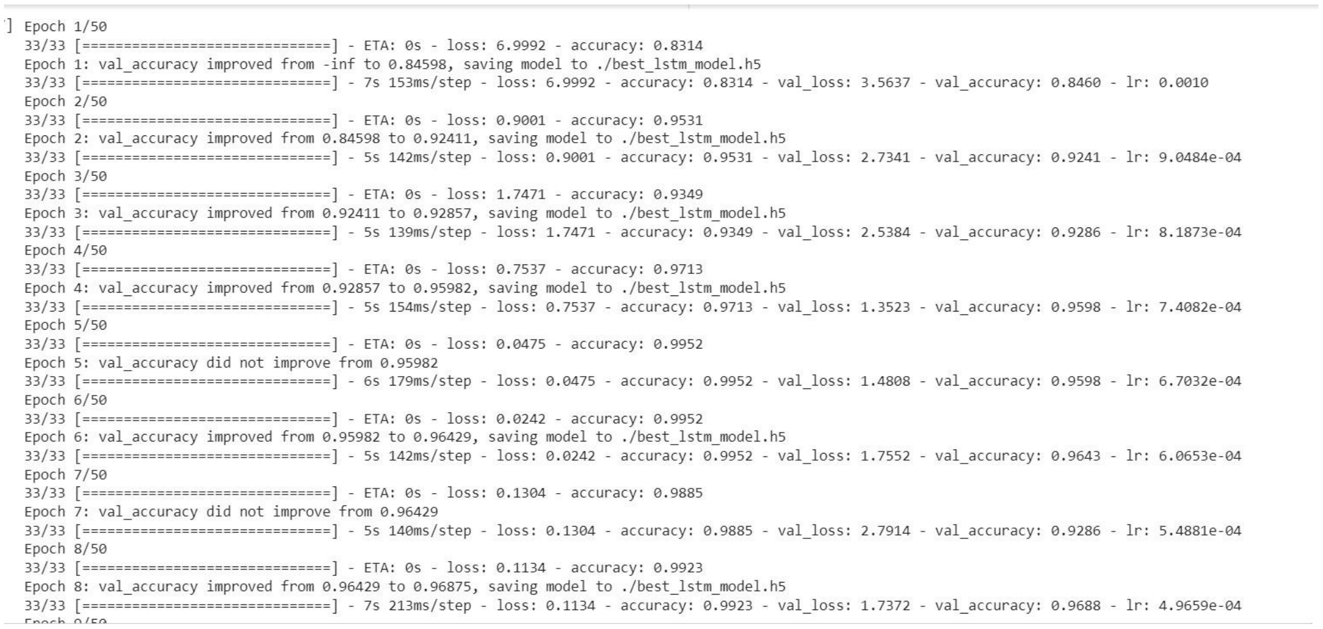
Fig 12: figure shows the data splitting to train and test set

The paper Emotion Recognition from EEG Signals Using Recurrent Neural Networks developed three models like RNN, GRU, LSTM. I have implemented all the models in the paper with the same dataset. In addition, implemented DNN and few classifier ensemble techniques with PCA with ensemble techniques for better results. Deep neural networks (DNN) were opted because it enables models to learn complicated characteristics and execute more heavy computational tasks more efficiently.

IMPLEMENTING THE MODEL: Now different model building is implemented, and their results are shown in pictorial representation. The below is the code, in the same way the rest of the models are built that is available in google collab file.

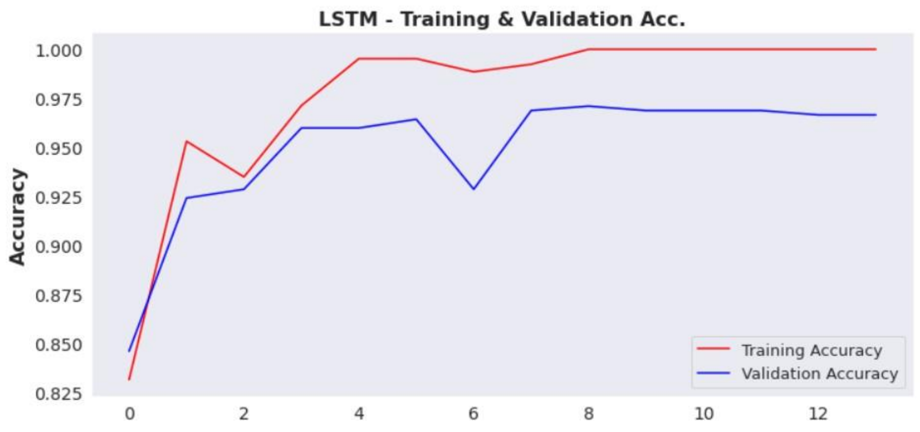
RESULTS:

LSTM Architecture results:

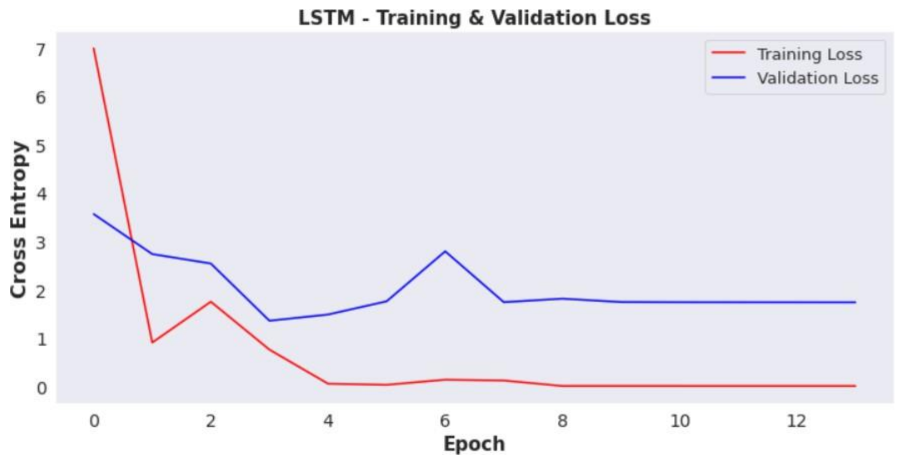


```
] Epoch 1/50
  33/33 [==============================] - ETA: 0s - loss: 6.9992 - accuracy: 0.8314
  Epoch 1: val_accuracy improved from -inf to 0.84598, saving model to ./best_lstm_model.h5
  33/33 [==============================] - 7s 153ms/step - loss: 6.9992 - accuracy: 0.8314 - val_loss: 3.5637 - val_accuracy: 0.8460 - lr: 0.0010
  Epoch 2/50
  33/33 [==============================] - ETA: 0s - loss: 0.9001 - accuracy: 0.9531
  Epoch 2: val_accuracy improved from 0.84598 to 0.92411, saving model to ./best_lstm_model.h5
  33/33 [==============================] - 5s 142ms/step - loss: 0.9001 - accuracy: 0.9531 - val_loss: 2.7341 - val_accuracy: 0.9241 - lr: 9.0484e-04
  Epoch 3/50
  33/33 [==============================] - ETA: 0s - loss: 1.7471 - accuracy: 0.9349
  Epoch 3: val_accuracy improved from 0.92411 to 0.92857, saving model to ./best_lstm_model.h5
  33/33 [==============================] - 5s 139ms/step - loss: 1.7471 - accuracy: 0.9349 - val_loss: 2.5384 - val_accuracy: 0.9286 - lr: 8.1873e-04
  Epoch 4/50
  33/33 [==============================] - ETA: 0s - loss: 0.7537 - accuracy: 0.9713
  Epoch 4: val_accuracy improved from 0.92857 to 0.95982, saving model to ./best_lstm_model.h5
  33/33 [==============================] - 5s 154ms/step - loss: 0.7537 - accuracy: 0.9713 - val_loss: 1.3523 - val_accuracy: 0.9598 - lr: 7.4082e-04
  Epoch 5/50
  33/33 [==============================] - ETA: 0s - loss: 0.0475 - accuracy: 0.9952
  Epoch 5: val_accuracy did not improve from 0.95982
  33/33 [==============================] - 6s 179ms/step - loss: 0.0475 - accuracy: 0.9952 - val_loss: 1.4808 - val_accuracy: 0.9598 - lr: 6.7032e-04
  Epoch 6/50
  33/33 [==============================] - ETA: 0s - loss: 0.0242 - accuracy: 0.9952
  Epoch 6: val_accuracy improved from 0.95982 to 0.96429, saving model to ./best_lstm_model.h5
  33/33 [==============================] - 5s 142ms/step - loss: 0.0242 - accuracy: 0.9952 - val_loss: 1.7552 - val_accuracy: 0.9643 - lr: 6.0653e-04
  Epoch 7/50
  33/33 [==============================] - ETA: 0s - loss: 0.1304 - accuracy: 0.9885
  Epoch 7: val_accuracy did not improve from 0.96429
  33/33 [==============================] - 5s 140ms/step - loss: 0.1304 - accuracy: 0.9885 - val_loss: 2.7914 - val_accuracy: 0.9286 - lr: 5.4881e-04
  Epoch 8/50
  33/33 [==============================] - ETA: 0s - loss: 0.1134 - accuracy: 0.9923
  Epoch 8: val_accuracy improved from 0.96429 to 0.96875, saving model to ./best_lstm_model.h5
  33/33 [==============================] - 7s 213ms/step - loss: 0.1134 - accuracy: 0.9923 - val_loss: 1.7372 - val_accuracy: 0.9688 - lr: 4.9659e-04
```

Fitting results by using the LSTM architecture.

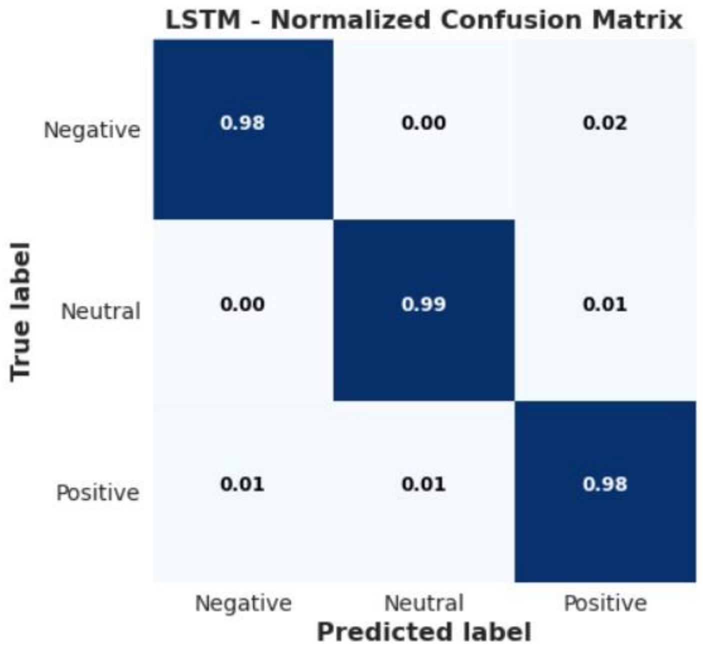


Accuracy curve by using LSTM



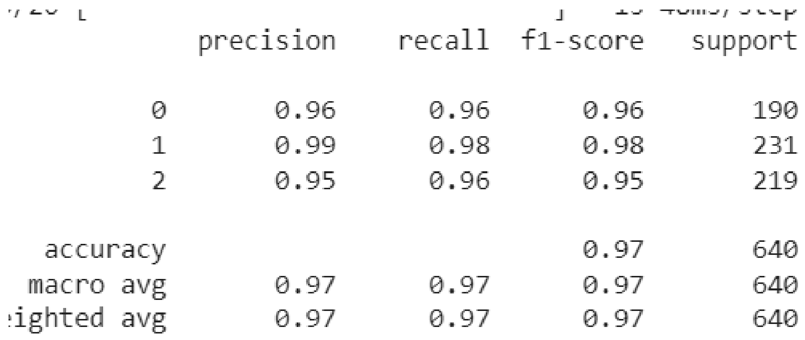
Loss curve by using LSTM

Normalized confusion matrix



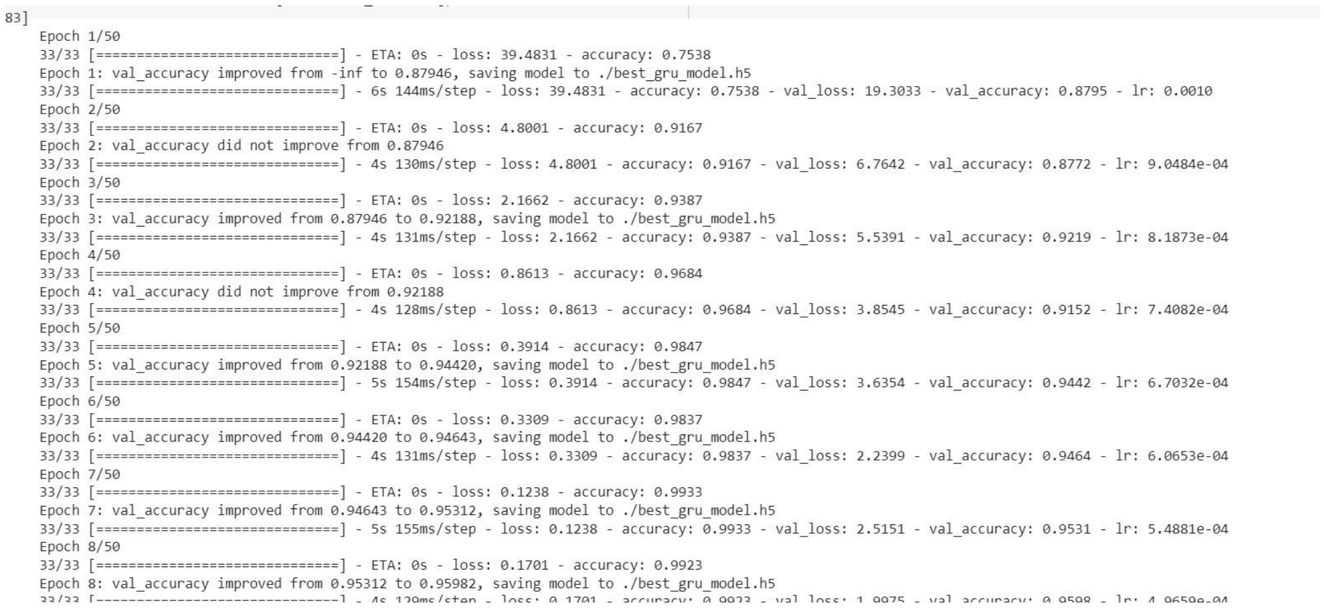
Confusion matrix by using LSTM model.

Confusion matrix displays the test data of 640 samples for the three emotions of negative, neutral, and positive. The performance measures are calculated by using TP, TN, FP, and FN values that are taken from the confusion matrix. From the below calculations, the F1 score is 0.95, and the accuracy of the model by using LSTM is 97%.



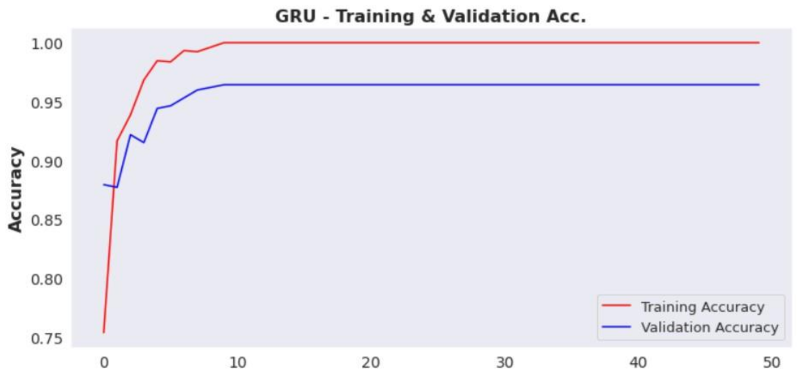
```
                precision    recall  f1-score   support

           0       0.96      0.96      0.96       190
           1       0.99      0.98      0.98       231
           2       0.95      0.96      0.95       219

    accuracy                           0.97       640
   macro avg       0.97      0.97      0.97       640
ighted avg       0.97      0.97      0.97       640
```

GRU Architecture results:



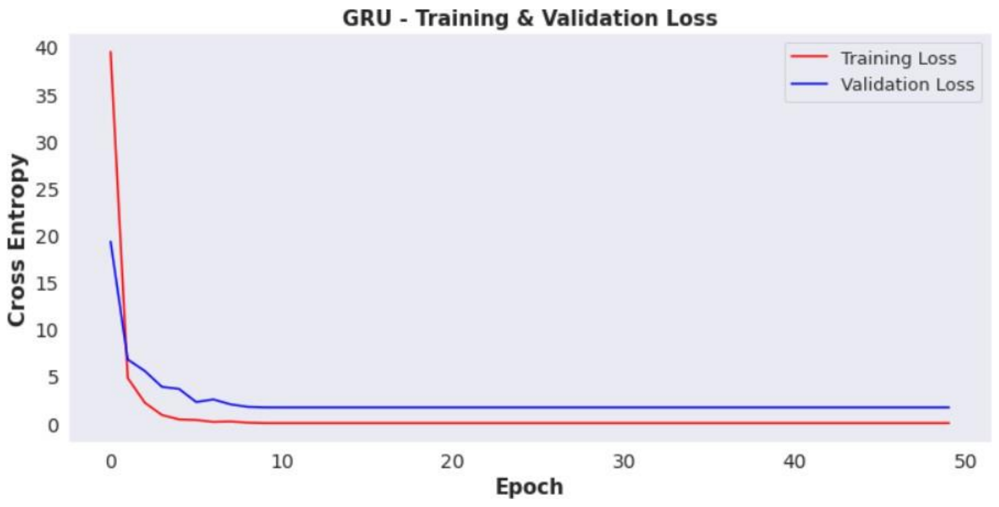
```
83]
    Epoch 1/50
    33/33 [==============================] - ETA: 0s - loss: 39.4831 - accuracy: 0.7538
    Epoch 1: val_accuracy improved from -inf to 0.87946, saving model to ./best_gru_model.h5
    33/33 [==============================] - 6s 144ms/step - loss: 39.4831 - accuracy: 0.7538 - val_loss: 19.3033 - val_accuracy: 0.8795 - lr: 0.0010
    Epoch 2/50
    33/33 [==============================] - ETA: 0s - loss: 4.8001 - accuracy: 0.9167
    Epoch 2: val_accuracy did not improve from 0.87946
    33/33 [==============================] - 4s 130ms/step - loss: 4.8001 - accuracy: 0.9167 - val_loss: 6.7642 - val_accuracy: 0.8772 - lr: 9.0484e-04
    Epoch 3/50
    33/33 [==============================] - ETA: 0s - loss: 2.1662 - accuracy: 0.9387
    Epoch 3: val_accuracy improved from 0.87946 to 0.92188, saving model to ./best_gru_model.h5
    33/33 [==============================] - 4s 131ms/step - loss: 2.1662 - accuracy: 0.9387 - val_loss: 5.5391 - val_accuracy: 0.9219 - lr: 8.1873e-04
    Epoch 4/50
    33/33 [==============================] - ETA: 0s - loss: 0.8613 - accuracy: 0.9684
    Epoch 4: val_accuracy did not improve from 0.92188
    33/33 [==============================] - 4s 128ms/step - loss: 0.8613 - accuracy: 0.9684 - val_loss: 3.8545 - val_accuracy: 0.9152 - lr: 7.4082e-04
    Epoch 5/50
    33/33 [==============================] - ETA: 0s - loss: 0.3914 - accuracy: 0.9847
    Epoch 5: val_accuracy improved from 0.92188 to 0.94420, saving model to ./best_gru_model.h5
    33/33 [==============================] - 5s 154ms/step - loss: 0.3914 - accuracy: 0.9847 - val_loss: 3.6354 - val_accuracy: 0.9442 - lr: 6.7032e-04
    Epoch 6/50
    33/33 [==============================] - ETA: 0s - loss: 0.3309 - accuracy: 0.9837
    Epoch 6: val_accuracy improved from 0.94420 to 0.94643, saving model to ./best_gru_model.h5
    33/33 [==============================] - 4s 131ms/step - loss: 0.3309 - accuracy: 0.9837 - val_loss: 2.2399 - val_accuracy: 0.9464 - lr: 6.0653e-04
    Epoch 7/50
    33/33 [==============================] - ETA: 0s - loss: 0.1238 - accuracy: 0.9933
    Epoch 7: val_accuracy improved from 0.94643 to 0.95312, saving model to ./best_gru_model.h5
    33/33 [==============================] - 5s 155ms/step - loss: 0.1238 - accuracy: 0.9933 - val_loss: 2.5151 - val_accuracy: 0.9531 - lr: 5.4881e-04
    Epoch 8/50
    33/33 [==============================] - ETA: 0s - loss: 0.1701 - accuracy: 0.9923
    Epoch 8: val_accuracy improved from 0.95312 to 0.95982, saving model to ./best_gru_model.h5
    33/33 [==============================] - 4s 129ms/step - loss: 0.1701 - accuracy: 0.9923 - val_loss: 1.9975 - val_accuracy: 0.9598 - lr: 4.9659e-04
```

Fitting results by using the GRU architecture.



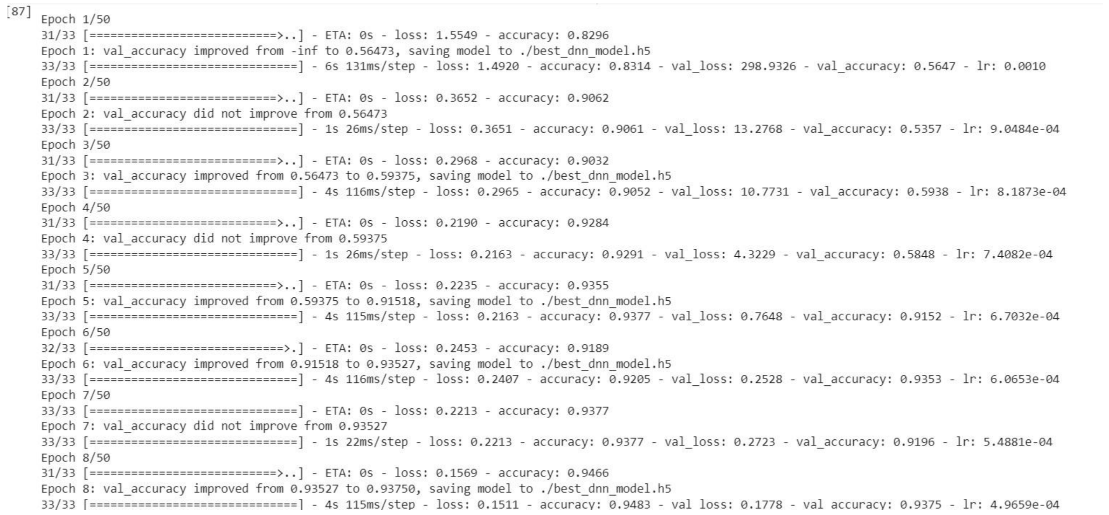
Accuracy curve by using GRU

By Krishnaveni



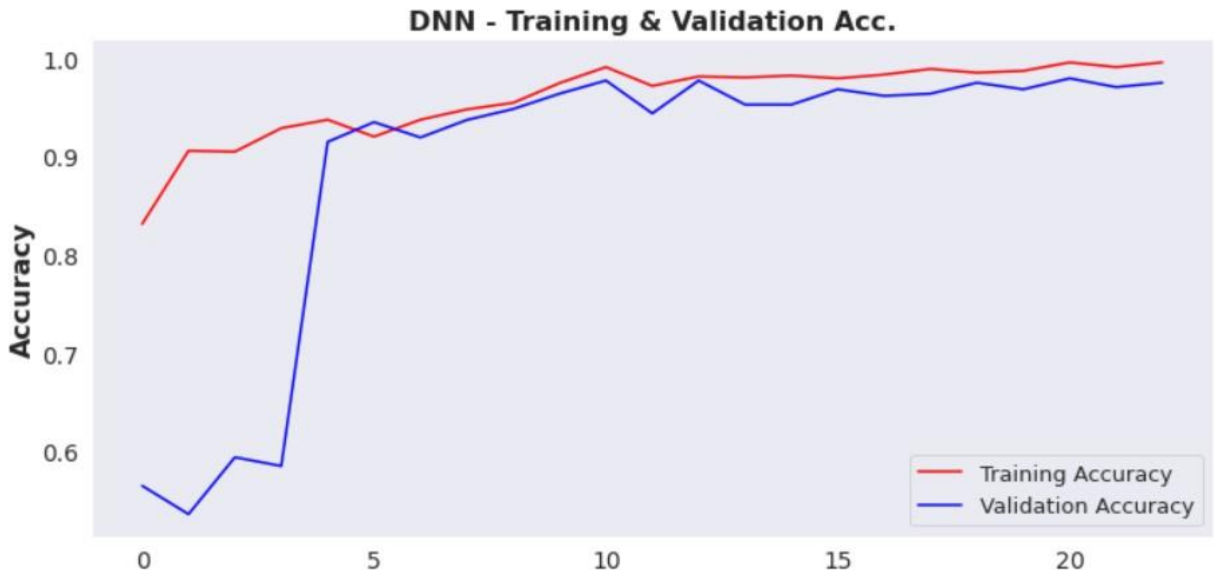
Loss curve by using GRU

DNN Architecture:

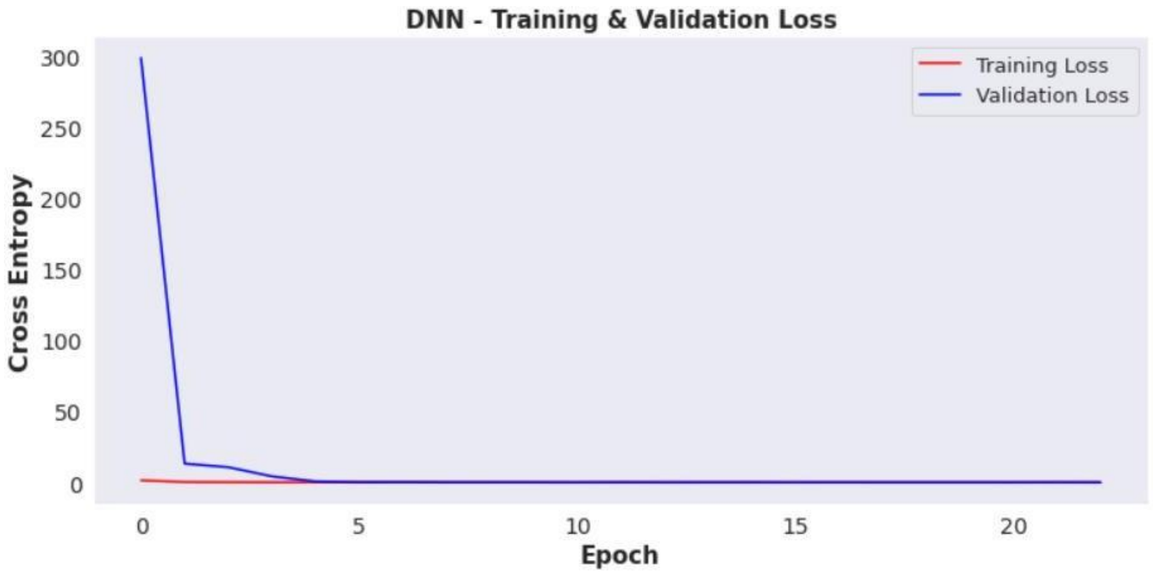


```
[87]
Epoch 1/50
31/33 [===========================>..] - ETA: 0s - loss: 1.5549 - accuracy: 0.8296
Epoch 1: val_accuracy improved from -inf to 0.56473, saving model to ./best_dnn_model.h5
33/33 [==============================] - 6s 131ms/step - loss: 1.4920 - accuracy: 0.8314 - val_loss: 298.9326 - val_accuracy: 0.5647 - lr: 0.0010
Epoch 2/50
31/33 [===========================>..] - ETA: 0s - loss: 0.3652 - accuracy: 0.9062
Epoch 2: val_accuracy did not improve from 0.56473
33/33 [==============================] - 1s 26ms/step - loss: 0.3651 - accuracy: 0.9061 - val_loss: 13.2768 - val_accuracy: 0.5357 - lr: 9.0484e-04
Epoch 3/50
31/33 [===========================>..] - ETA: 0s - loss: 0.2968 - accuracy: 0.9032
Epoch 3: val_accuracy improved from 0.56473 to 0.59375, saving model to ./best_dnn_model.h5
33/33 [==============================] - 4s 116ms/step - loss: 0.2965 - accuracy: 0.9052 - val_loss: 10.7731 - val_accuracy: 0.5938 - lr: 8.1873e-04
Epoch 4/50
31/33 [===========================>..] - ETA: 0s - loss: 0.2190 - accuracy: 0.9284
Epoch 4: val_accuracy did not improve from 0.59375
33/33 [==============================] - 1s 26ms/step - loss: 0.2163 - accuracy: 0.9291 - val_loss: 4.3229 - val_accuracy: 0.5848 - lr: 7.4082e-04
Epoch 5/50
31/33 [===========================>..] - ETA: 0s - loss: 0.2235 - accuracy: 0.9355
Epoch 5: val_accuracy improved from 0.59375 to 0.91518, saving model to ./best_dnn_model.h5
33/33 [==============================] - 4s 115ms/step - loss: 0.2163 - accuracy: 0.9377 - val_loss: 0.7648 - val_accuracy: 0.9152 - lr: 6.7032e-04
Epoch 6/50
32/33 [============================>.] - ETA: 0s - loss: 0.2453 - accuracy: 0.9189
Epoch 6: val_accuracy improved from 0.91518 to 0.93527, saving model to ./best_dnn_model.h5
33/33 [==============================] - 4s 116ms/step - loss: 0.2407 - accuracy: 0.9205 - val_loss: 0.2528 - val_accuracy: 0.9353 - lr: 6.0653e-04
Epoch 7/50
33/33 [==============================] - ETA: 0s - loss: 0.2213 - accuracy: 0.9377
Epoch 7: val_accuracy did not improve from 0.93527
33/33 [==============================] - 1s 22ms/step - loss: 0.2213 - accuracy: 0.9377 - val_loss: 0.2723 - val_accuracy: 0.9196 - lr: 5.4881e-04
Epoch 8/50
31/33 [===========================>..] - ETA: 0s - loss: 0.1569 - accuracy: 0.9466
Epoch 8: val_accuracy improved from 0.93527 to 0.93750, saving model to ./best_dnn_model.h5
33/33 [==============================] - 4s 115ms/step - loss: 0.1511 - accuracy: 0.9483 - val_loss: 0.1778 - val_accuracy: 0.9375 - lr: 4.9659e-04
```

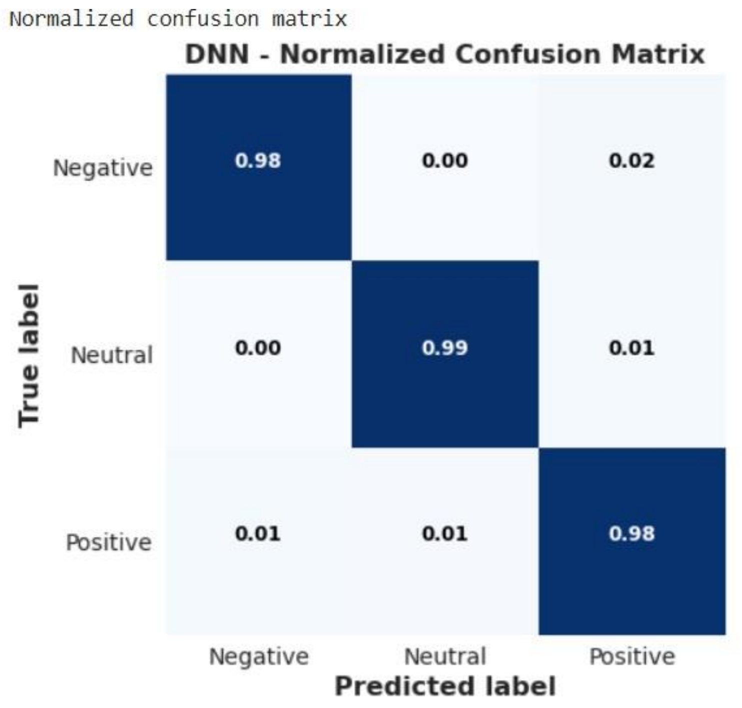
Fitting results by using the DNN architecture.



Accuracy curve by using DNN



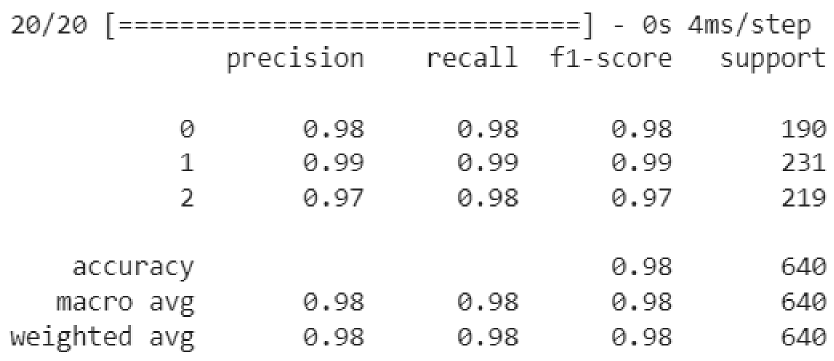
Loss curve by using DNN



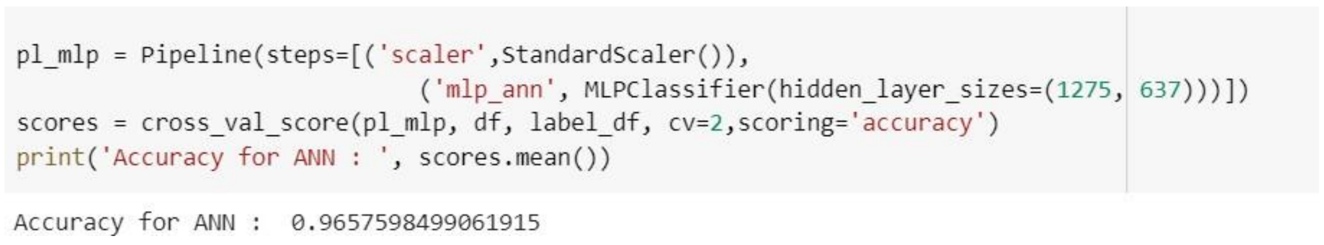
	Negative	Neutral	Positive
Negative	0.98	0.00	0.02
Neutral	0.00	0.99	0.01
Positive	0.01	0.01	0.98

Confusion matrix by using DNN model.

Confusion matrix displays the test data of 640 samples for the three emotions of negative, neutral, and positive. The performance measures are calculated by using TP, TN, FP, and FN values that are taken from the confusion matrix. From the below calculations, the F1 score is 0.97, and the accuracy of the model by using DNN is 98%.



```
20/20 [==============================] - 0s 4ms/step
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       190
           1       0.99      0.99      0.99       231
           2       0.97      0.98      0.97       219

    accuracy                           0.98       640
   macro avg       0.98      0.98      0.98       640
weighted avg       0.98      0.98      0.98       640
```

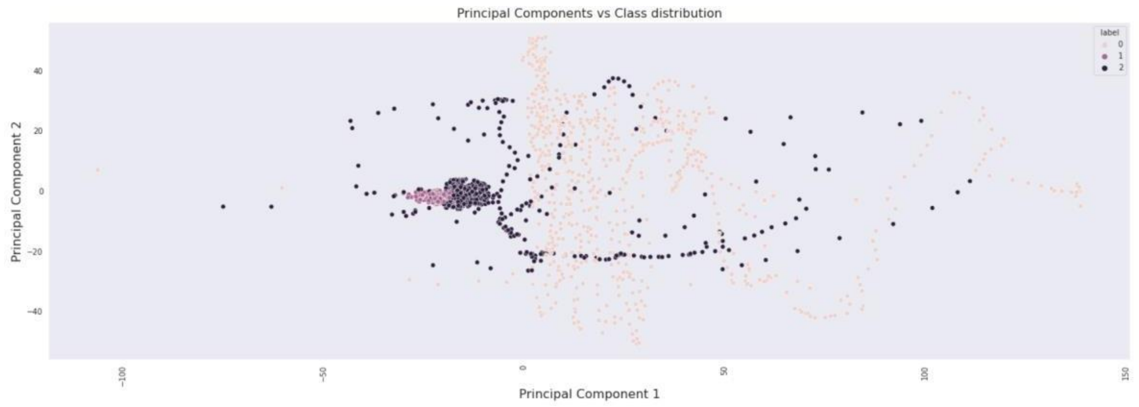


```
pl_mlp = Pipeline(steps=[('scaler',StandardScaler()),
                         ('mlp_ann', MLPClassifier(hidden_layer_sizes=(1275, 637)))])
scores = cross_val_score(pl_mlp, df, label_df, cv=2,scoring='accuracy')
print('Accuracy for ANN : ', scores.mean())
```

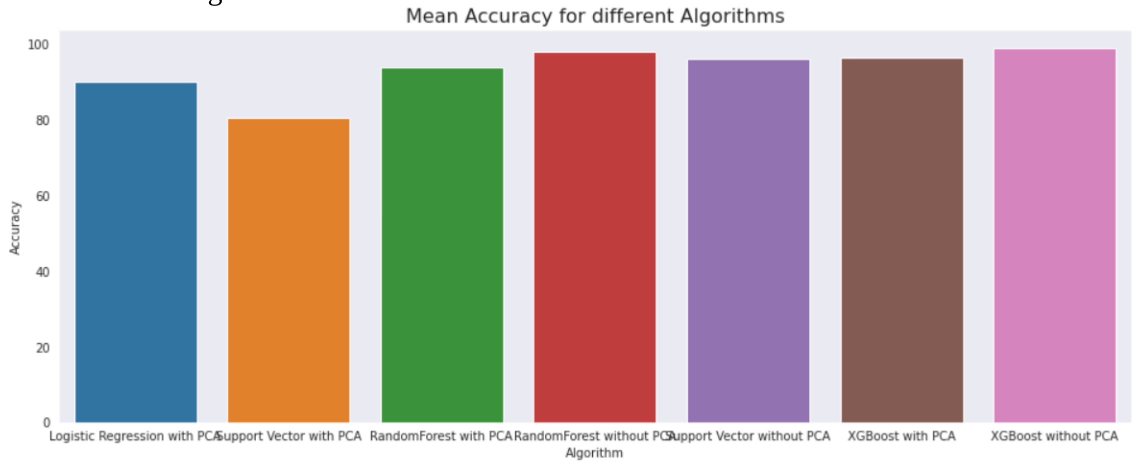
```
Accuracy for ANN :  0.9657598499061915
```

PCA:

Pipelines creation is done for execution of classifier models after dimensionality reduction. Using Standard scaler method data preprocessing is executed, and the dimensions are reduced by using PCA. The image shows the target value label components distribution.



Different classifier models were implemented for emotion analysis and their corresponding means accuracies are shown in the below figure.



Mean accuracy of different algorithms

By Krishnaveni

Discussion:

Understanding human emotions through biological brain signals is becoming more appealing. Detecting emotion with EEG data entails a series of processes that must be completed to meet the standards of a brain-computer interface (BCI). EEG signals are commonly employed in emotion detection because devices for detecting the signals are non-invasive, low-cost, and wearable. Electroencephalography (EEG) is a proven and cost-effective method for measuring brain activity. Non-invasive detection of emotional states has the potential to be beneficial in a variety of sectors, including human-robot interaction and mental healthcare. It may provide another level of engagement between the user and the gadget, as well as give physical information that is not dependent on verbal communication. The paper that was taken for reference has implemented RNN, LSTM and GRU for recognition of emotions from EEG Signals. In my project I have made few changes while implementing the models for the available dataset. I have used more number of units in layers of model architecture. Fore instance, in the existing dataset they have used 128 unit whereas I have opted 256 units in the model implementation for LSTM, GRU. An additional DNN architecture was also employed for emotion recognition which gave the best accuracy results in comparison with other models. The reason for this might be usage of Standard Scale for the DNN data. Finally, I believe the performance can also be improved a bit further with data processing and hyperparameter tuning. Also, tried implementing different classifier with PCA and PCA to study the best accuracy model for detection of emotion from EEG Signals.

Conclusion:

A new deep network is proposed to classify EEG signals for emotion recognition. Different models - DNN, LSTM, GRU, ANN and different ensemble models have been created. The EEG Brain Wave Database was used in the investigations. The accuracy attained with the DNN model is 98%, 97% with the LSTM model, and 95.9% with the GRU model, 96.4% with ANN and 96.2% with Random Forest without PCA. DNN outperformed both the LSTM and GRU models in terms of accuracy. In this investigation, LSTM outperformed GRU by 1% in terms of accuracy.. A bigger number of emotions can be recognized if a larger number of samples are supplied. In the future, these networks will be used to distinguish emotions in multimodal datasets and real-time data.

Acknowledgements:

This Project helps the people in identifying the Emotions using EEG signals by implementing BCI Methods. In an ideal world, it would assist patients suffering neuro problems and helps to communicate their feeling easily.

My lecturer led me through every step, from setting up a google colab notebook to highly complicated ideas of algorithms, making it easier for folks with no coding or IT expertise.

References:

1. **https://www.mdpi.com/2079-9292/11/15/2387**
2. **https://www.researchgate.net/publication/335173767_A_Deep_Evolutionary_Approach_to_Bioinspired_Classifier_Optimisation_for_Brain-Machine_Interaction**
3. **https://www.researchgate.net/publication/329403546_Mental_Emotional_Sentiment_Classification_with_an_EEG-based_Brain-machine_Interface**