A) Data Set Description

WESAD is collection of multi modal sensory data for predicting stress levels in human beings. Given problem is a **classification problem.** Data set is collected from 17 different subjects, but sensors on 2 subjects were malfunctioned thus data from S1 and S12 are missing. Therefore WESAD data set has **15 folders** namely S1 to S17 where each corresponds to one subject, each folder has 5 files, out of which only .pkl files are of interest, since they contain combined data of all **10 sensors.**

Attributes of data set are **4,255,300 rows x 11 columns** available for each subject, with many **outliers and missing values.** Data set is unbalanced, since data collected for one particular label or class is 3 times more than data collected for all other classes. Additionally data collected form first sensor device is 21 times more than data acquired from second sensor device.

There are **11 input variables** collected from two devices namely a chest-worn device (RespiBAN) and wrist worn device (Empatica E4). Chest worn samples are 4,060,772, wrist worn samples are 194,528. Input variables can be further elaborated as following:

The RespiBAN device provides electrocardiogram (ECG), electrodermal activity (EDA), electromyogram (EMG), respiration, body temperature, and 3-Axis Acceleration. All signals are sampled at 700 Hz. Thus contributing to 6 input variables.

Then Empatica E4 device provides blood volume pulse (BVP, 64 Hz), electrodermal activity (EDA, 4 Hz), body temperature (4 Hz), and three-axis acceleration (32 Hz). Accelerometer has resolution of 1/64g for all 3 x, y and z axis. Data from EDA is provided in μS. Temperature sensor is collected in °C. Thus contributing to remaining 4 input variables.

Thus data set has **10 input variables.** Since data is collected from 2 different devices, thus for synchronization data from both devices is stored in .pkl only after subject double taps his arm.

Output variables are given in 11th column in data set as output label value for all input variable readings taken in that time period. **Thus supervised learning is possible** using this data set. Label ID ranges from 0 to 7 where 0 = not defined / transient, 1 = baseline, 2 = stress, 3 = amusement, 4 = meditation, 5/6/7 = should be ignored in this dataset.

Data type of WESAD data set is float64, double-precision floating-point format. There are no strings or integers present in data set.

WESAD data set is provided by UCI, which is publicly available. Data set **is not clean,** organized into one pickle file, has more samples for chest worn device and less samples for wrist worn device, has **many outliers, missing values but supports supervised learning models.** It is evident that WESAD data set has more samples for Respiban device with respect to data collected from Empatica E4 device. There are **4,000,000 missing values** in WESAD data set for Empatica E4 sensors.

WESAD data set has combined stress data set from different labs and provides mood evaluation report with each data set before and after completing the test. Testing has been ensured in lab environment and total data set size is 2.1 GB.

B) Data set visualization

Data set is collected from 15 patients stored in separate folders, each folder has one pickle file that contains data from chest and wrist devices called respiban and empatica. Following insights are collected after running various visualization tests.

1.1 Classification problem with supervised learning.

Given data set has well defined output labels thus data set can be used to implement supervised learning algorithms. Since data set has limited output labels, qualitative outputs with finite states ranging from state 1 to state 7 thus classification models can be developed using this data set.

1.2 Data set has unbalanced data samples for each output label.

Given data set has 8 output labels, with different number of samples shown below.

0 = not defined / transient state has 800800 samples.

1 = baseline state has 800800 samples.

2 = stress state has 430500 samples.

3 = amusement state has 253400 samples.

4 = meditation state has 537599 samples.

5/6/7 = should be ignored in this data set.

Since each output state has different number of samples against each label, thus it can be concluded that WESAD data set is highly unbalanced and accuracy matrix should be used for misclassification, rather recall matrix should be used. The same numbers are represented in figure 1.2 as a bar chart for visualization against all 8 labels.

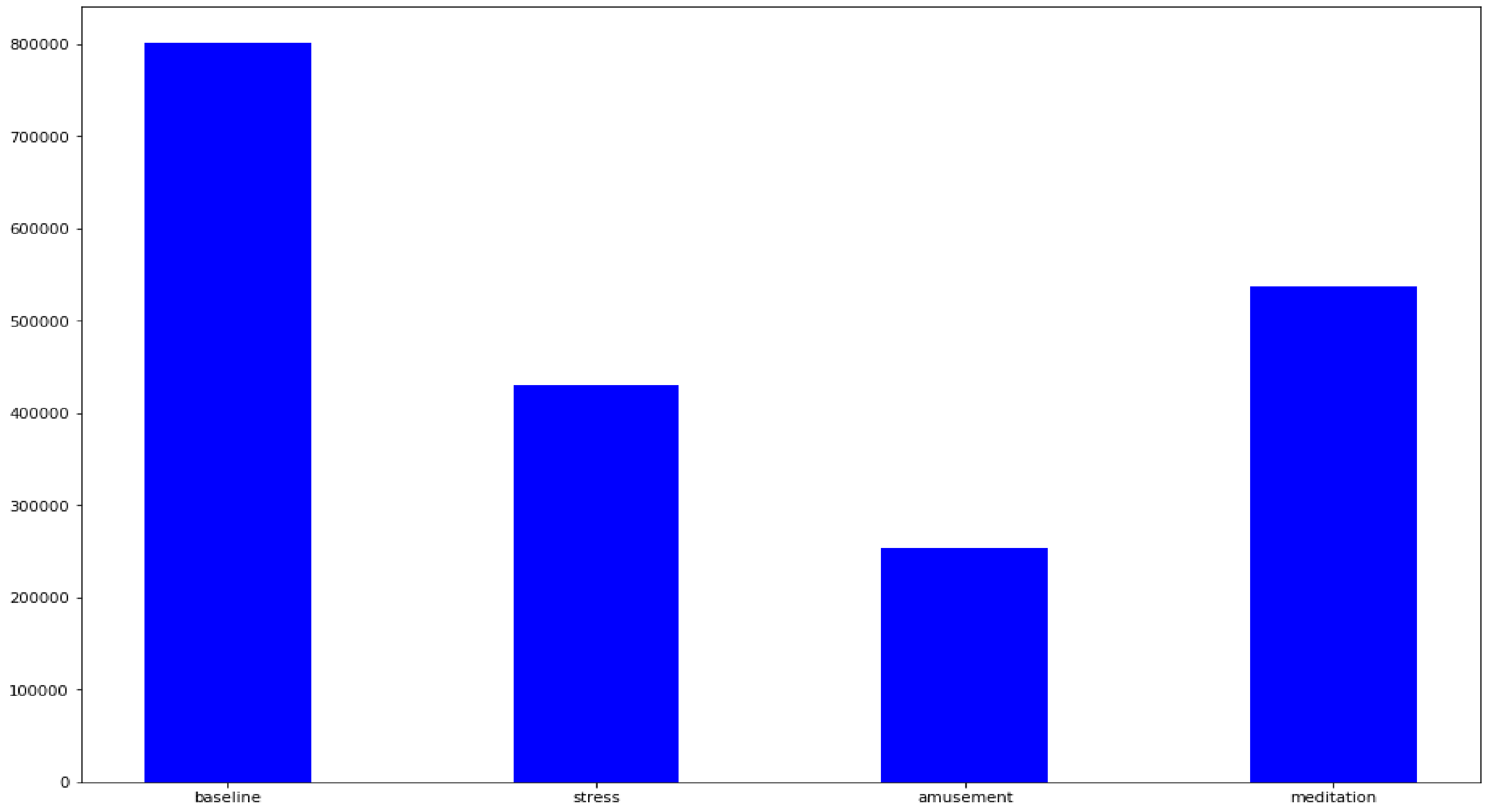


Figure 1.2: Represents total number of samples collected for each label.

Figure 1.2 represent total number of samples collected for each label, which are not equal to each other, therefore it can be concluded that given data set is highly unbalanced. Samples collected for baseline state is more than other states, similarly samples for stress state are two times more than amusement state. It can be concluded that data set is unbalanced and therefore precision and recall misclassification matrix should be used. Accuracy score will not useful in this case.

<u>1.3 Data collected from respiban and empatica is unbalanced and has many missing values.</u>

There are 4255300 samples collected from respiban device, whereas there are only 194528 samples collected from empatica device, consider figure 1.3.

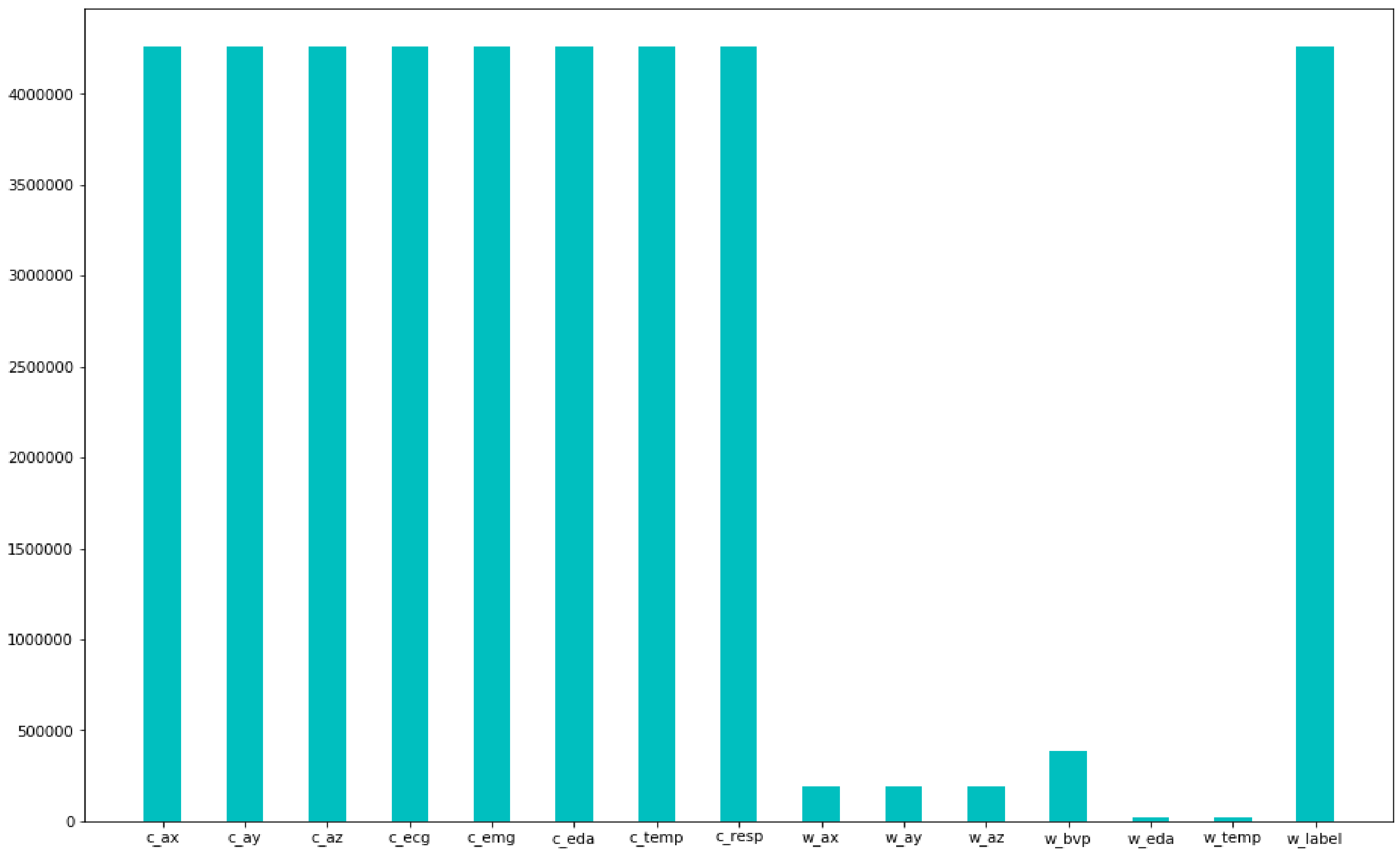


Figure 1.3: Unbalanced samples collected from respiban and empatica devices.

Figure 1.3 shows total number of samples collected from respiban and empatica devices, c_ax, c_ay, c_az, c_ecg, c_emg, c_eda, c_temp belong to respiban device, which have 4,255,300 samples, whereas others belong to empatica device.

Empatica sensors like w_ax, w_ay, w_az, w_bvp, w_temp have less than 500,000 samples each. Thus respiban samples are 21 times more than empatica samples.

1.4 NA Values Present in Dataset

There are 4255300 labels present in dataset which have all 4255300 values from chest device, but only 194528 samples from wrist device are present in dataset. Thus 4060772 NA values are present due to wrist device. Table 1.4.1 shown below describes total NA values present in the data set.

Table 1.4.1: 4060772 NA values present from wrist device.

Device	Values per Column
Chest Device	4255300
Wrist Device	194528
Total NA Values = 4060772 from wrist device	

Therefore I had 2 options either to drop 4060772 samples of chest device and then form my data frame so that data frame can have equal number of samples from each device. Or drop 194528 samples from wrist device and form a data frame with 4255300 samples exclusively from chest device NA values. Thus I dropped data samples from wrist device so that my data set can have more samples. Moreover sensors in wrist device are almost similar to sensors present in chest device. Therefore it can be concluded that data collected is highly unbalanced and data from empatica (wrist) device will be ignored thereafter, since similar sensors are used in each device located on different spots.

1.5 Forming Pandas Data Frame from Dictionary

WESAD dataset is very tricky and is in form of dictionary, which could not be directly converted into a data frame. Also it had 4060772 missing values from wrist device. Therefore I decided to remove 194528 wrist device samples and keep all chest device 4255300 samples. Thus data frame contains values from chest device only. Moreover sensors in wrist device are almost similar to sensors present in chest device.

1.6 Raw data set has outliers

Table 1.6.1 shows list of IQR values for each sensor associated with respiban device on chest.

Table 1.6.1: IQR values vs respiban sensors.

Sensor name	IQR value
c_ax	0.271200
c_ay	0.054000
c_az	0.507400
c_ecg	0.112335
c_emg	0.012314
c_eda	0.746918
c_temp	1.240173
c_resp	3.100586
w_label	2.000000

Consider table 1.6.1 that displays list of all IQR values for all 8 sensors. It is evident that Resp sensor has highest IQR values thus it has highest spread and maximum number of outliers will be present in this data set. IQR values are followed by sensor data of Temp and EDA , which further assures that data of these sensors have high spread and outliers. All IQR values are close to each other and fairly above

0.5, thus it can be concluded that data set has outliers in all 8 different sensor data. WESAD data set has many outliers, consider figure 1.6.2, where box plots show outliers associated with all sensors in respiban device.

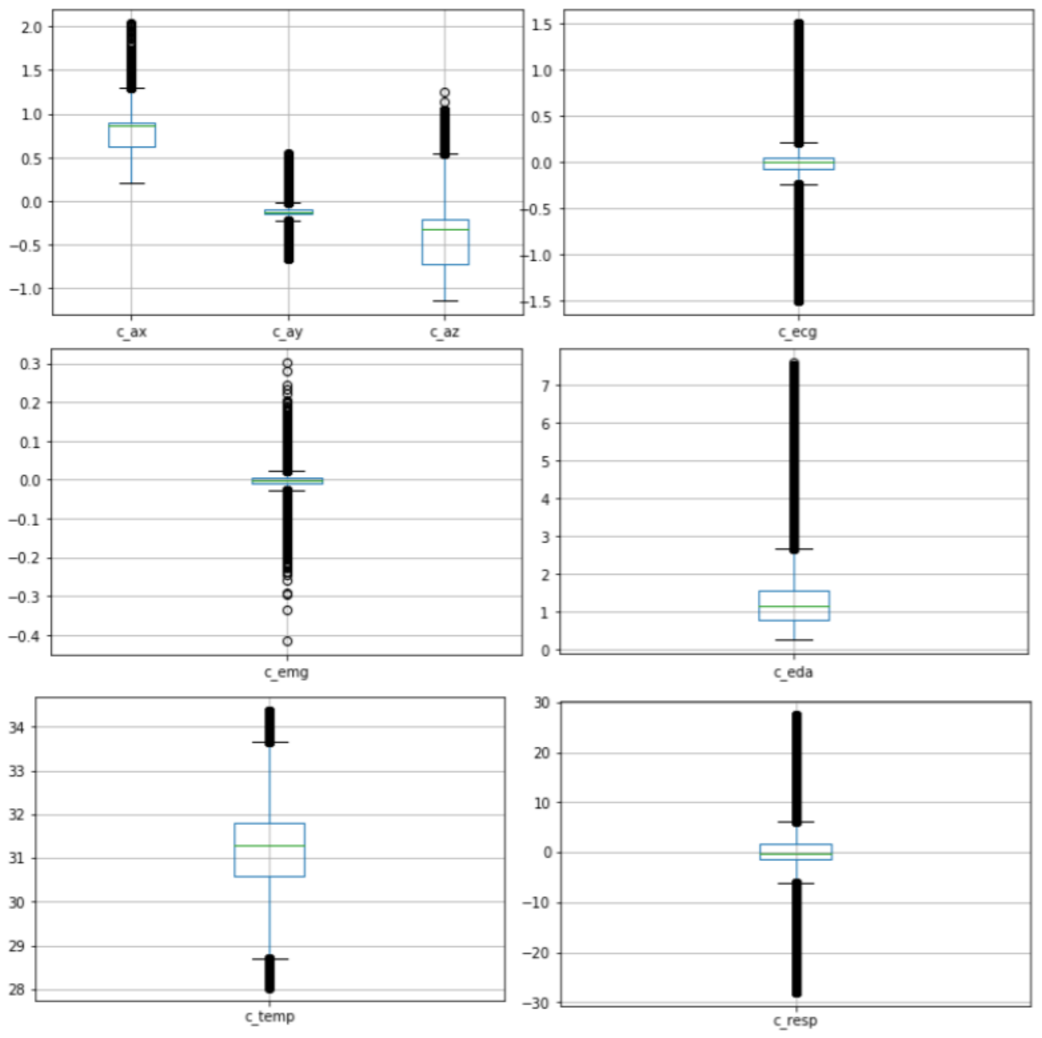


Figure 1.6.2: Box plots for all 8 sensors.

Figure 1.6.2 shows box plots for all 8 sensors inside respiban device, where many outliers are present. EMG, EDA, Temp and Resp sensors have maximum number of outliers. Outliers in this data set will cause misclassification problems and hinder the accuracy of my model.

1.7 Estimating variation range of data set.

Figure 1.7 shows line plot of all 8 sensor values, for estimating their range individually.

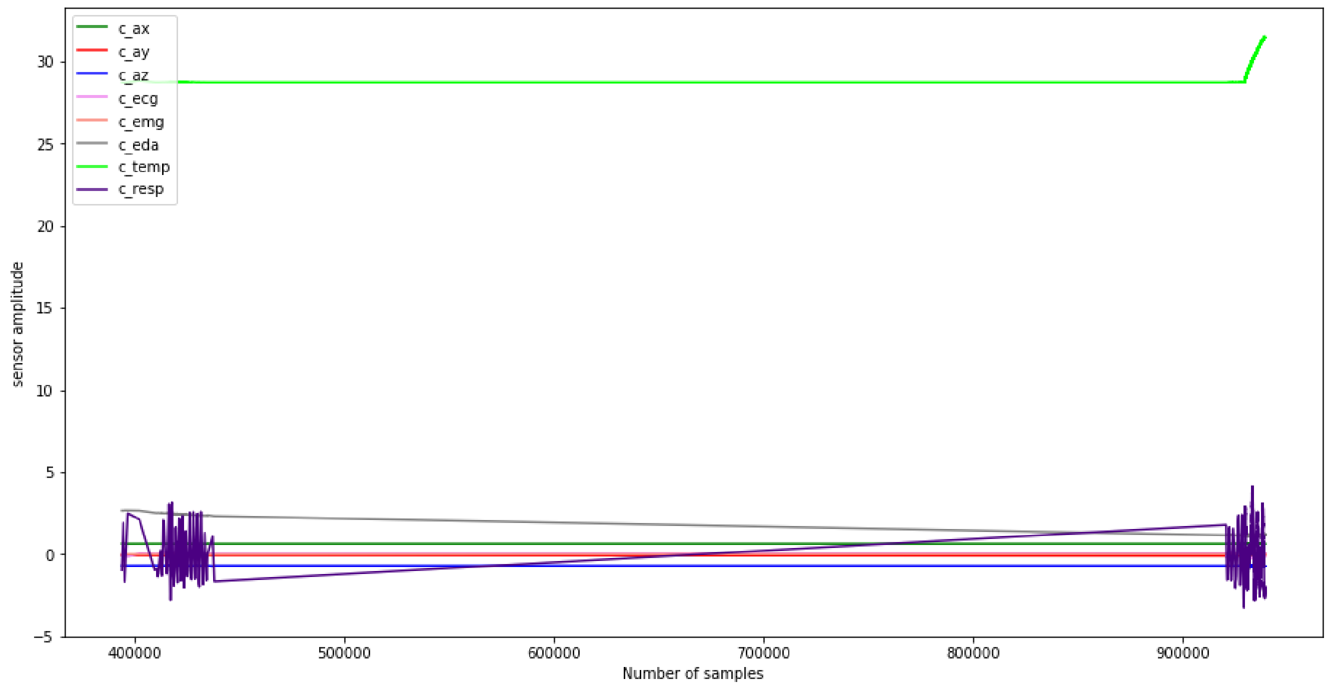
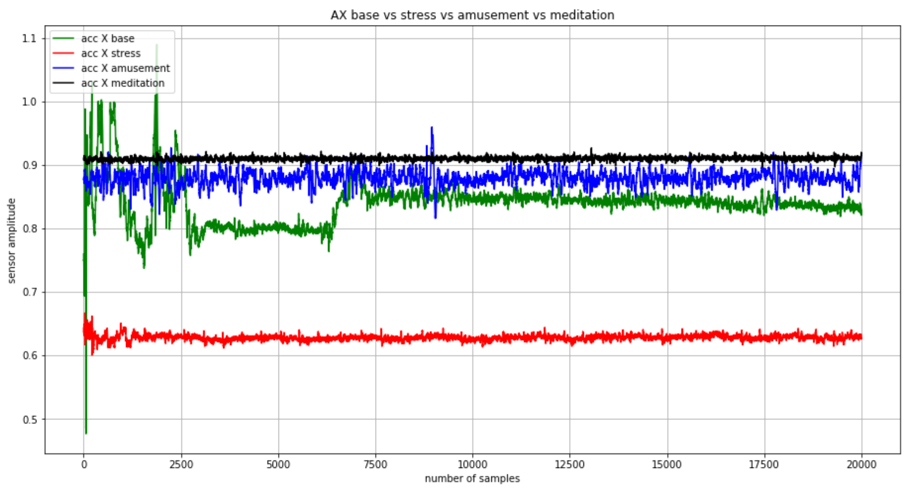


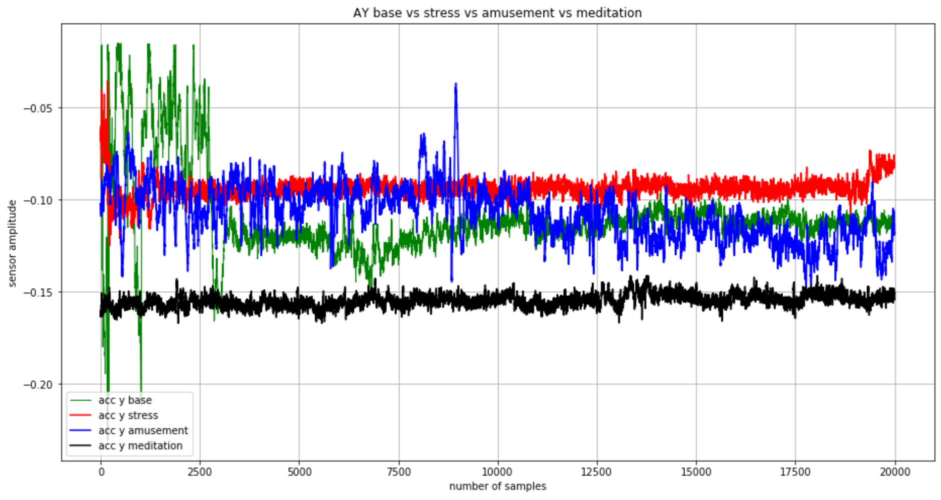
Figure 1.7: Approximation of peak to peak variation in sensor data.

Consider figure 1.7 which shows that acceleration x,y,z axis, ecg, emg, eda and resp sensor values are between static interval of -2 to 4. Whereas temp sensor values are distinctly static between 26 to 34 degree centigrade. This confirms that given data is clean with certain fluctuations but no outliers.

1.8 Analyzing sensor data for 4 distinct states.

All 8 sensor values are plotted separately for 4 distinct states as given above namely baseline state, stress state, amusement state and meditation state. General trends of data are estimated from figures given below. Figure 1.8.1 displays accelerometer x, y and z axis data points for all 4 states.





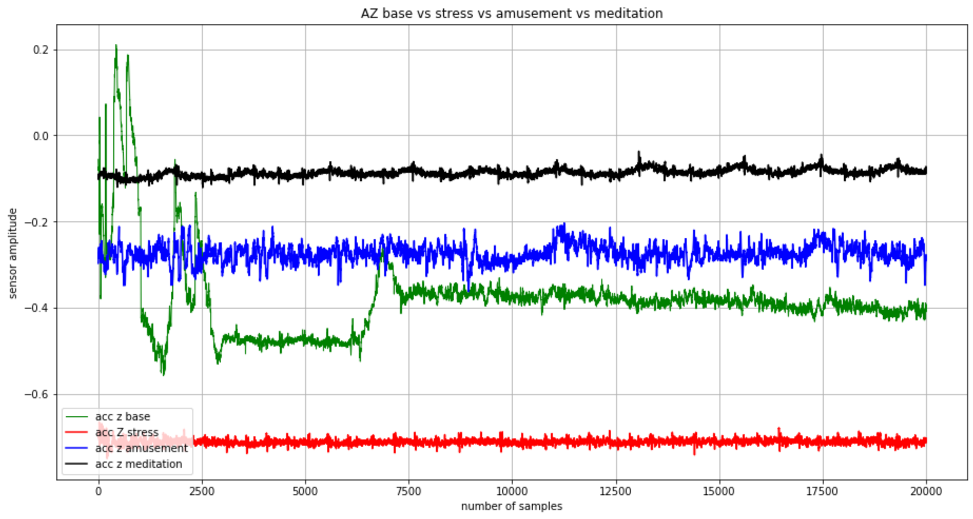


Figure 1.8.1: Accelerometer x, y and z axis data for 4 different states.

It can be estimated form figure 1.8.1 that under stress state given subject does minimum activities in x and z direction whereas under meditation state minimum activity is done under y direction. Similarly baseline state has a unique feature of sudden drop in all three axis simultaneously at a given time. Unfortunately accelerometer does not given distinctive data for amusement state. Consider figure 1.8.2 for understanding temperature sensor pattern for 4 different states.

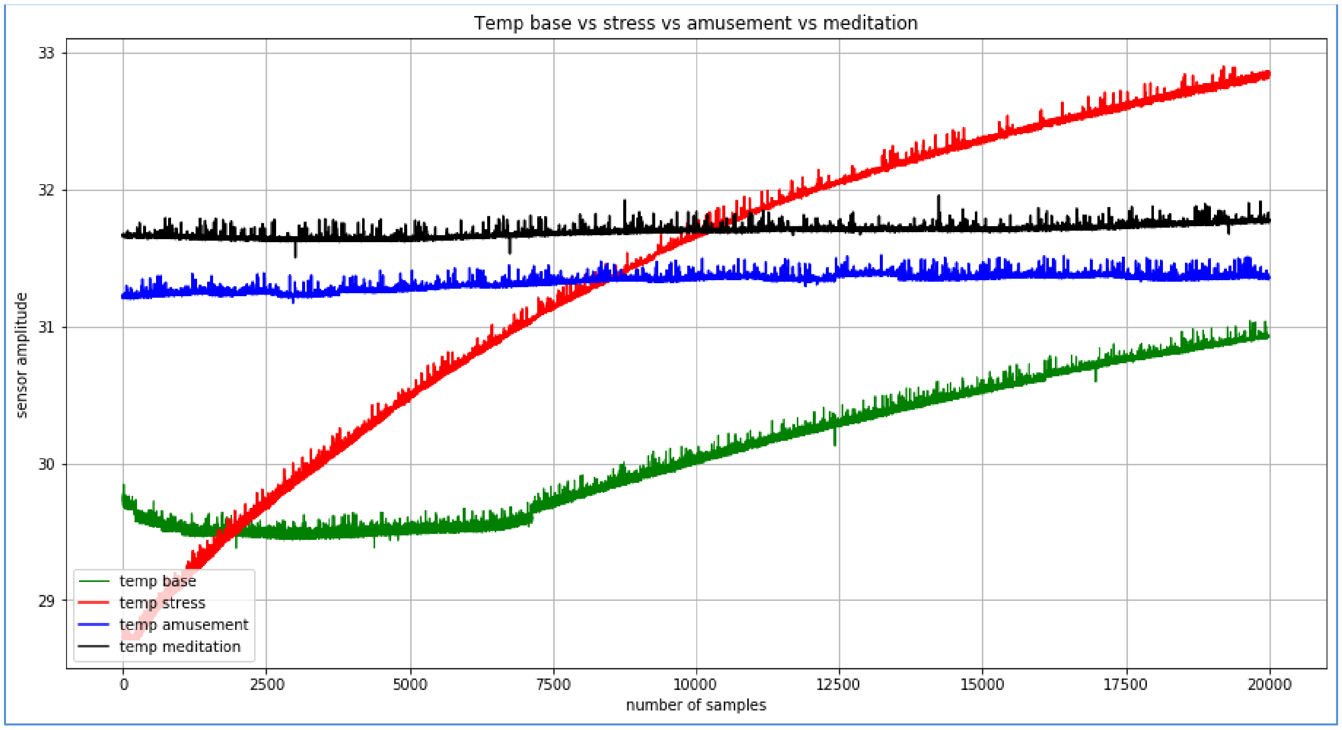


Figure 1.8.2: Temperature sensor values for baseline, stress, amusement, meditation state.

Figure 1.8.2 clearly indicates that under stress state temperature values increase gradually and attain max value after 15,000 samples. Similarly meditation values are constant at 31.8 degree centigrade throughout. Whereas temperature values for amusement state are static throughout at 31.2 degree centigrade. On the contrary baseline state temperature values gradually rise at constant slope and become constant at 31 degree centigrade. Consider figure 1.8.3 for analyzing ECG sensor data for all 4 states.

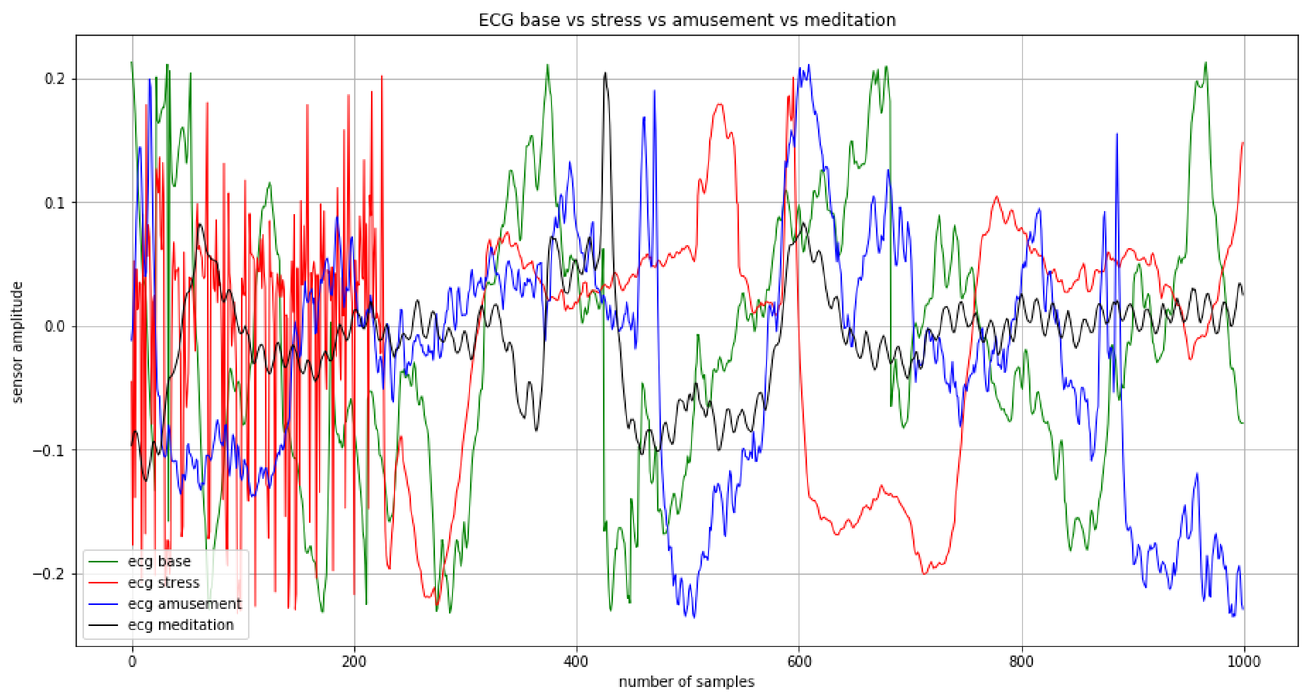


Figure 1.8.3: ECG sensor values for baseline, stress, amusement, meditation state.

Figure 1.8.3 clearly displays that heart rate has maximum frequency under stress state, where as minimum frequency is obtained at meditation state. On the contrary with respect to baseline state amusement state has steeper slope. Consider figure 1.8.4 for analyzing EDA sensor data for all 4 states.

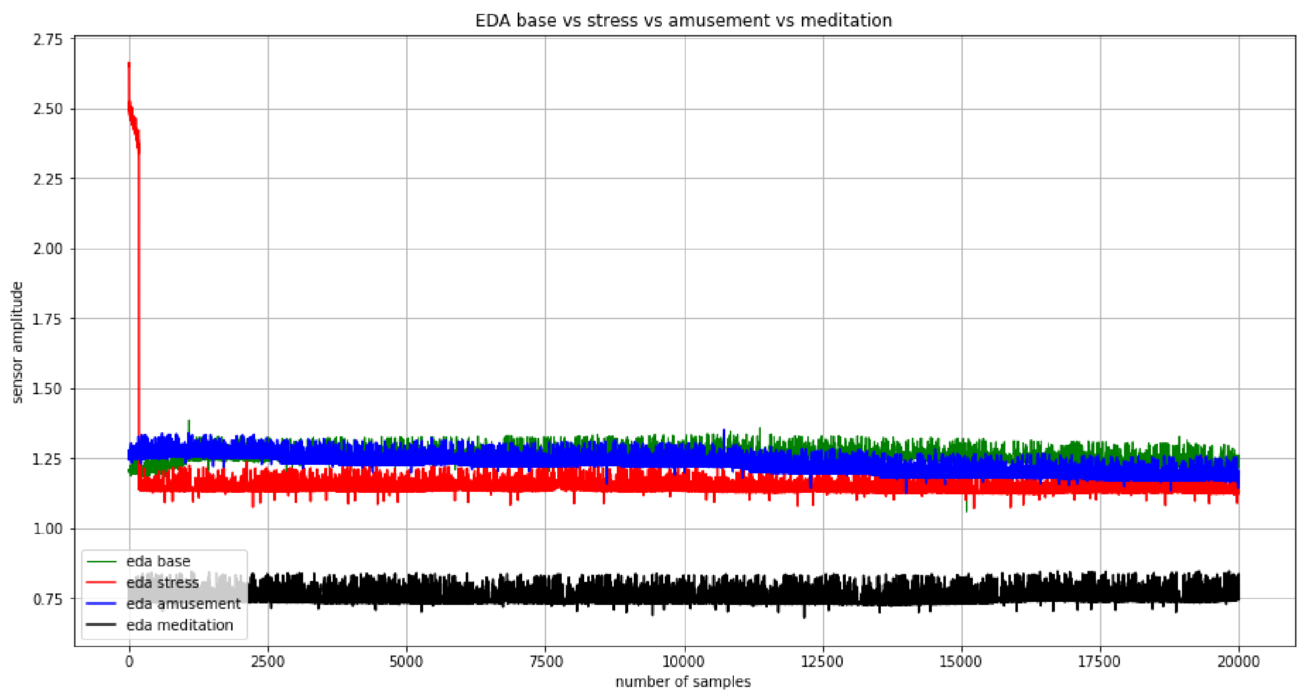


Figure 1.8.4: EDA sensor values for baseline, stress, amusement, meditation state.

Figure 1.8.4 clearly shows that minimum sensor values at meditation state are lowest and statically maintained between 0.75 to 0.85. whereas, for stress state eda sensor values are above meditation state but below remaining two states. Unfortunately baseline state and amusement state have similar values throughout. Consider figure 1.8.5 for analyzing EMG sensor data for all 4 states.

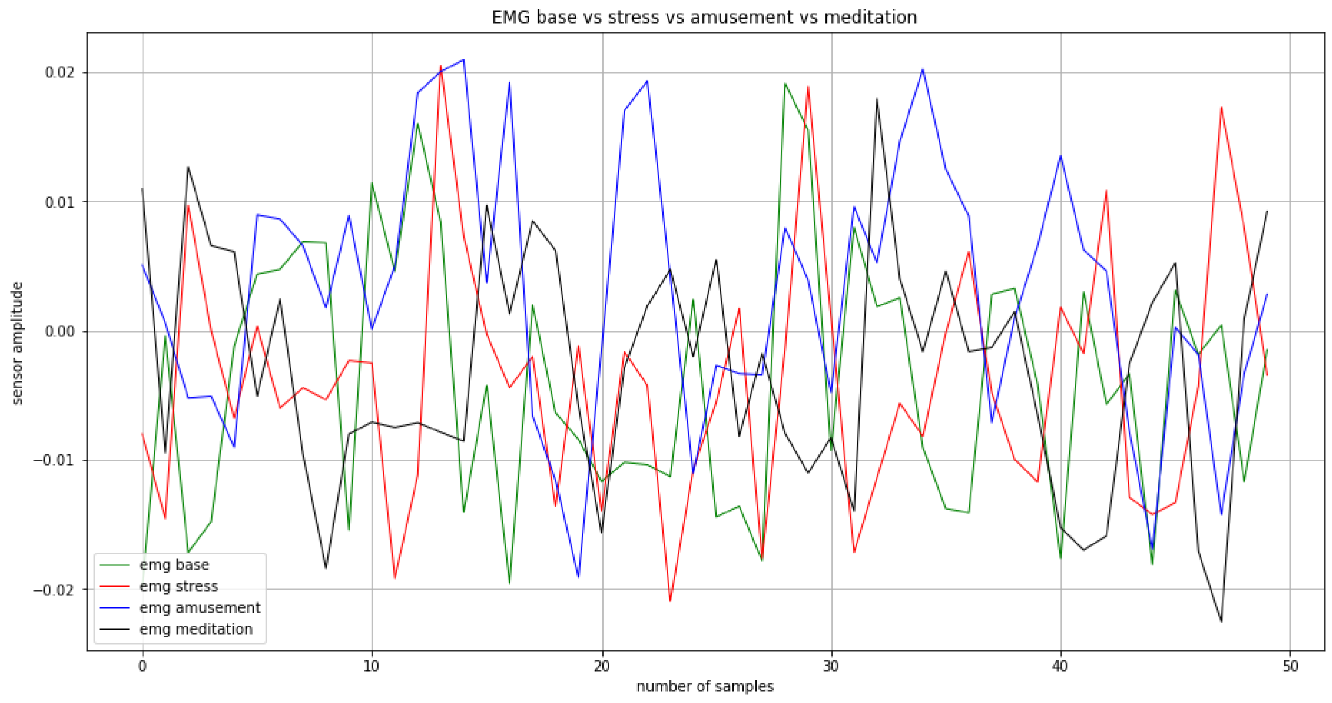


Figure 1.8.5: EMG sensor values for baseline, stress, amusement, meditation state.

Figure 1.8.5 shows that amusement state and stress state emg data is completely opposite in phase this means if one increases then other decreases at same time. Whereas for meditation state emg data frequency is smallest. On contrary baseline data can be used as reference which has higher frequency than meditation state and lower frequency than stress state. Consider figure 1.8.6 for analyzing respiratory sensor data for all 4 states.

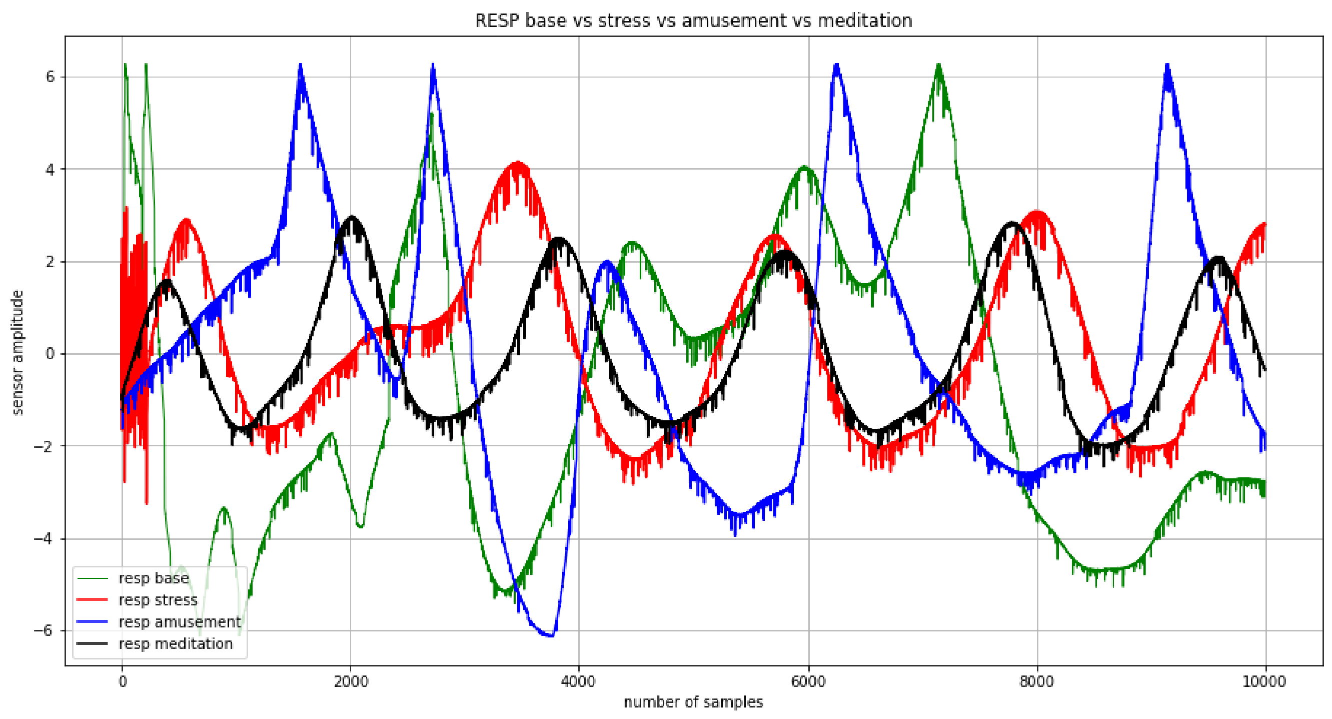
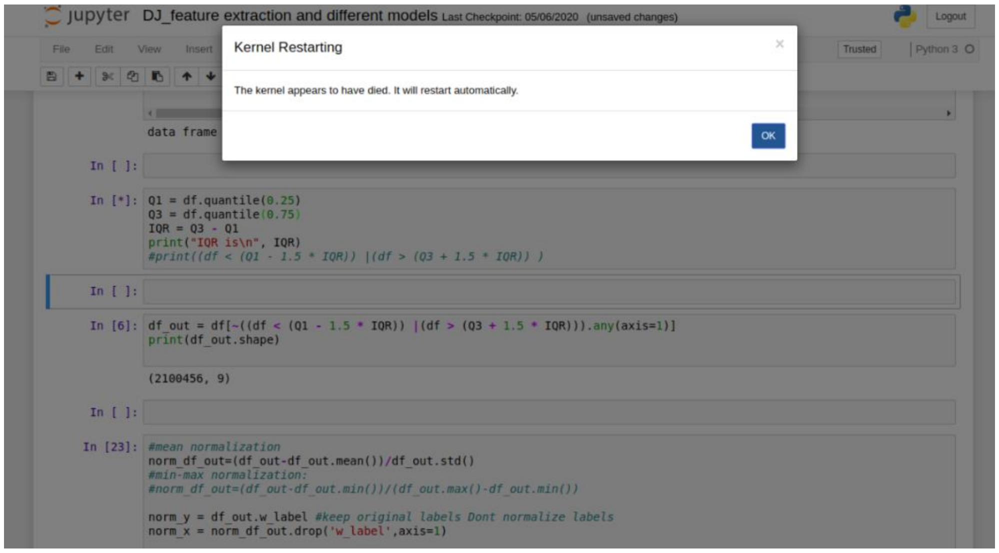


Figure 1.8.6: Respiratory sensor values for baseline, stress, amusement, meditation state.

It is evident from figure 1.8.6 that steepest slope is present in amusement state, which signifies sudden change in respiratory rate. Constant frequency and lowest peak to peak deviation around 6 units is a significant characteristic of meditation state. Similarly constant frequency and higher peak to peak deviation around 10 units is a significant characteristic of baseline state. Stress state has uneven slope which signifies quick change in breathing pattern, but signal shape is sinusoidal unlike amusement state shape which is triangular.

It should be worth mentioning that since data set is very large, thus a global classifier is created. In which only name of subject can be changed for visualizing and training and testing different models. Please check figure 1.8.7 that shows kernel hangs and then restarts if all data is combined together.



DJ_feature extraction and different models Last Checkpoint: 05/06/2020 (unsaved changes)

Kernel Restarting

The kernel appears to have died. It will restart automatically.

```
data frame

In [ ]:

In [*]: Q1 = df.quantile(0.25)
        Q3 = df.quantile(0.75)
        IQR = Q3 - Q1
        print("IQR is\n", IQR)
        #print((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR)) )

In [ ]:

In [6]: df_out = df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
        print(df_out.shape)

        (2100456, 9)

In [ ]:

In [23]: #mean normalization
         norm_df_out=(df_out-df_out.mean())/df_out.std()
         #min-max normalization:
         #norm_df_out=(df_out-df_out.min())/(df_out.max()-df_out.min())

         norm_y = df_out.w_label #keep original labels Dont normalize labels
         norm_x = norm_df_out.drop('w_label',axis=1)
```

Figure-1.8.7: Kernel will restart automatically if data from all subjects is combined together.

Figure 1.8.7 shows that kernel will restart automatically if data from all subjects is combined together. Therefore models are developed and trained using a global classifier as mentioned in the code and shown in figure 1.8.8.

```
data_set = '/home/dhananjai/Downloads/Spring_2020/Machine_Learning/ML_Project/dataset/WESAD/'
s2_path = data_set + 'S2/S2.pkl'
#DO-NOT enable all at once otherwise Laptop goes to sleep due to very huge data size.
#s3_path = data_set + 'S3/S3.pkl'
#s4_path = data_set + 'S4/S4.pkl'
#s5_path = data_set + 'S5/S5.pkl'
#s6_path = data_set + 'S6/S6.pkl'
#s7_path = data_set + 'S7/S7.pkl
#s8_path = data_set + 'S8/S8.pkl'
#s9_path = data_set + 'S9/S9.pkl'
#s10_path = data_set + 'S10/S10.pkl'
#s11_path = data_set + 'S11/S11.pkl'
#s13_path = data_set + 'S13/S13.pkl'
#s14_path = data_set + 'S14/S14.pkl'
#s15_path = data_set + 'S15/S15.pkl'
#s16_path = data_set + 'S16/S16.pkl'
#s17_path = data_set + 'S17/S17.pkl'
```

Figure-1.8.8: Global classifier for handling all subjects.

Please note that data is visualized and trained for one subject but tested rigorously for 7 different subjects. Moreover results are provided for the best model and performance is analyzed for all 7 subjects differently. Results are properly documented and presented for all different subjects individually in later sections.

C) Data Set Cleaning

As concluded above that dataset initially has many values beyond IQR, thus following sections will perform normalization, feature selection and removing outliers.

2.1 Removing NA values

NA values in form of wrist device samples are already removed in visualization section above.

2.2 Removing Outliers

IQR is used to remove outliers from each sensor beyond certain rage. Consider figure 2.2 which shows new box plots for all 8 sensors.

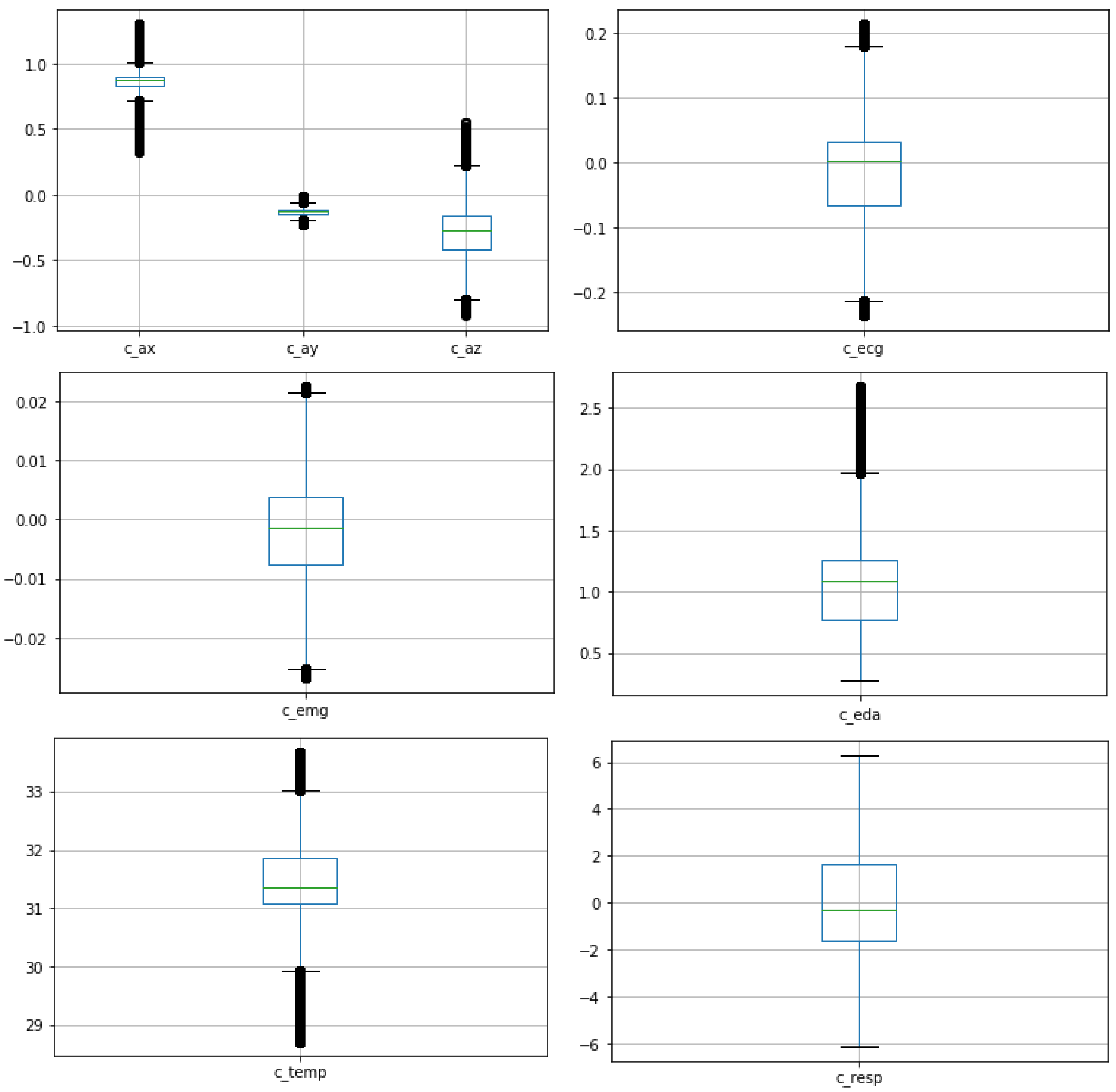
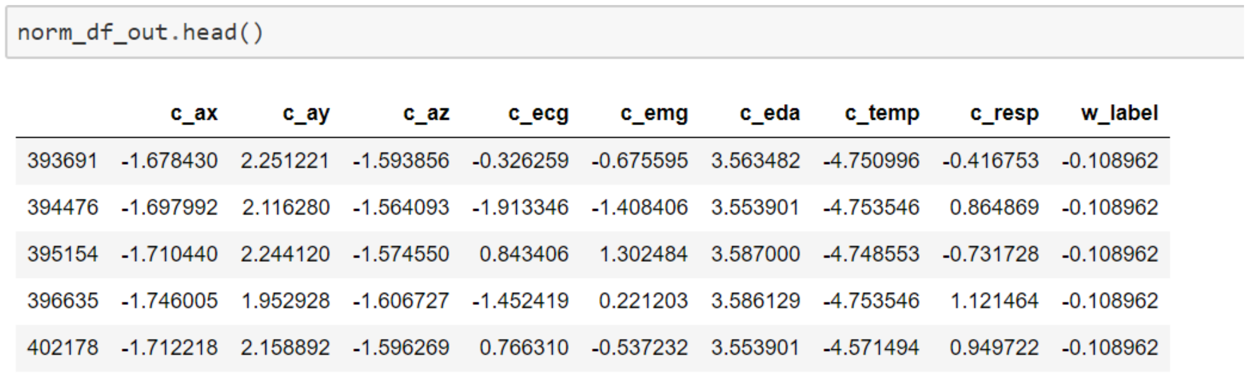


Figure 2.2: Box plots after cleaning sensor data and removing outliers

Figure 2.2 shows cleaned sensor data, where outliers are completely removed, thus this data can be used for creating and training models. Therefore data points between (Q1 - 1.5 * IQR) to (Q3 + 1.5 * IQR) are considered as outliers and were successfully removed from the data set.

2.3 Cleaned Dataset Head()



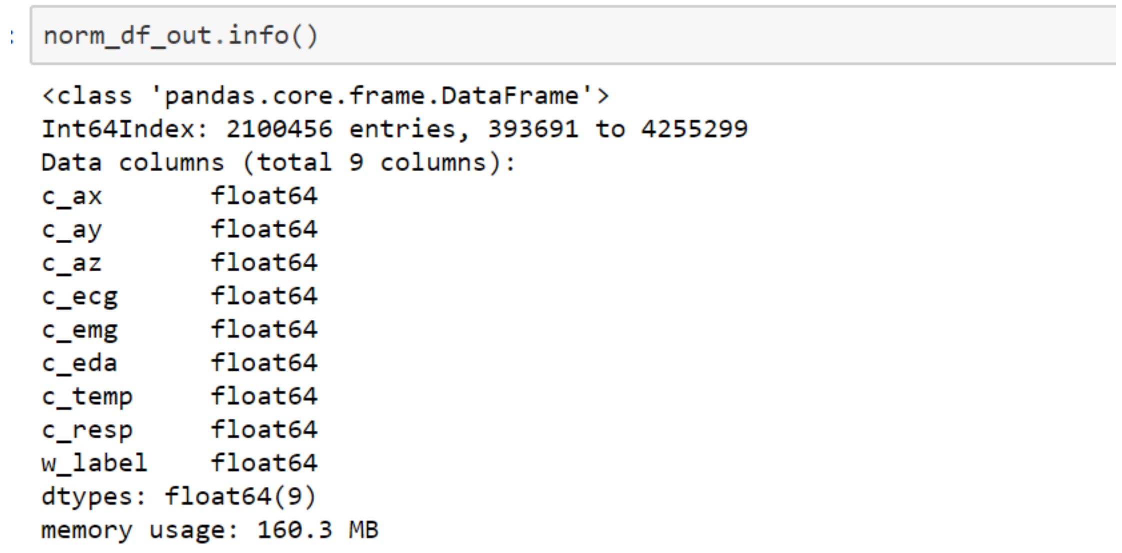
```
norm_df_out.head()
```

	c_ax	c_ay	c_az	c_ecg	c_emg	c_eda	c_temp	c_resp	w_label
393691	-1.678430	2.251221	-1.593856	-0.326259	-0.675595	3.563482	-4.750996	-0.416753	-0.108962
394476	-1.697992	2.116280	-1.564093	-1.913346	-1.408406	3.553901	-4.753546	0.864869	-0.108962
395154	-1.710440	2.244120	-1.574550	0.843406	1.302484	3.587000	-4.748553	-0.731728	-0.108962
396635	-1.746005	1.952928	-1.606727	-1.452419	0.221203	3.586129	-4.753546	1.121464	-0.108962
402178	-1.712218	2.158892	-1.596269	0.766310	-0.537232	3.553901	-4.571494	0.949722	-0.108962

Figure-2.3: First 5 values of cleaned data.

Figure displays cleaned and normalized data developed after removing all outliers and NaN values from complex dictionary structure.

2.4 Cleaned Dataset Info()



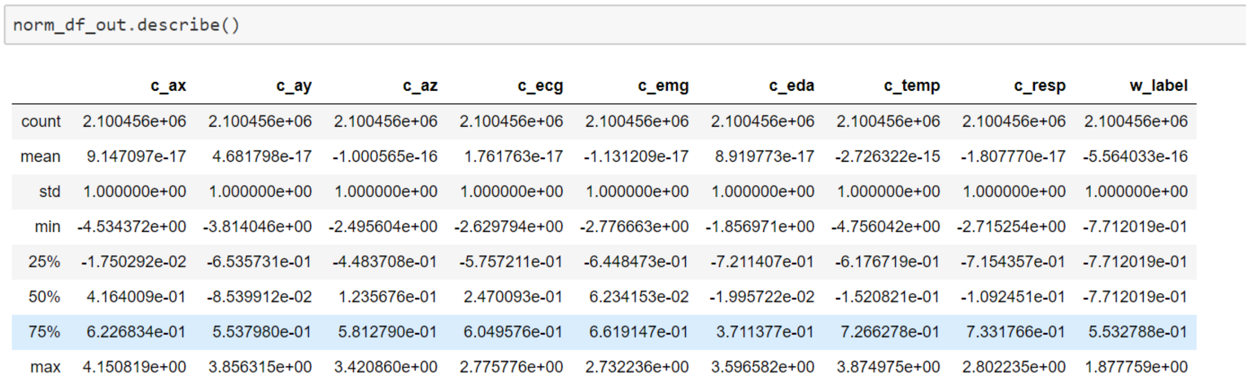
```
norm_df_out.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2100456 entries, 393691 to 4255299
Data columns (total 9 columns):
c_ax       float64
c_ay       float64
c_az       float64
c_ecg      float64
c_emg      float64
c_eda      float64
c_temp     float64
c_resp     float64
w_label    float64
dtypes: float64(9)
memory usage: 160.3 MB
```

Figure-2.4: Important information about total number of rows and columns for cleaned data.

Figure displays important information about total number of rows and columns for cleaned and normalized data developed after removing all outliers and NaN values from complex dictionary structure.

2.5 No Strings found in Dataset



```
norm_df_out.describe()
```

```
              c_ax          c_ay          c_az         c_ecg         c_emg         c_eda        c_temp        c_resp       w_label
count  2.100456e+06  2.100456e+06  2.100456e+06  2.100456e+06  2.100456e+06  2.100456e+06  2.100456e+06  2.100456e+06  2.100456e+06
mean   9.147097e-17  4.681798e-17 -1.000565e-16  1.761763e-17 -1.131209e-17  8.919773e-17 -2.726322e-15 -1.807770e-17 -5.564033e-16
std    1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00  1.000000e+00
min   -4.534372e+00 -3.814046e+00 -2.495604e+00 -2.629794e+00 -2.776663e+00 -1.856971e+00 -4.756042e+00 -2.715254e+00 -7.712019e-01
25%   -1.750292e-02 -6.535731e-01 -4.483708e-01 -5.757211e-01 -6.448473e-01 -7.211407e-01 -6.176719e-01 -7.154357e-01 -7.712019e-01
50%    4.164009e-01 -8.539912e-02  1.235676e-01  2.470093e-01  6.234153e-02 -1.995722e-02 -1.520821e-01 -1.092451e-01 -7.712019e-01
75%    6.226834e-01  5.537980e-01  5.812790e-01  6.049576e-01  6.619147e-01  3.711377e-01  7.266278e-01  7.331766e-01  5.532788e-01
max    4.150819e+00  3.856315e+00  3.420860e+00  2.775776e+00  2.732236e+00  3.596582e+00  3.874975e+00  2.802235e+00  1.877759e+00
```

Figure-2.5: Important statistical information about cleaned data set.

Figure displays important statistical information about minimum, maximum and average deviation for cleaned and normalized data developed after removing all outliers and NaN values from complex dictionary structure. WESAD data does not contain any string values, thus conversion of a string to float is not required. Moreover normalized data now has a confined range and limited variation after removing all outliers.

2.6 Mean, standard deviation, minimum and maximum values reduce after removing outliers.

Table 2.6.1 shows comparison between data with outliers and data without outliers after removing outliers.

Table 2.6.1: Comparison of statistical data for between unclean vs clean data.

Sensor name	Unclean Min	Clean Min	Unclean Max	Clean Max	Unclean Std	Clean Std
c_ax	0.32005	0.231800	2.029800	1.298600	0.175116	0.112467
c_ay	-0.660000	-0.230800	0.539000	-0.014800	0.064173	0.028160
c_az	-1.135400	-0.920800	1.246800	0.550200	0.300236	0.248628
c_ecg	-1.499542	-0.236481	1.499313	0.212860	0.154077	0.083126

c_emg	-0.414963	-0.026779	0.300980	0.022430	0.011652	0.008933
c_eda	0.273214	0.266473	7.576752	2.666855	1.238425	0.437950
c_temp	28.745258	28.014905	34.370392	33.674652	1.281106	0.574642
c_resp	-27.903748	-6.132507	27.378845	6.269836	2.917749	2.247824

Consider table 2.6.1 that shows mean, standard deviation, minimum and maximum values for each sensor is reduced. This is correct because all outliers are removed and spread or range of data is minimized. Since outliers are those values which increase the spread and deviation in data set, thus their removal adds stability to sensor data. This is very important since only relevant data will be used for training models. Consider table 2.6.2 that shows new number of data samples after removal of outliers.

Table 2.6.2: Reduced number of samples in WESAD data set.

Initial data samples with outliers	Final data samples without outliers
4255300	2100456

Table 2.6.2 shows that total number of data samples and over all spread are reduced after removal of outliers, thus training and testing will be done on remaining 2100456 data samples.

<u>2.7 All data samples have gaussian distribution thus LDA or QDA can be used</u>

Consider figure 2.7 which shows histogram of all 8 sensor values.

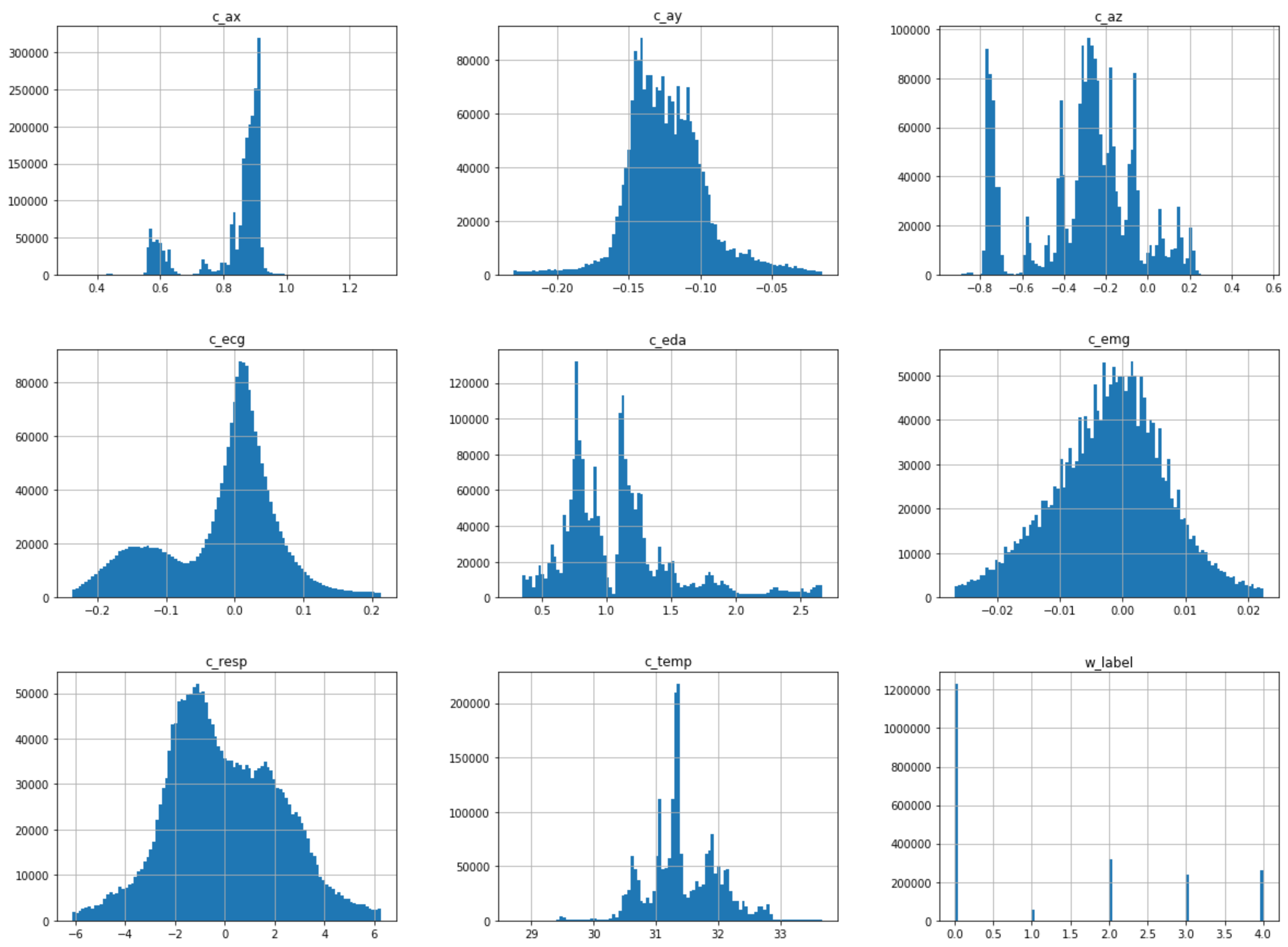


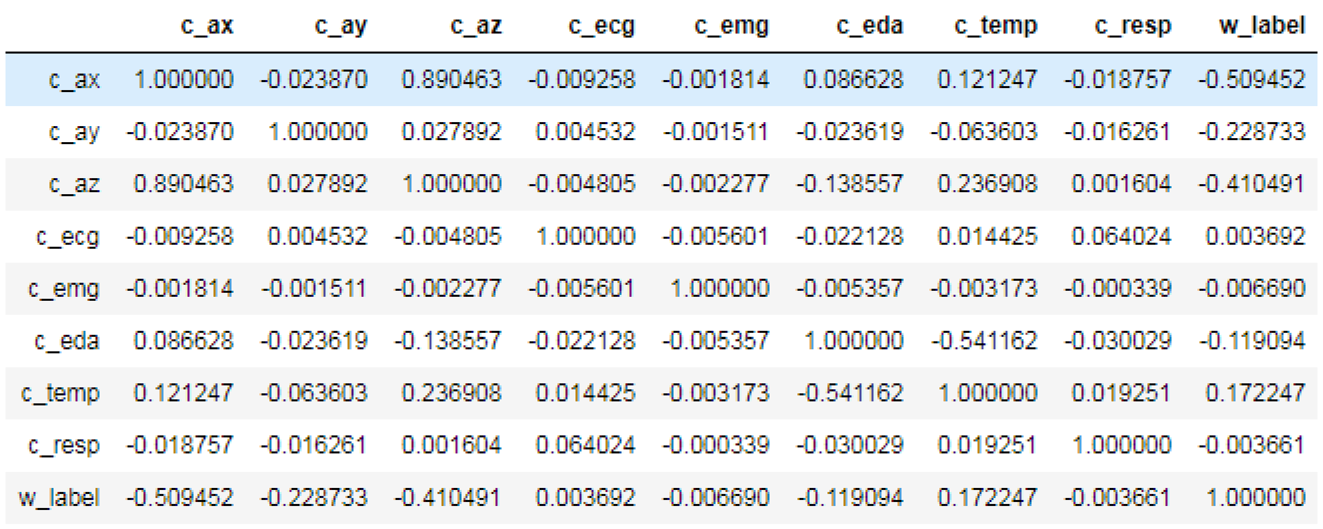
Figure 2.7: Gaussian distribution can be confirmed by histogram shape.

It is evident from the shape of histograms that all 8 sensor data samples have a distinctive Gaussian distribution, since data boundary is bell shaped. This implies that LDA and QDA models can be used while training, since both models primarily require input data to be Gaussian. On contrary Logistic regression can not be used and it will perform poorly.

Also it can be concluded that data is concentrated around the mean, since maximum concentration of histogram plot proves the same.

2.8 Correlation matrix.

Consider figure 2.8 that shows correlation matrix for all sensor values in the data set.



	c_ax	c_ay	c_az	c_ecg	c_emg	c_eda	c_temp	c_resp	w_label
c_ax	1.000000	-0.023870	0.890463	-0.009258	-0.001814	0.086628	0.121247	-0.018757	-0.509452
c_ay	-0.023870	1.000000	0.027892	0.004532	-0.001511	-0.023619	-0.063603	-0.016261	-0.228733
c_az	0.890463	0.027892	1.000000	-0.004805	-0.002277	-0.138557	0.236908	0.001604	-0.410491
c_ecg	-0.009258	0.004532	-0.004805	1.000000	-0.005601	-0.022128	0.014425	0.064024	0.003692
c_emg	-0.001814	-0.001511	-0.002277	-0.005601	1.000000	-0.005357	-0.003173	-0.000339	-0.006690
c_eda	0.086628	-0.023619	-0.138557	-0.022128	-0.005357	1.000000	-0.541162	-0.030029	-0.119094
c_temp	0.121247	-0.063603	0.236908	0.014425	-0.003173	-0.541162	1.000000	0.019251	0.172247
c_resp	-0.018757	-0.016261	0.001604	0.064024	-0.000339	-0.030029	0.019251	1.000000	-0.003661
w_label	-0.509452	-0.228733	-0.410491	0.003692	-0.006690	-0.119094	0.172247	-0.003661	1.000000

Figure 2.8: Correlation matrix for all 8 sensors.

It is evident from figure 2.8 that only accelerometer x axis and z axis are highly correlated, on the contrary no other sensor value have high correlation. Although P-value should be considered before concluding usage of a weight associated with certain parameter, but data set is highly uncorrelated. Thus sensor values are unique and independent of each other.

2.9 Feature Scaling by Normalizing Cleaned Dataset

Mean normalization method is used to scale down entire data set, so that models can converge faster and less computational power is required while processing. Since large float values and uneven numbers can result in poor models.

D) Related Work

Philip Schmidt’s paper titled “Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection” was used for understanding data set and baseline algorithm.

Problem addressed:

This paper solved a classification problem of differentiating between three distinct states like baseline mood, stress state and amusement state. Further a binary classification problem was solved for differentiating between stress vs non-stress state. Lastly a comparison was provided between sensor modalities of wrist device and chest device.

Usage of Data set:

Authors separated data for wrist device and chest device and separately calculated accuracy matrix for each device. It should be noted that data from all subjects were combined together and a sliding window of 0.25 seconds was created that had 133000 windows. Out of which 53% were of baseline class, stress class had 30% windows and 17% belonged to amusement condition.

Cleaning of data set:

Authors used 5Hz low pass filters, bandpass filters and min max based normalization methods for cleaning the data set.

Feature Extraction:

Various signal processing based algorithms were used to extract useful features. Peak detection algorithms, standard deviation, dynamic range detection, variance and slope detection algorithms were used to extract important features.

Misclassification Matrix:

As data set is highly unbalanced thus authors avoided using accuracy matrix rather they used only F1 score matrix to conclude their results.

<u>Results obtained:</u>

Authors used 5 different algorithms to compute f1 scores namely Decision Tree, Random Forest, AdaBoost Decision Tree, LDA, kNN. For classifying baseline state, stress state and, amusement state score of 80% was achieved by AdaBoost Decision Tree.

For classifying stress state and non-stress state score of 93% was achieved by AdaBoost Decision Tree.

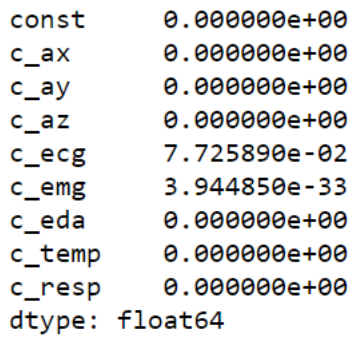
Furthermore it was presented that chest device gave more stable and accurate score thus chest base device resulted in best classification results and by adding wrist device data no further improvements were achieved.

<u>Conclusion presented:</u>

It was concluded that AdaBoost Decision Tree outperformed all other classifiers with highest score. Also wrist device did not help in improving overall score. Furthermore data from wrist device could be ignored while preparing models, which will reduce dimensions of multi modal data set.

<u>E) Feature Extraction</u>

Consider figure E-1.0 which shows all p-values that are very close to 0.



```
const     0.000000e+00
c_ax      0.000000e+00
c_ay      0.000000e+00
c_az      0.000000e+00
c_ecg     7.725890e-02
c_emg     3.944850e-33
c_eda     0.000000e+00
c_temp    0.000000e+00
c_resp    0.000000e+00
dtype: float64
```

Figure E-1.0: P-values for all coefficients.

It is evident from the figure that p-values of all coefficients is almos, which means that all coefficients are very important. It should be noted that smaller p values means it is relevant feature.

<u>E-1. Filter Method:</u>

A simple correlation matrix using SNS heat map can be used for manually eliminating features. Consider figure E-1.1 that shows sns heat map with correlation values.

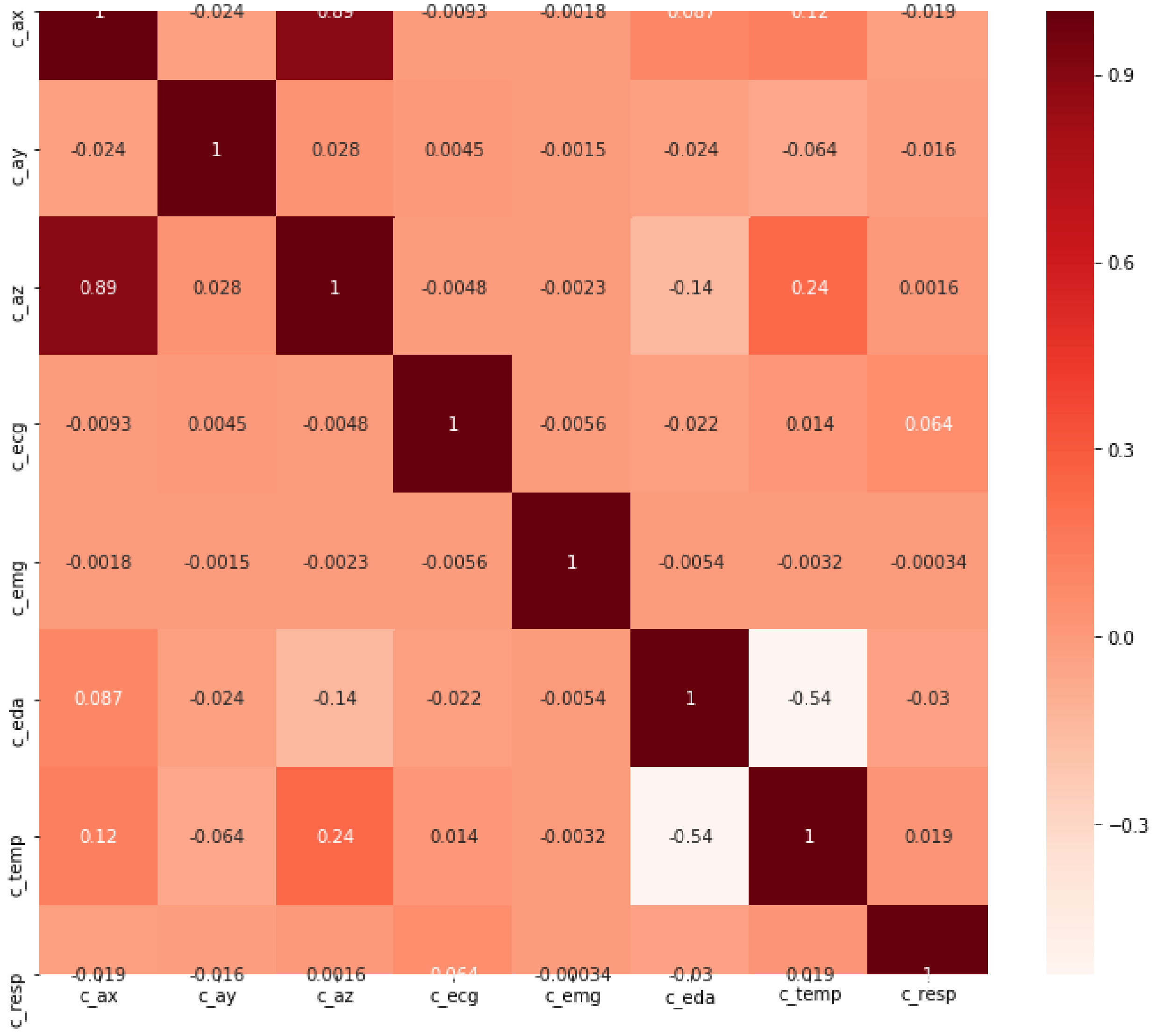
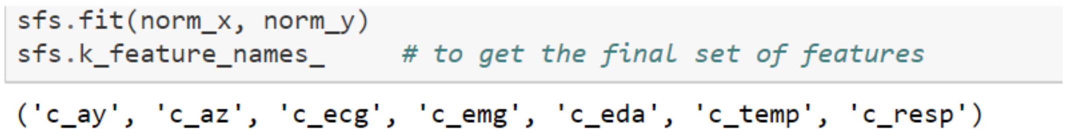


Figure E-1.1: Showing sns heat map with correlation values.

It is evident that data is highly uncorrelated thus all parameters must be used.

<u>E-2. Wrapper Method (Forward and Backward Selection):</u>

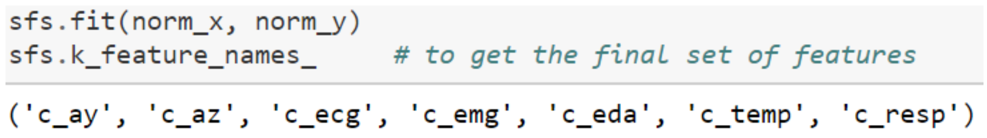
Wrapper method consists of forward selection and backward selection, consider figure E-2.1 and figure E-2.2 that shows forward selection vales and backward selection values.



```
sfs.fit(norm_x, norm_y)
sfs.k_feature_names_     # to get the final set of features
```

```
('c_ay', 'c_az', 'c_ecg', 'c_emg', 'c_eda', 'c_temp', 'c_resp')
```

Figure E-2.1 Forward selection values.



```
sfs.fit(norm_x, norm_y)
sfs.k_feature_names_     # to get the final set of features
```

```
('c_ay', 'c_az', 'c_ecg', 'c_emg', 'c_eda', 'c_temp', 'c_resp')
```

Figure E-2.1 Backward selection values.

It is very evident from the results that all 7 features namely 'c_ay', 'c_az', 'c_ecg', 'c_emg', 'c_eda', 'c_temp' and 'c_resp' are very important. Thus none of the features should be eliminated.

<u>E-3. Embedded Method (L1 and L2 Regularization):</u>

L1 and L2 regularization are regularly used with all classifiers below so that coefficients are either removed or optimized during iterations.

It should be noted that while training classifiers below L1 regularization will give sparse solutions as it eliminates coefficients. On the contrary L2 regularization will give smooth solution.

F) Model Development

Model-1.1: Logistic Regression as Baseline Model

1) Development

Logistic regression is a classification model that follows supervised learning to predict probability of labels or desired variables. In WESAD data set milti class logistic model is constructed with various parameters as shown in Table M-1.1.1 below. It should be clearly noted that logistic regression assumes that data set is linear and **not Gaussian** which could be a major drawback for this model.

Table M-1.1.1: Logistic regression important parameters.

Model-1.1: Logistic Regression Important Parameters	
Important Parameters	Value
Penalty	L2
Solver	Newton_cg
Multi_class	Multinomial
Max_iter	100

L2 penalty is used because data is highly uncorrelated hence impact of each weight is important and L2 regularization does not eliminate any coefficient.

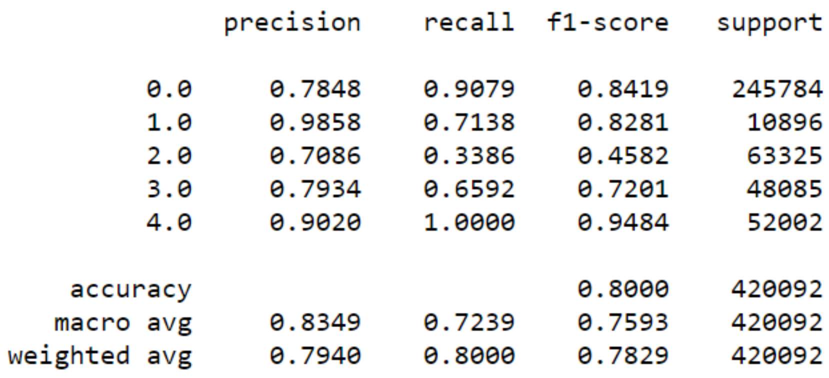
Newton-cg solver is used because it supports L2 regularization and optimization of the problem. It is most commonly used in multiclass classification problem. Whereas Saga solve supports only L1 reugularization.

Multi_class is an important parameter in Logistic regression which determines that convergence of either multiclass or binary problem is required. In this data set 4 distinct classes area available thus multinomial mode is selected for better convergence. This parameter is set to OvR for fitting binary problem on each label.

Stopping criteria max_iter is set to default value of 100, this helps in avoiding the machine to enter very long computational periods and exits the loop after desired convergence value.

2) Results and Discussions:

Classification matrix of recall, precision, accuracy and f1-score are used to determine performance of this model as shown in figure M-1.1.2 below.



```
              precision    recall  f1-score   support

         0.0     0.7848    0.9079    0.8419    245784
         1.0     0.9858    0.7138    0.8281     10896
         2.0     0.7086    0.3386    0.4582     63325
         3.0     0.7934    0.6592    0.7201     48085
         4.0     0.9020    1.0000    0.9484     52002

    accuracy                         0.8000    420092
   macro avg     0.8349    0.7239    0.7593    420092
weighted avg     0.7940    0.8000    0.7829    420092
```

Figure M-1.1.2: Recall, precision, accuracy and f1-score with macro average and weighted average.

It should be denoted that this data set in unbalanced thus accuracy score does not contribute here, recall matrix can be considered as best performance measure for this data set.

Precision Matrix: Precision score is extremely low for class 2 and overall score is around 83%.

Recall Matrix: Extremely poor performance is presented by this model as recall score is around 72%, which means how many correct predictions were made out of actual samples present in that class.

Accuracy Score: Accuracy score for unbalanced data set is deceptive in this case, it is around 80%.

F1-score: This gives harmonic mean of precision and recall which is very low around 75%.

As logistic regression assumes that data has to be non-gaussian which is not the case with this data set thus, it performs poorly. Also logistic regression is simplest form of classification model, therefore all other models should perform better than this model. Any further model performing poorly than Linear regression base model will be rejected immediately.

<u>Model-2.1: Linear Discriminant Analysis (LDA)</u>

LDA is a supervised method used for solving classification problems. LDA has two basic assumptions about any given data set.

1) Data set distribution should be Gaussian

2) Each attribute must have same variance i.e. each value should vary around the same mean with similar average.

1) Development

Important parameters for designing a LDA model are given below in table M-2.1.1.1, which will be tweaked ahead to improve model performance.

Linear Discriminant Analysis (LDA) Important Parameters	
Parameter	Value
Solver	SVD
Shrinkage	Automatic
N_components	3

SVD solver is used because “Singular value decomposition” (SVD) does not compute on covariance matrix thus gives best results from data set with many number features.

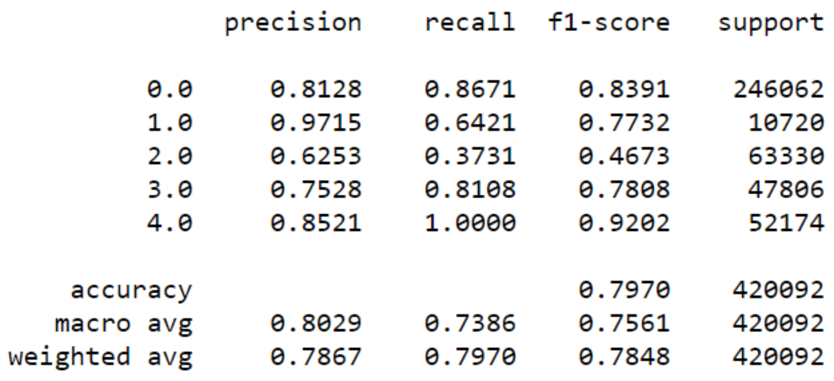
Shrinkage is set to auto because hard coding it to a float value will increase computational time also, shrinkage works only with either LSQR or Eigen solvers.

N_components should always be set less than total number of classes minus one for dimensionality reduction.

Dimensionality reduction is achieved by selecting appropriate solver, shrinkage and n_components value as mentioned above. Linear Discriminant Analysis in simple terms creates a third axis by utilizing information given by two or more different axis (features) by minimizing variance and maximizing overall distance between means.

2) Results and Discussions:

Classification matrix of recall, precision, accuracy and F1-score are used to determine performance of this model as shown in figure M-2.1.2 below.



```
              precision    recall  f1-score   support

         0.0     0.8128    0.8671    0.8391    246062
         1.0     0.9715    0.6421    0.7732     10720
         2.0     0.6253    0.3731    0.4673     63330
         3.0     0.7528    0.8108    0.7808     47806
         4.0     0.8521    1.0000    0.9202     52174

    accuracy                         0.7970    420092
   macro avg     0.8029    0.7386    0.7561    420092
weighted avg     0.7867    0.7970    0.7848    420092
```

Figure M-2.1.2: Recall, precision, accuracy and f1-score with macro average and weighted average.

It should be denoted that this data set in unbalanced thus accuracy score does not contribute here, recall matrix can be considered as best performance measure for this data set.

Precision Matrix: Precision score is low for class 2 and overall score is around 80%, which is better than Linear regression model.

Recall Matrix: Extremely poor performance is presented by this model as recall score is around 73%, which means only 73% correct predictions were made out of actual samples present in that class. This score is better than Linear Regression model.

Accuracy Score: Accuracy score for unbalanced data set is deceptive in this case, it is around 79%.

F1-score: This gives harmonic mean of precision and recall which is very low around 75%.

As LDA assumes that data set is Gaussian in nature, which is correct assumption with this data set thus performance slightly improves. But overall it can be concluded that LDA is not very impressive because each attribute does not have same variance throughout which is 2nd assumption of LDA model.

Model-3.1: Quadratic Discriminant Analysis (QDA)

QDA is a supervised method used for solving classification problems. QDA has three basic characteristics and gives best performance when all three cases are matched.

1) Data set distribution should be Gaussian

2) Each attribute must have different variance.

3) Data set is uncorrelated and quadratic in nature.

4) Works better for large data set.

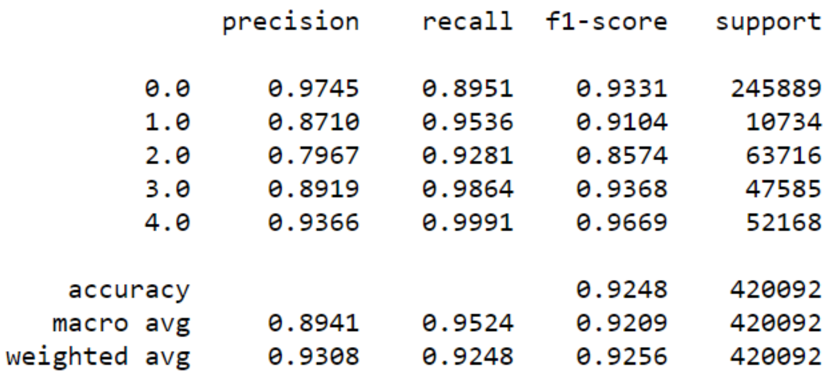
There is only one tuning parameter in this case which is reg_param, it should be between 0 and 1. By default value is set to 0 and this value will be varied in sections below. This parameter performs regularization on per class covariance.

1) Development

QDA has one hyper parameter which is called reg_param. This parameter ranges from 0 to 1 and helps in regularizing the covariance.

2) Results and Discussions:

Classification matrix of recall, precision, accuracy and F1-score are used to determine performance of this model as shown in figure M-3.1.1 below.



```
              precision    recall  f1-score   support

         0.0     0.9745    0.8951    0.9331    245889
         1.0     0.8710    0.9536    0.9104     10734
         2.0     0.7967    0.9281    0.8574     63716
         3.0     0.8919    0.9864    0.9368     47585
         4.0     0.9366    0.9991    0.9669     52168

    accuracy                         0.9248    420092
   macro avg     0.8941    0.9524    0.9209    420092
weighted avg     0.9308    0.9248    0.9256    420092
```

Figure M-3.1.1: Recall, precision, accuracy and f1-score with macro average and weighted average.

It should be denoted that this data set in unbalanced thus accuracy score does not contribute here, recall matrix can be considered as best performance measure for this data set.

Precision Matrix: Precision score improves for class 2 and similarly overall precision score is around 89%, which is better than Linear regression model and LDA.

Recall Matrix: Excellent performance score is presented by this model as recall score is around 95%, which means only 95% correct predictions were made out of actual samples present in that class. This score is better than Linear Regression model and LDA.

Accuracy Score: Accuracy score for unbalanced data set is deceptive in this case, it is around 82%.

F1-score: This gives harmonic mean of precision and recall which is high around 92%.

QDA works better than LDA because it is suitable for complex and large data set as QDA has low bias and high variance. Also QDA works better for highly uncorrelated data set and assumes that each attribute must have different variance. Thus QDA outperforms LDA.

Model-4.1: K Nearest Neighbors Method (KNN)

KNN is a supervised and non-parametric method used for solving classification problems and regression problems. KNN algorithm will classify data based on the proximity of K number of nearest neighbors. Thus if incoming data is closely surrounded by more number of data points belonging to class-1 then incoming data is predicted to be of class-1 itself. KNN has one basic hyper parameter called K that is number of nearest neighbors. Other important parameters are given in table M-4.1.1 below.

KNN has 2 basic properties as discussed in following points:

1) KNN assumes that data set is very complex.

2) Very small values of k results in over fitting and large k values give linear behavior.

Table M-4.1.1: Displaying important parameters for KNN algorithm

KNN Important Parameters	
Parameter	Value
K	5
Algorithm for computing distance	Auto
Weights	Uniform

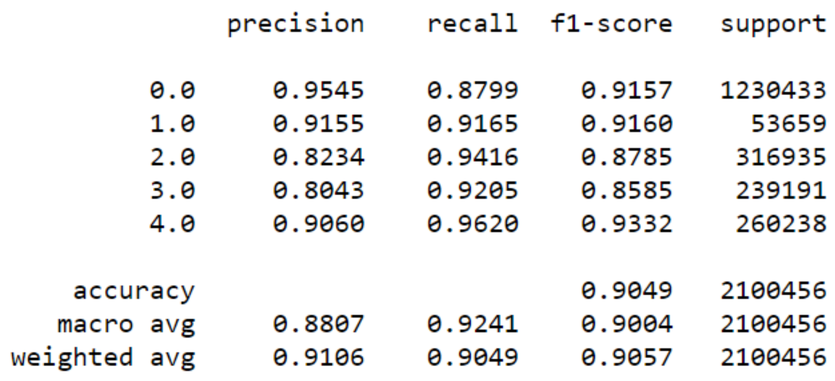
K stands for total number of neighbors whose proximity will be considered by incoming data point. If K is set to 1 then over fitting occurs, since decision region becomes very complex and highly non linear. Thus default value is set to 5, which can be increased up to 10. If k is set to a very large number then model becomes close to have linear behavior.

Weights are set to uniform so that all inputs are weighed equally. Though they can weighed according to their distance too.

Algorithm for computing distance is set to auto so that best algorithm is picked

2) Results and Discussions:

Classification matrix of recall, precision, accuracy and F1-score are used to determine performance of this model as shown in figure M-4.1.2 below.



```
              precision    recall  f1-score   support

         0.0     0.9545    0.8799    0.9157   1230433
         1.0     0.9155    0.9165    0.9160     53659
         2.0     0.8234    0.9416    0.8785    316935
         3.0     0.8043    0.9205    0.8585    239191
         4.0     0.9060    0.9620    0.9332    260238

    accuracy                         0.9049   2100456
   macro avg     0.8807    0.9241    0.9004   2100456
weighted avg     0.9106    0.9049    0.9057   2100456
```

Figure M-4.1.2: Recall, precision, accuracy and f1-score with macro average and weighted average.

It should be denoted that this data set in unbalanced thus accuracy score does not contribute here, recall matrix can be considered as best performance measure for this data set.

Precision Matrix: Precision score improves for class 2 and similarly overall precision score is around 88%, which is better than Linear regression model and LDA.

Recall Matrix: Excellent performance score is presented by this model as recall score is around 92%, which means only 92% correct predictions were made out of actual samples present in that class. This score is better than Linear Regression model, LDA.

Accuracy Score: Accuracy score for unbalanced data set is deceptive in this case, it is around 90%.

F1-score: This gives harmonic mean of precision and recall which is high around 90%.

KNN works better than LDA because it is suitable for more complex and large data set as KNN model is highly flexible and has more variance and less bias as compared to LDA.

<u>Model-5.1: Simple Decision Trees</u>

Decision Tree is a non parametric and supervised learning method which can be used for both regression and classification problem. In this data set decision tree is used to solve classification problem. This algorithm works like a human brain, where predictions are made according to simple greater than or less than threshold based decisions. Important parameters associated with this algorithm are given in table M-5.1.1 below.

Table M-5.1.1: Displaying important parameters for Simple Decision Tree.

Simple Decision Tree Important Parameters	
Parameter	Value
Criterion	Entropy
Max depth	5
min_samples_leaf	5

Criterion can be set to gini index or entropy for either measuring gini index while making splits or measuring amount of impurity via entropy. Gini impurity measures actual probability of divergence between target attribute values. And smallest gini index is preferred. and splits a node such that it gives the least amount of impurity. Similarly impurity value for same class should be zero and maximum between two or more different classes.

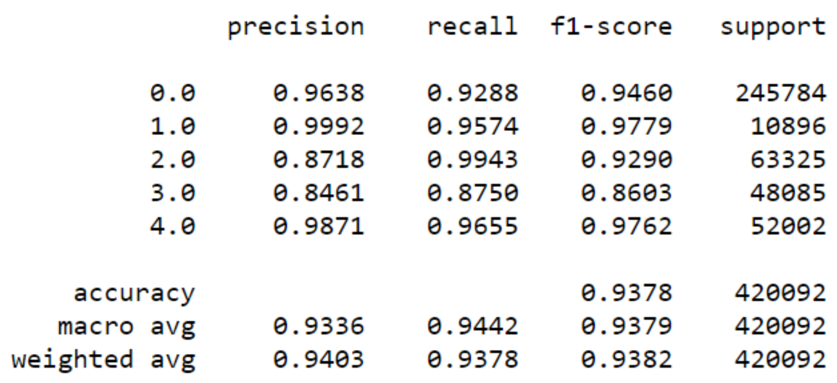
Max_depth is a important hyper parameter in controlling both under fitting and over fitting of this algorithm. It controls the total splitting depth of a tree and stops splitting after reaching certain depth. If this number is very small then over fitting will happen and if this number is very large then under fitting will happen.

Min_samples_leaf is used to control over fitting and under fitting of decision trees. It controls performance of these trees be ensuring minimum number of elements in each leaf. If min_samples_leaf is selected to be a very small value then over fitting will happen. On contrary if value is very large then under fitting will happen. Thus average number should be selected for this parameter.

Algorithm for computing distance is set to auto so that best algorithm is picked

2) Results and Discussions:

Classification matrix of recall, precision, accuracy and F1-score are used to determine performance of this model as shown in figure M-5.1.2 below.



```
              precision    recall  f1-score   support

         0.0     0.9638    0.9288    0.9460    245784
         1.0     0.9992    0.9574    0.9779     10896
         2.0     0.8718    0.9943    0.9290     63325
         3.0     0.8461    0.8750    0.8603     48085
         4.0     0.9871    0.9655    0.9762     52002

    accuracy                         0.9378    420092
   macro avg     0.9336    0.9442    0.9379    420092
weighted avg     0.9403    0.9378    0.9382    420092
```

Figure M-5.1.2: Recall, precision, accuracy and f1-score with macro average and weighted average.

It should be denoted that this data set in unbalanced thus accuracy score does not contribute here, recall matrix can be considered as best performance measure for this data set.

Precision Matrix: Precision score improves for class 2 and similarly overall precision score is around 93%, which is better than Linear regression model, LDA, KNN and QDA.

Recall Matrix: Excellent performance score is presented by this model as recall score is around 94%, which means only 94% correct predictions were made out of actual samples present in that class. This score is better than Linear Regression model, LDA, KNN and QDA.

Accuracy Score: Accuracy score for unbalanced data set is deceptive in this case, it is around 93%

F1-score: This gives harmonic mean of precision and recall which is high around 93%.

Decision trees perform better than KNN and QDA because it is suitable for more complex and large data set as KNN model is highly flexible and has more variance and less bias as compared to QDA. Consider figure M-5.1.3 that shows most important variables in splitting the tree.

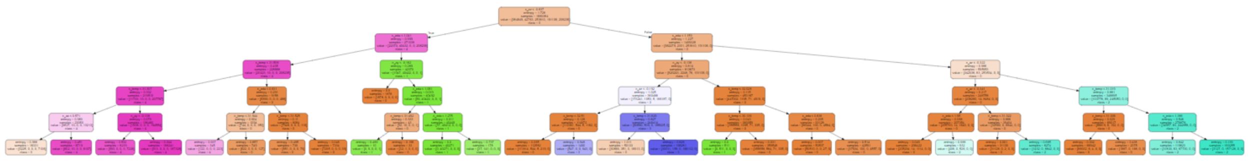


Figure M-5.1.3: Decision tree visualization.

The figure M-5.1.3 shows acceleration in z axis is most important variable, followed by eda sensor data. Subject’s body temperature and acceleration along y axis are third splitting points. It should be mentioned that ECG and EMG data are completely ignored while making final decisions.

<u>Conclusion:</u>

It should be noted that simple decision tree based method gives better precision and recall score than KNN, LDA and QDA classifiers. Decision tree is interpretability is simple and easy to follow, where it eliminates ECG, RESP and EMG sensor data while making final classification decisions.

<u>Model-6.1: XGBoost Decision Trees with gradient boosting</u>

XGBoost algorithm uses gradient boosting based decision tree framework. XGBoost is very popular machine learning algorithm that is recommended for tabular data. when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now. Gradient boosting is a special type of boosting method where errors are minimized using well known gradient descent algorithm that tries to converge towards lowest missclassification error. Important parameters associated with this algorithm are given in table M-6.1.1 below.

Table M-6.1.1: Displaying important parameters for XGBoost algorithm.

XGBoost Important Parameters	
Parameter	Value
Objective	Multi:Softmax
Colsample_bytree	0.3
Learning_rate	0.1
Max_depth	10
Alpha	10
N_estimators	50
Gamma	10
Lambda	L2 regularization

Objective: It fixes the desired loss function, for multiclass classification problem softmax is preferred over simple binary logistic optimizer.

Colsample_bytree: This helps in controlling total percentage of essential features to be used in each tree. It should be kept small, because higher values result in over fitting.

Learning_rate: Has to be within 0 to 1 for shrinking the step size. It helps in preventing over fitting.

Max_depth: Controls the maximum depth allowed for each tree can grow to during boosting. Very small value will lead to under fitting and high values result in over fitting.

Alpha: This helps in applying L1 regularization on weights associated with each leaf. It should be noted that L1 regularization will give sparse solutions.

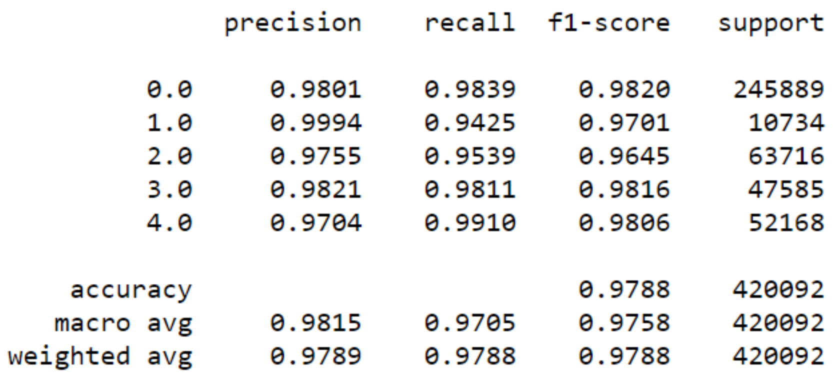
Lambda: This helps in applying L2 regularization on weights associated with each leaf. It should be noted that L2 regularization will give smooth solutions.

N_estimators: Total number of trees.

Gamma: Higher gamma values result in fewer number of splits. It control whether node will split or not after encountering expected reduction in loss.

<u>2) Results and Discussions:</u>

Classification matrix of recall, precision, accuracy and F1-score are used to determine performance of this model as shown in figure M-6.1.2 below.



```
              precision    recall  f1-score   support

         0.0     0.9801    0.9839    0.9820    245889
         1.0     0.9994    0.9425    0.9701     10734
         2.0     0.9755    0.9539    0.9645     63716
         3.0     0.9821    0.9811    0.9816     47585
         4.0     0.9704    0.9910    0.9806     52168

    accuracy                         0.9788    420092
   macro avg     0.9815    0.9705    0.9758    420092
weighted avg     0.9789    0.9788    0.9788    420092
```

Figure M-6.1.2: Recall, precision, accuracy and f1-score with macro average and weighted average.

It should be denoted that this data set in unbalanced thus accuracy score does not contribute here, recall matrix can be considered as best performance measure for this data set.

Precision Matrix: Precision score is excellent for class 2 and similarly overall precision score is around 98%, which is better than Linear regression model, LDA, KNN, Simple Decision Tree and QDA.

Recall Matrix: Excellent performance score is presented by this model as recall score is around 97%, which means only 97% correct predictions were made out of actual samples present in that class. This score is better than Linear Regression model, LDA, KNN, Simple Decision Tree and QDA.

Accuracy Score: Accuracy score for unbalanced data set is deceptive in this case though it is 97%.

F1-score: This gives harmonic mean of precision and recall which is high around 97%.

Conclusion:

XGBoost trees perform better than Decision trees, KNN, LDA, Logistic regression and QDA because it follows gradient boosting framework which is based on minimizing miss classification error with help of gradient descent algorithm.

Model-7.1: Support Vector Machine (SVM)

SVM is supervised learning method specifically used for handling complex data. SVM gives high accuracy and small missclassification error due to usage of radial kernel. Important parameters associated with this algorithm are given in table M-7.1.1 below.

Table M-7.1.1: Displaying important parameters for SVM algorithm.

SVM Important Parameters	
Parameter	Value
Kernel	RBF
Random_state	0
Gamma	10
C	100
Shrinkage	True
Tol	1
Cache_size	1
Max_iter	20000

Kernel: Radial bias function kernel is used in this case rather than linear because data set is complex with nonlinear distribution.

Random_state: Scikit learn uses random number generator for shuffling data in order to estimate probability. This randomness is nullified by hard coding random_state to 0, so that randomness in final result can be avoided.

Gamma: It is a tuning parameter associated with RBF kernel for controlling total spread of the kernel. If gamma is high then decision boundary is very curvy and over fitting occurs.

C: It is a parameter of SVC algorithm, also know as penalty term. Small value of this term does not put any penalty on data points that are missclassified. But large value of this parameter add penalty on missclassification. Large value of C means model is highly intolerant to missclassification.

Shrinkage: SVM removes unused number of kernels while iterating on large data set, that helps in converging faster.

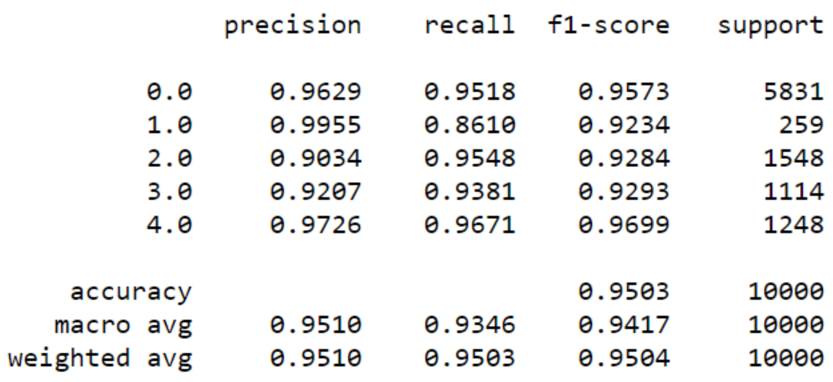
Tol: Helps in reducing randomness in predicted output for same input during different iterations.

Cache_size: This parameter reserves certain amount of processing space required while processing large data set with radial kernel.

Max_iter: Maximum number of allowed iterations. Small number of iterations can under fit, due to early stopping. Whereas large value of this variable will take days to converge.

2) Results and Discussions:

Classification matrix of recall, precision, accuracy and F1-score are used to determine performance of this model as shown in figure M-7.1.2 below.



```
              precision    recall  f1-score   support

         0.0     0.9629    0.9518    0.9573      5831
         1.0     0.9955    0.8610    0.9234       259
         2.0     0.9034    0.9548    0.9284      1548
         3.0     0.9207    0.9381    0.9293      1114
         4.0     0.9726    0.9671    0.9699      1248

    accuracy                         0.9503     10000
   macro avg     0.9510    0.9346    0.9417     10000
weighted avg     0.9510    0.9503    0.9504     10000
```

Figure M-7.1.2: Recall, precision, accuracy and f1-score with macro average and weighted average.

It should be denoted that this data set in unbalanced thus accuracy score does not contribute here, recall matrix can be considered as best performance measure for this data set.

Precision Matrix: Precision score is excellent for class 2 and similarly overall precision score is around 95%, which is better than Linear regression model, LDA, KNN, Simple Decision Tree and QDA. But SVM could not outperform XGBoost.

Recall Matrix: Excellent performance score is presented by this model as recall score is around 93%, which means only 93% correct predictions were made out of actual samples present in that class. This score is better than Linear Regression model, LDA, KNN, Simple Decision Tree and QDA. But SVM could not outperform XGBoost.

Accuracy Score: Accuracy score for unbalanced data set is deceptive in this case though it is 95%.

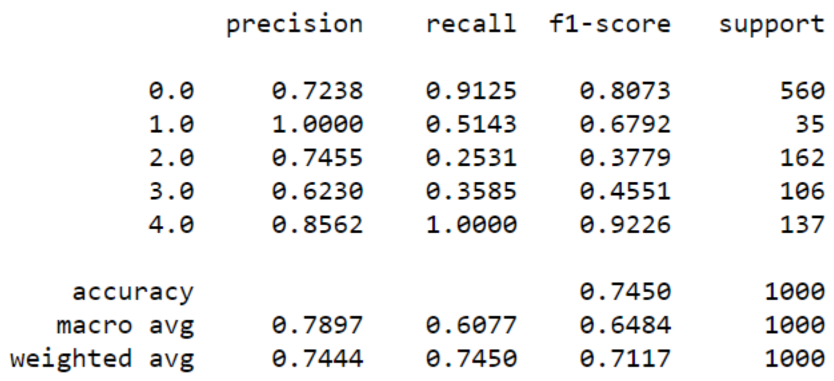
F1-score: This gives harmonic mean of precision and recall which is high around 94%. But SVM could not outperform XGBoost.

Conclusion:

SVM takes more than 36 hours to completely converge and blocks all other associated on chip resources. Moreover SVM fails to outperform XGBoost. But performs better than Simple Decision trees, KNN, LDA, Logistic regression and QDA because SVM uses radial bias kernel function for minimizing misclassification error.

Moreover these results are calculated using lesser amount of data so that model could converge, even then it could not outperform XGBoost although multiple iterations were tested. Thus from hereafter SVM will be ignored.

Similarly SVC performs poorer than logistic regression model because it can only work for linearly separable data and fails to perform better when data in highly non linear. Consider figure M-7.1.3 that shows poor results.



```
              precision    recall  f1-score   support

         0.0     0.7238    0.9125    0.8073       560
         1.0     1.0000    0.5143    0.6792        35
         2.0     0.7455    0.2531    0.3779       162
         3.0     0.6230    0.3585    0.4551       106
         4.0     0.8562    1.0000    0.9226       137

    accuracy                         0.7450      1000
   macro avg     0.7897    0.6077    0.6484      1000
weighted avg     0.7444    0.7450    0.7117      1000
```

Figure-M-7.1.3: SVC results for lesser amount of data.

Precision, Recall and F1 score are deteriorated for SVC when compared to logistic regression, because data in WESAD data set is highly non linear and can not be separated linearly. Thus from hereon SVC model will not fine tuned.

G) Fine-tune your models & Feature Set

Method-1: Changing and fine tuning important parameters of all models.

Various models will be fine tuned and tested multiple times in this section and results will be presented in this section below in terms of misclassification error matrix of precision, recall, f1 score and accuracy.

1) Model-1.2: Logistic regression with L2 regularization:

L2 regularization is applied in logistic regression model and figure M-1.2.1 shows results obtained.

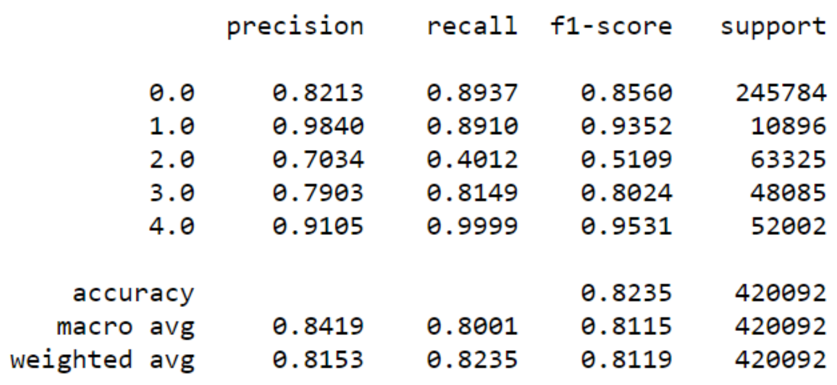
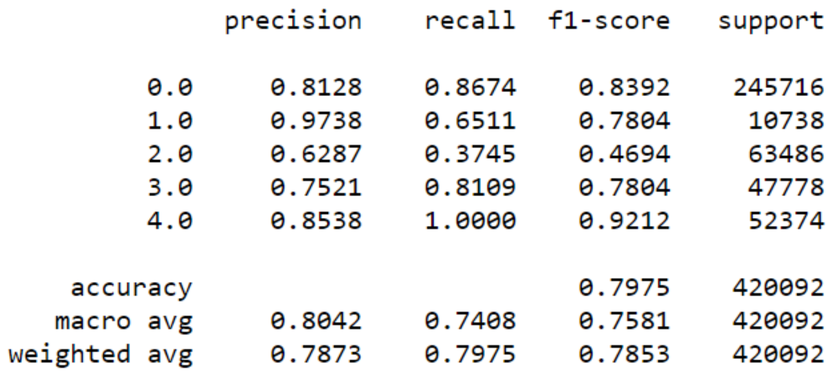

```
              precision    recall  f1-score   support

         0.0     0.8213    0.8937    0.8560    245784
         1.0     0.9840    0.8910    0.9352     10896
         2.0     0.7034    0.4012    0.5109     63325
         3.0     0.7903    0.8149    0.8024     48085
         4.0     0.9105    0.9999    0.9531     52002

    accuracy                         0.8235    420092
   macro avg     0.8419    0.8001    0.8115    420092
weighted avg     0.8153    0.8235    0.8119    420092
```

Figure M-1.2.1: Logistic regression with L2 regularization.

It is evident that Precision matrix decreases and recall matrix increases since coefficients are minimized to very small values. But overall F1 score is increased to 81%. This means L2 regularization successfully minimized unimportant coefficients to provide optimized model.

2) Model-2.2: LDA with Normalization and Cross Validation:

LDA is iterated again this time with normalized input data and misclassification matrix is achieved by doing K-fold cross validation where K is set to 5, results obtained can be seen in figure M-2.2.1 below.



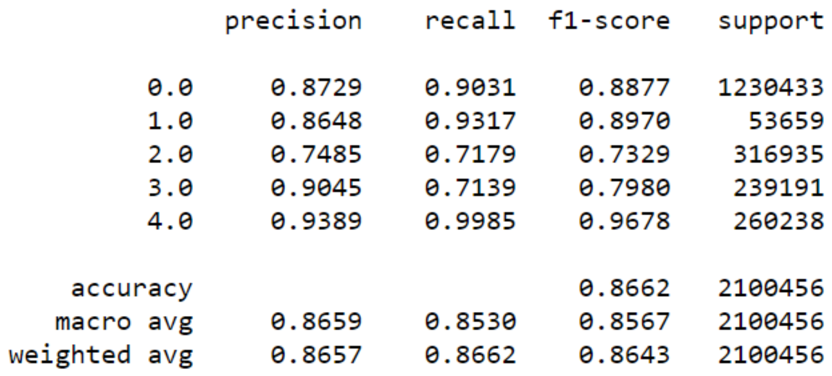
```
              precision    recall  f1-score   support

         0.0     0.8128    0.8674    0.8392    245716
         1.0     0.9738    0.6511    0.7804     10738
         2.0     0.6287    0.3745    0.4694     63486
         3.0     0.7521    0.8109    0.7804     47778
         4.0     0.8538    1.0000    0.9212     52374

    accuracy                         0.7975    420092
   macro avg     0.8042    0.7408    0.7581    420092
weighted avg     0.7873    0.7975    0.7853    420092
```

Figure-M-2.2.1: New misclassification matrix obtained after performing LDA with CV and Normalization.

It is evident that there is no significant improvement in precision, recall and f1 score matrix because LDA works best with linearly separable data. Though normalized data helps in faster computation.

3) Model-3.2: QDA with Normalization and Cross Validation=5:

QDA is iterated again with normalized input data and misclassification matrix is measured using K fold cross validation, where k is set to 5, results obtained are shown in figure M-3.2.1 below.



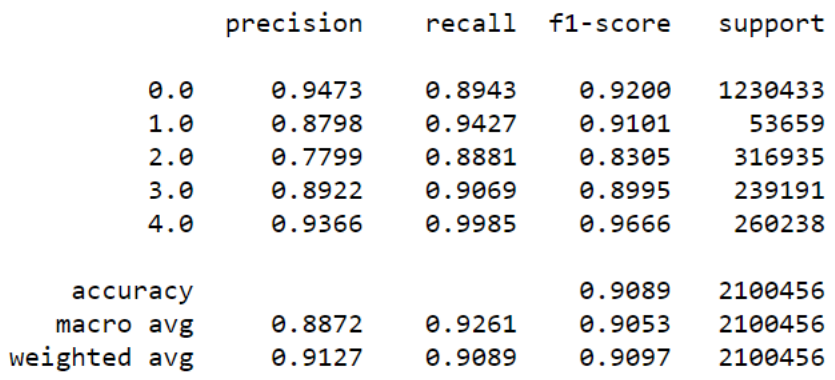
```
              precision    recall  f1-score   support

         0.0     0.8729    0.9031    0.8877   1230433
         1.0     0.8648    0.9317    0.8970     53659
         2.0     0.7485    0.7179    0.7329    316935
         3.0     0.9045    0.7139    0.7980    239191
         4.0     0.9389    0.9985    0.9678    260238

    accuracy                         0.8662   2100456
   macro avg     0.8659    0.8530    0.8567   2100456
weighted avg     0.8657    0.8662    0.8643   2100456
```

Figure-M-3.2.1: Misclassification matrix for QDA with normalization and CV = 5.

It is evident that with respect to base model without cross validation the misclassification error increases for all matrices, but this result is more trust worthy since error is averaged out.

4) Model-3.3: QDA with Normalization and Cross Validation=10:

QDA is iterated again with normalized input data and misclassification matrix is measured using K fold cross validation, where k is set to 10, results obtained are shown in figure M-3.3.1 below.



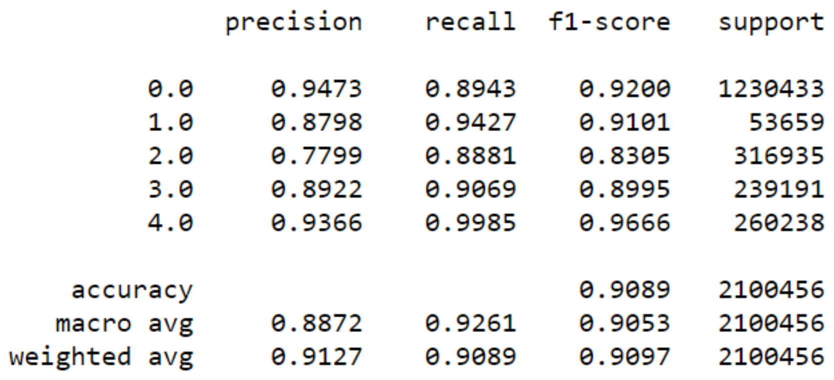
```
              precision    recall  f1-score   support

         0.0     0.9473    0.8943    0.9200   1230433
         1.0     0.8798    0.9427    0.9101     53659
         2.0     0.7799    0.8881    0.8305    316935
         3.0     0.8922    0.9069    0.8995    239191
         4.0     0.9366    0.9985    0.9666    260238

    accuracy                         0.9089   2100456
   macro avg     0.8872    0.9261    0.9053   2100456
weighted avg     0.9127    0.9089    0.9097   2100456
```

Figure-M-3.3.1: Misclassification matrix for QDA with normalization and CV = 10.

It is evident that with respect to QDA with CV=5 this model with CV=10 performs better as all misclassification matrices improve drastically.

5) Model-3.4: QDA with Normalization and Cross Validation=20:

QDA is iterated again with normalized input data and misclassification matrix is measured using K fold cross validation, where k is set to 20, results obtained are shown in figure M-3.4.1 below.



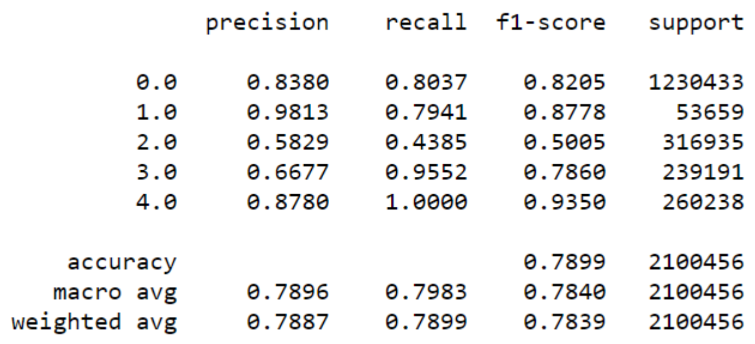
```
              precision    recall  f1-score   support

         0.0     0.9473    0.8943    0.9200   1230433
         1.0     0.8798    0.9427    0.9101     53659
         2.0     0.7799    0.8881    0.8305    316935
         3.0     0.8922    0.9069    0.8995    239191
         4.0     0.9366    0.9985    0.9666    260238

    accuracy                         0.9089   2100456
   macro avg     0.8872    0.9261    0.9053   2100456
weighted avg     0.9127    0.9089    0.9097   2100456
```

Figure-M-3.4.1: Misclassification matrix for QDA with normalization and CV = 20.

It is evident that with respect to QDA with CV=10 this model with CV=20 does not show any significant improvement, thus CV will be set to 10 thereafter. Moreover larger value of K takes extra computational time.

<u>6) Model-3.5: QDA with Normalization, Cross Validation=10 & reg_param=0.5:</u>

QDA is iterated again with normalized input data and misclassification matrix is measured using K fold cross validation, where k is set to 10 and reg_param set to 0.5. Results obtained are shown in figure M-3.5.1 below.



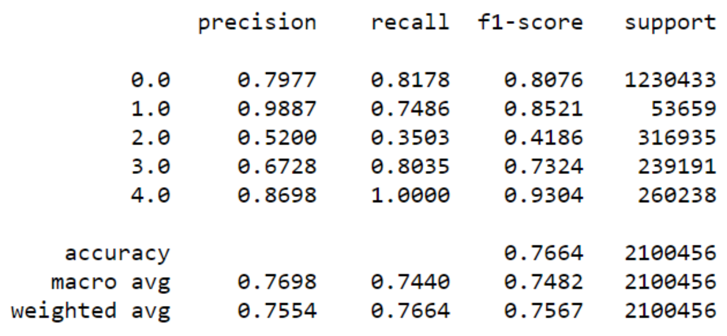
```
              precision    recall  f1-score   support

         0.0     0.8380    0.8037    0.8205   1230433
         1.0     0.9813    0.7941    0.8778     53659
         2.0     0.5829    0.4385    0.5005    316935
         3.0     0.6677    0.9552    0.7860    239191
         4.0     0.8780    1.0000    0.9350    260238

    accuracy                         0.7899   2100456
   macro avg     0.7896    0.7983    0.7840   2100456
weighted avg     0.7887    0.7899    0.7839   2100456
```

Figure-M-3.5.1: Misclassification matrix for QDA with normalization, CV = 10, reg_param = 0.5

It is evident that performance drastically spoils when reg_param is increased from 0 to 0.5 because excessive regularization is done on per class covariance estimation while iterating this algorithm.

<u>7) Model-3.6: QDA with Normalization, Cross Validation=10 & reg_param=0.7:</u>

QDA is iterated again with normalized input data and misclassification matrix is measured using K fold cross validation, where k is set to 10 and reg_param set to 0.7. Results obtained are shown in figure M-3.6.1 below.



```
              precision    recall  f1-score   support

         0.0     0.7977    0.8178    0.8076   1230433
         1.0     0.9887    0.7486    0.8521     53659
         2.0     0.5200    0.3503    0.4186    316935
         3.0     0.6728    0.8035    0.7324    239191
         4.0     0.8698    1.0000    0.9304    260238

    accuracy                         0.7664   2100456
   macro avg     0.7698    0.7440    0.7482   2100456
weighted avg     0.7554    0.7664    0.7567   2100456
```

Figure-M-3.6.1: Misclassification matrix for QDA with normalization, CV = 10, reg_param = 0.7

It is evident that performance drastically spoils when reg_param is increased from 0.5 to 0.7 because excessive regularization is done on per class covariance estimation while iterating this algorithm. This means that algorithm is under fitting with increasing value of reg_param.

8) Model-3.7: QDA with Normalization, Cross Validation=10 & reg_param=0.9:

QDA is iterated again with normalized input data and misclassification matrix is measured using K fold cross validation, where k is set to 10 and reg_param set to 0.9. Results obtained are shown in figure M-3.7.1 below.

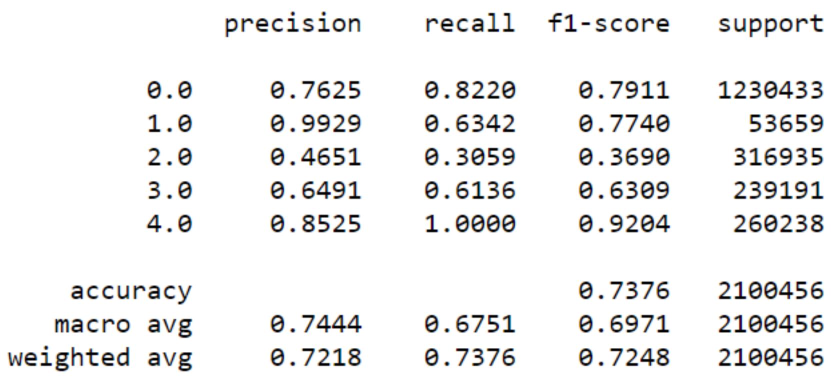

```
              precision    recall  f1-score   support

         0.0     0.7625    0.8220    0.7911   1230433
         1.0     0.9929    0.6342    0.7740     53659
         2.0     0.4651    0.3059    0.3690    316935
         3.0     0.6491    0.6136    0.6309    239191
         4.0     0.8525    1.0000    0.9204    260238

    accuracy                         0.7376   2100456
   macro avg     0.7444    0.6751    0.6971   2100456
weighted avg     0.7218    0.7376    0.7248   2100456
```

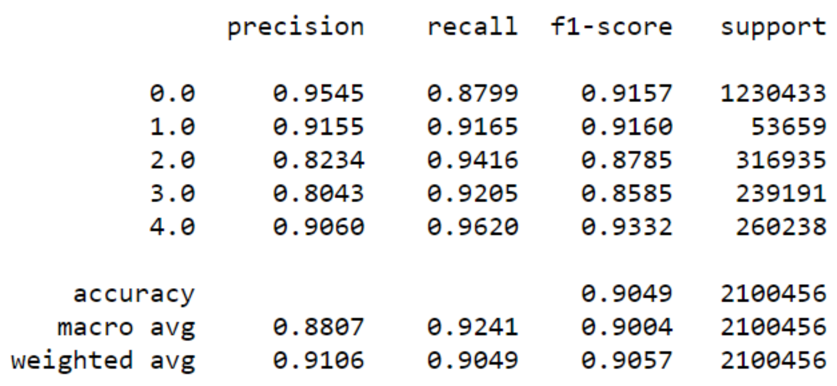
Figure-M-3.7.1: Misclassification matrix for QDA with normalization, CV = 10, reg_param = 0.9

It is evident that performance drastically spoils when reg_param is increased from 0.7 to 0.9 because excessive regularization is done on per class covariance estimation while iterating this algorithm. This means that algorithm is under fitting with increasing value of reg_param and therefore reg_param should be set to 0.

It can be concluded that best QDA model is achieved with reg_param set to 0 and k set to 10 while cross validation.

9) Model-4.2: KNN with Normalization, Cross Validation=20 & K neighbors=5:

KNN is iterated again with normalized input data, k-fold cross validation set to 20 and nearest neighbors set to 5. Results obtained are shown in figure M-4.2.1 below.



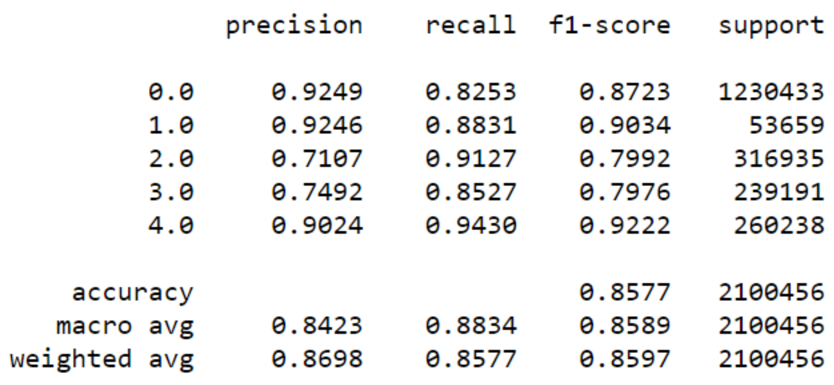
```
              precision    recall  f1-score   support

         0.0     0.9545    0.8799    0.9157   1230433
         1.0     0.9155    0.9165    0.9160     53659
         2.0     0.8234    0.9416    0.8785    316935
         3.0     0.8043    0.9205    0.8585    239191
         4.0     0.9060    0.9620    0.9332    260238

    accuracy                         0.9049   2100456
   macro avg     0.8807    0.9241    0.9004   2100456
weighted avg     0.9106    0.9049    0.9057   2100456
```

Figure-M-4.2.1: Misclassification matrix for KNN with normalization, CV = 20, neighbors =5.

It is evident that performance drastically improves when nearest neighbors are increased from k=3 to k=5, this means that data is not extremely complex. Moreover performance is closing towards QDA model. It should be noted that with k-fold cross validation set to 20, results obtained are very stable and trustworthy. But too much cross validation consumes extra computational time.

<u>10) Model-4.3: KNN with Normalization, Cross Validation=10 & K neighbors=10:</u>

KNN is iterated again with normalized input data, k-fold cross validation set to 10 and nearest neighbors set to 10. Results obtained are shown in figure M-4.3.1 below.



```
              precision    recall  f1-score   support

         0.0     0.9249    0.8253    0.8723   1230433
         1.0     0.9246    0.8831    0.9034     53659
         2.0     0.7107    0.9127    0.7992    316935
         3.0     0.7492    0.8527    0.7976    239191
         4.0     0.9024    0.9430    0.9222    260238

    accuracy                         0.8577   2100456
   macro avg     0.8423    0.8834    0.8589   2100456
weighted avg     0.8698    0.8577    0.8597   2100456
```

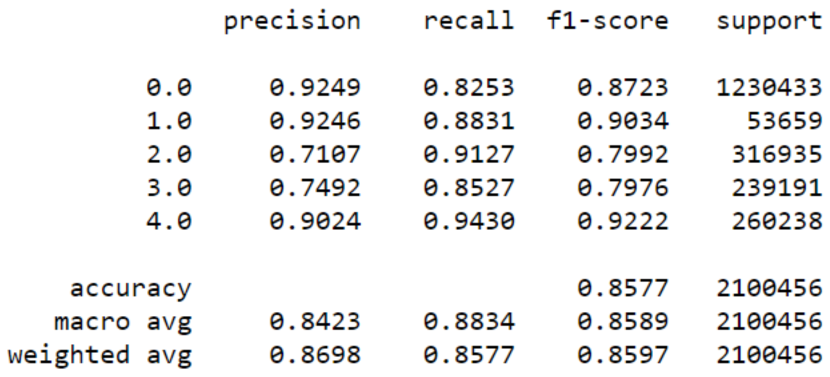
Figure-M-4.3.1: Misclassification matrix for KNN with normalization, CV = 10, neighbors =10.

It is evident that performance drastically improves when nearest neighbors are increased from k=5 to k=10, this means that data is not extremely complex. Moreover performance is closing towards QDA model. It should be noted that with k-fold cross validation set to 10, results obtained are very stable and trustworthy. KNN with CV=10 and K-neighbors=10 fails to outperform QDA, moreover computational time consumed by KNN is very large as compared to QDA.

It can be concluded that with k=10 performance improves, this means that data set is not extremely complex and might have a degree 2 or degree 4 type actual distribution.

<u>11) Model-4.4: KNN with Normalization, Cross Validation=10 & K neighbors=20:</u>

KNN is iterated again with normalized input data, k-fold cross validation set to 10 and nearest neighbors set to 20. Results obtained are shown in figure M-4.4.1 below.



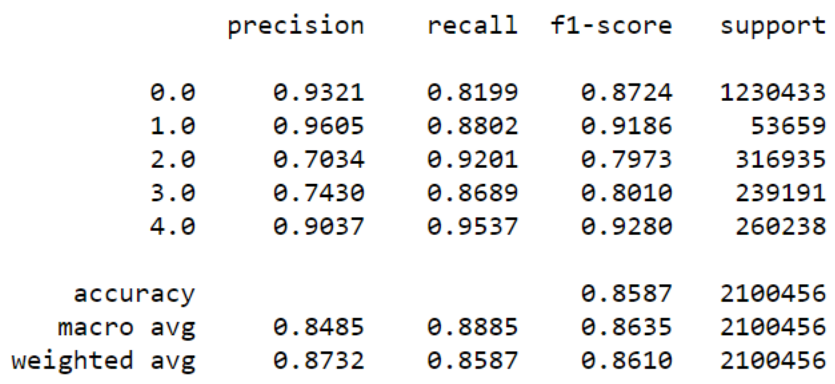
```
              precision    recall  f1-score   support

         0.0     0.9249    0.8253    0.8723   1230433
         1.0     0.9246    0.8831    0.9034     53659
         2.0     0.7107    0.9127    0.7992    316935
         3.0     0.7492    0.8527    0.7976    239191
         4.0     0.9024    0.9430    0.9222    260238

    accuracy                         0.8577   2100456
   macro avg     0.8423    0.8834    0.8589   2100456
weighted avg     0.8698    0.8577    0.8597   2100456
```

Figure-M-4.4.1: Misclassification matrix for KNN with normalization, CV = 10, neighbors =20.

It is evident that performance drastically improves when nearest neighbors are increased from k=10 to k=20,

<u>12) Model-4.5: KNN with Normalization, Cross Validation=10 & K neighbors=30:</u>

KNN is iterated again with normalized input data, k-fold cross validation set to 10 and nearest neighbors set to 30. Results obtained are shown in figure M-4.5.1 below.



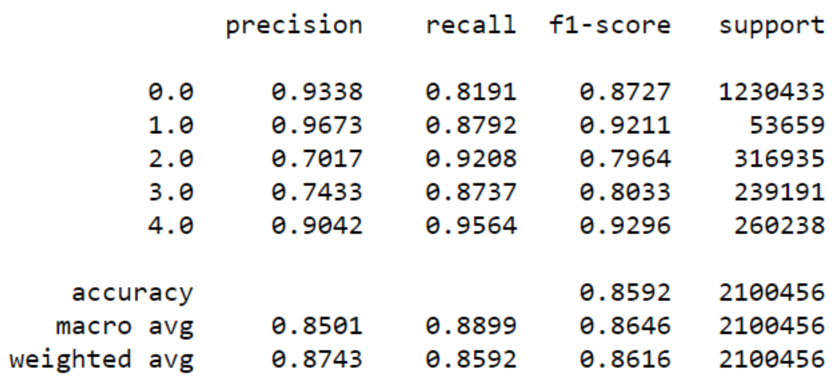
```
              precision    recall  f1-score   support

         0.0     0.9321    0.8199    0.8724   1230433
         1.0     0.9605    0.8802    0.9186     53659
         2.0     0.7034    0.9201    0.7973    316935
         3.0     0.7430    0.8689    0.8010    239191
         4.0     0.9037    0.9537    0.9280    260238

    accuracy                         0.8587   2100456
   macro avg     0.8485    0.8885    0.8635   2100456
weighted avg     0.8732    0.8587    0.8610   2100456
```

Figure-M-4.5.1: Misclassification matrix for KNN with normalization, CV = 10, neighbors =30.

It is evident that performance gets constant with increasing number of K, this means that data is not very complex.

13) Model-4.6: KNN with Normalization, Cross Validation=10 & K neighbors=40:

KNN is iterated again with normalized input data, k-fold cross validation set to 10 and nearest neighbors set to 40. Results obtained are shown in figure M-4.6.1 below.



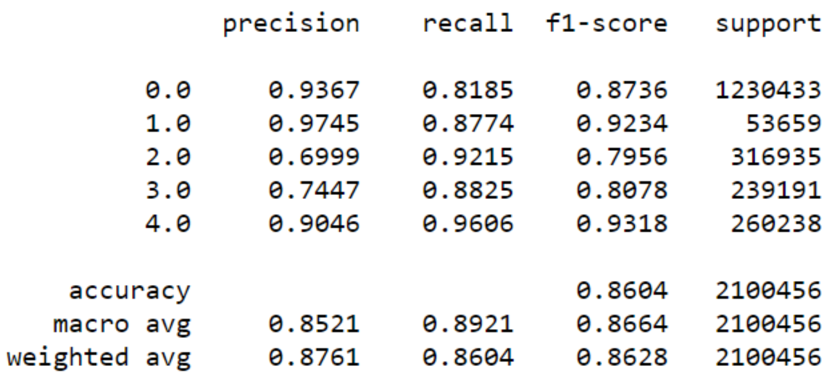
```
              precision    recall  f1-score   support

         0.0     0.9338    0.8191    0.8727   1230433
         1.0     0.9673    0.8792    0.9211     53659
         2.0     0.7017    0.9208    0.7964    316935
         3.0     0.7433    0.8737    0.8033    239191
         4.0     0.9042    0.9564    0.9296    260238

    accuracy                         0.8592   2100456
   macro avg     0.8501    0.8899    0.8646   2100456
weighted avg     0.8743    0.8592    0.8616   2100456
```

Figure-M-4.6.1: Misclassification matrix for KNN with normalization, CV = 10, neighbors =40.

It is evident that performance gets constant with increasing number of K, this means that data is not very complex.

14) Model-4.7: KNN with Normalization, Cross Validation=10 & K neighbors=60:

KNN is iterated again with normalized input data, k-fold cross validation set to 10 and nearest neighbors set to 60. Results obtained are shown in figure M-4.7.1 below.



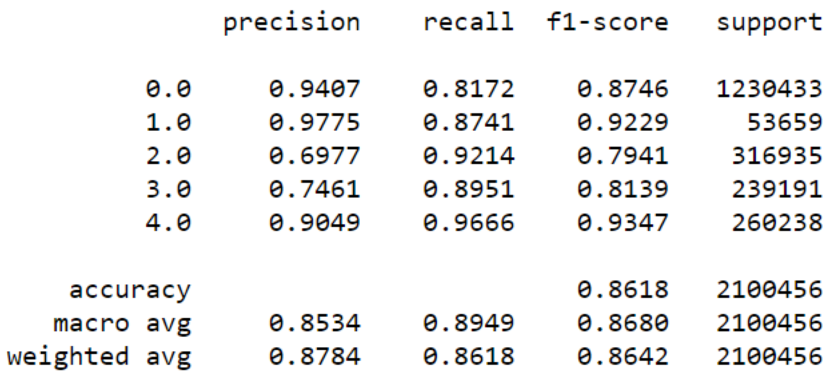
```
              precision    recall  f1-score   support

         0.0     0.9367    0.8185    0.8736   1230433
         1.0     0.9745    0.8774    0.9234     53659
         2.0     0.6999    0.9215    0.7956    316935
         3.0     0.7447    0.8825    0.8078    239191
         4.0     0.9046    0.9606    0.9318    260238

    accuracy                         0.8604   2100456
   macro avg     0.8521    0.8921    0.8664   2100456
weighted avg     0.8761    0.8604    0.8628   2100456
```

Figure-M-4.7.1: Misclassification matrix for KNN with normalization, CV = 10, neighbors =60.

It is evident that performance gets constant with increasing number of K, this means that data is not very complex.

<u>15) Model-4.8: KNN with Normalization, Cross Validation=10 & K neighbors=100:</u>

KNN is iterated again with normalized input data, k-fold cross validation set to 10 and nearest neighbors set to 100. Results obtained are shown in figure M-4.8.1 below.



```
              precision    recall  f1-score   support

         0.0     0.9407    0.8172    0.8746   1230433
         1.0     0.9775    0.8741    0.9229     53659
         2.0     0.6977    0.9214    0.7941    316935
         3.0     0.7461    0.8951    0.8139    239191
         4.0     0.9049    0.9666    0.9347    260238

    accuracy                         0.8618   2100456
   macro avg     0.8534    0.8949    0.8680   2100456
weighted avg     0.8784    0.8618    0.8642   2100456
```

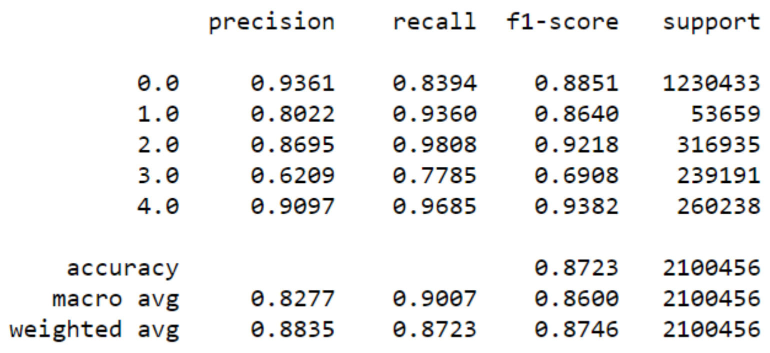
Figure-M-4.8.1: Misclassification matrix for KNN with normalization, CV = 10, neighbors =100.

It is evident that performance gets constant with increasing number of K, this means that data is not very complex.

It can be concluded that data set is not very complex since continuously stable results are obtained as K is increased from 20, 30, 40, 60 to 100. Whereas poor performance was achieved when value of K was very small.

16) Model-5.2: Simple Decision Tree with Depth=5, Normalization, Cross Validation=10:

Simple decision tree can easily overfit if maximum depth for trees in each individual iteration is not controlled, in this case maximum depth is set to 5 and results are evaluated using k-fold cross validation set to 10. Results obtained are shown in figure M-5.2.1 below with normalized data as input.



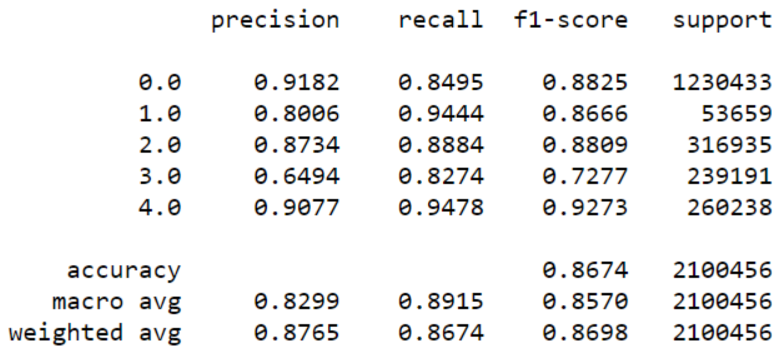
```
              precision    recall  f1-score   support

         0.0     0.9361    0.8394    0.8851   1230433
         1.0     0.8022    0.9360    0.8640     53659
         2.0     0.8695    0.9808    0.9218    316935
         3.0     0.6209    0.7785    0.6908    239191
         4.0     0.9097    0.9685    0.9382    260238

    accuracy                         0.8723   2100456
   macro avg     0.8277    0.9007    0.8600   2100456
weighted avg     0.8835    0.8723    0.8746   2100456
```

Figure-M-5.2.1: Misclassification matrix for Simple Decision tree with cross validation & max depth = 5.

It is evident that performance drastically reduces when K-fold cross validation is used with 10 folds. Misclassification matrix gives stable and trustworthy result with respect random training and test data split.

17) Model-5.3: Simple Decision Tree with Depth=7, Normalization, Cross Validation=10:

Simple decision tree can easily overfit if maximum depth for trees in each individual iteration is not controlled, in this case maximum depth is set to 7 and results are evaluated using k-fold cross validation set to 10. Results obtained are shown in figure M-5.3.1 below with normalized data as input.



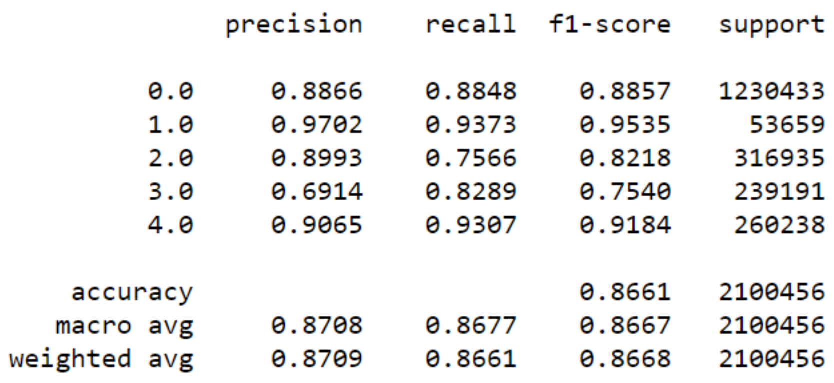
```
              precision    recall  f1-score   support

         0.0     0.9182    0.8495    0.8825   1230433
         1.0     0.8006    0.9444    0.8666     53659
         2.0     0.8734    0.8884    0.8809    316935
         3.0     0.6494    0.8274    0.7277    239191
         4.0     0.9077    0.9478    0.9273    260238

    accuracy                         0.8674   2100456
   macro avg     0.8299    0.8915    0.8570   2100456
weighted avg     0.8765    0.8674    0.8698   2100456
```

Figure-M-5.3.1: Misclassification matrix for Simple Decision tree with cross validation & max depth = 7.

It is evident that performance does not change maximum depth is changed from 5 to 7, this happens because usually tree depths within 10 do not over fit. Since classification error is constant thus increasing depth beyond this point will result in over fitting.

18) Model-5.4: Simple Decision Tree with Depth=5, Normalization, CV=10, Gini Index:

Simple decision tree can easily overfit if maximum depth for trees in each individual iteration is not controlled, in this case maximum depth is set to 5 and results are evaluated using k-fold cross validation set to 10. Results obtained are shown in figure M-5.4.1 below with normalized data as input. Furthermore in this model criterion is set to ginin index, thus minimum or smallest gini index is preferred for optimal performance.



```
              precision    recall  f1-score   support

         0.0     0.8866    0.8848    0.8857   1230433
         1.0     0.9702    0.9373    0.9535     53659
         2.0     0.8993    0.7566    0.8218    316935
         3.0     0.6914    0.8289    0.7540    239191
         4.0     0.9065    0.9307    0.9184    260238

    accuracy                         0.8661   2100456
   macro avg     0.8708    0.8677    0.8667   2100456
weighted avg     0.8709    0.8661    0.8668   2100456
```

Figure-M-5.4.1: Misclassification matrix for Simple Decision tree with CV, Gini Index & max depth = 7.

It is evident that precision matrix changes for class=2 thus gini index performs better than entropy criterion. Since all other misclassification values are almost constant.

It is also very evident that simple decision trees are not able to perform better than QDA models, even after constant training and optimization. Mostly because simple decision trees for RBS algorithm which greedy in nature and generally performs badly with multiclass continuous data.

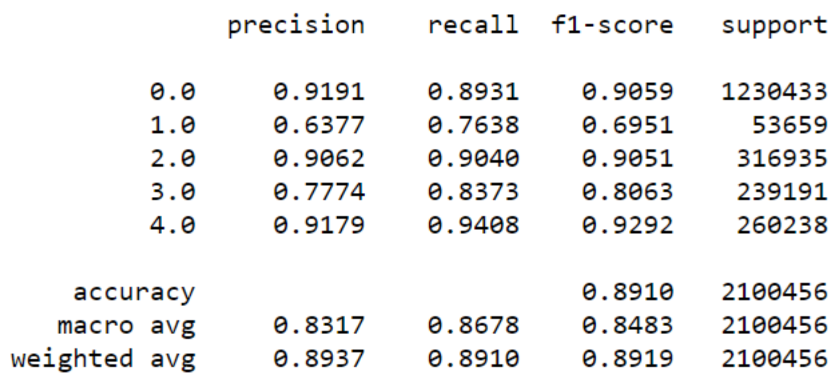
19) Model-6.2: XGBoost decision trees:

XGBoost has various controlling parameters which will be finetuned in this section as shown in table M-6.2.1.

Table-M-6.2.1: Changed hyperparameters for XGBoost algorithm

Max_depth	Alpha	N_estimators	Gamma	Learning Rate	K-Fold-CV
5	10	20	20	0.9	10

In this case XGBoost trees were evaluated on basis of normalized data and K-fold cross validation, which resulted in reduction of all three misclassification matrix. Lower performance was presented by this method as shown in figure M-6-2.2 below.



```
              precision    recall  f1-score   support

         0.0     0.9191    0.8931    0.9059   1230433
         1.0     0.6377    0.7638    0.6951     53659
         2.0     0.9062    0.9040    0.9051    316935
         3.0     0.7774    0.8373    0.8063    239191
         4.0     0.9179    0.9408    0.9292    260238

    accuracy                         0.8910   2100456
   macro avg     0.8317    0.8678    0.8483   2100456
weighted avg     0.8937    0.8910    0.8919   2100456
```

Figure-M-6.2.2: New misclassification matrix achieved by changing hyper parameters.

It should be noted that cross validation based results with normalized data are trust worthy with respect to results presented in base model using random 80% and 20% split. Also in this case learning rate is increased to 0.9 which means algorithm might miss optimal solution or global minima while to converge. Thus high learning rate of 0.9 results in under fitting and therefore spoils the performance of this model.

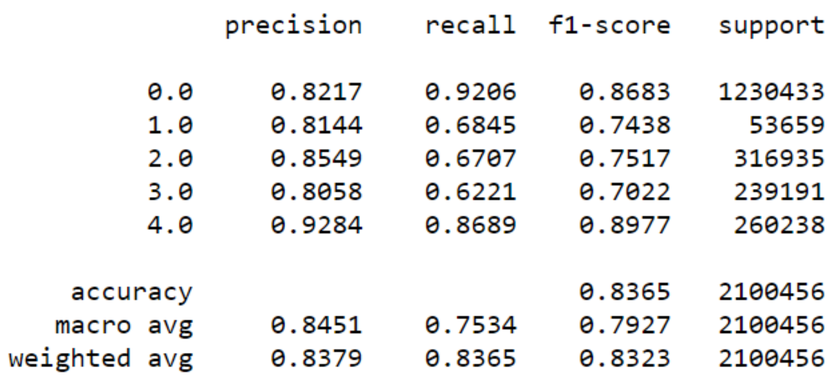
20) Model-6.3: XGBoost decision trees:

XGBoost has various controlling parameters which will be finetuned in this section as shown in table M-6.3.1.

Table-M-6.2.1: Changed hyperparameters for XGBoost algorithm

Max_depth	Alpha	N_estimators	Gamma	Learning Rate	K-Fold-CV
10	10	50	10	0.1	5

In this case XGBoost trees were evaluated on basis of normalized data and K-fold cross validation, which resulted in reduction of all three misclassification matrix. Lower performance was presented by this method as shown in figure M-6-3.2 below.



```
              precision    recall  f1-score   support

         0.0     0.8217    0.9206    0.8683   1230433
         1.0     0.8144    0.6845    0.7438     53659
         2.0     0.8549    0.6707    0.7517    316935
         3.0     0.8058    0.6221    0.7022    239191
         4.0     0.9284    0.8689    0.8977    260238

    accuracy                         0.8365   2100456
   macro avg     0.8451    0.7534    0.7927   2100456
weighted avg     0.8379    0.8365    0.8323   2100456
```

Figure-M-6.3.2: New misclassification matrix achieved by changing hyper parameters.

It should be noted that increasing maximum depth and total number of estimators to 10 and 50 respectively, where as K-fold cross validation is reduce to 5 folds thus small value of K results in under fitting. Thus performance of XGBoost tree considerably spoils due to under fitting because of small value of K-folds.

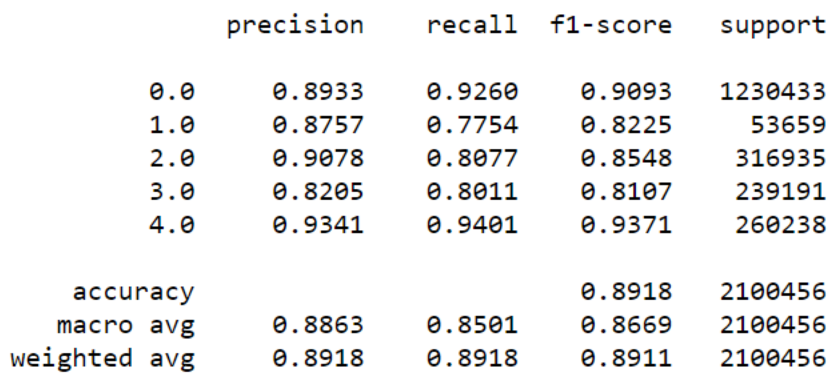
21) Model-6.4: XGBoost decision trees:

XGBoost has various controlling parameters which will be finetuned in this section as shown in table M-6.4.1.

Table-M-6.4.1: Changed hyperparameters for XGBoost algorithm

Max_depth	Alpha	N_estimators	Gamma	Learning Rate	K-Fold-CV
10	10	50	10	0.1	10

In this case XGBoost trees were evaluated on basis of normalized data and K-fold cross validation, which resulted in improvement of all three misclassification matrix. Higher and better performance was presented by this method as shown in figure M-6-4.2 below.



```
              precision    recall  f1-score   support

         0.0     0.8933    0.9260    0.9093   1230433
         1.0     0.8757    0.7754    0.8225     53659
         2.0     0.9078    0.8077    0.8548    316935
         3.0     0.8205    0.8011    0.8107    239191
         4.0     0.9341    0.9401    0.9371    260238

    accuracy                         0.8918   2100456
   macro avg     0.8863    0.8501    0.8669   2100456
weighted avg     0.8918    0.8918    0.8911   2100456
```

Figure-M-6.4.2: New misclassification matrix achieved by changing hyper parameters.

It should be noted that increasing maximum depth and total number of estimators to 10 and 50 respectively results in increasing complexity of the tree. Larger value of depth and estimators will result better handling of complex data, since more number of tree will handle complexity in an optimized way. Whereas K-fold cross validation is increased to 10 folds thus this is an optimal value for K, since performance of XGBoost tree considerably improves. Thus higher value of K, more number of trees and larger depth helps to enhance performance.

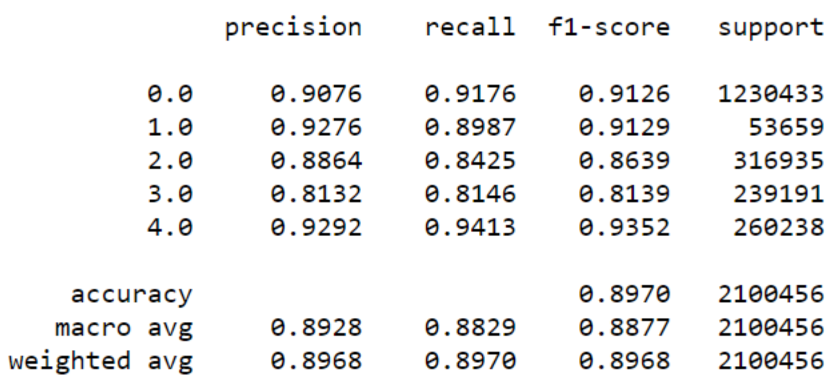
22) Model-6.5: XGBoost decision trees:

XGBoost has various controlling parameters which will be finetuned in this section as shown in table M-6.5.1.

Table-M-6.5.1: Changed hyperparameters for XGBoost algorithm

Max_depth	Alpha	N_estimators	Gamma	Learning Rate	K-Fold-CV
15	10	70	10	0.1	10

In this case XGBoost trees were evaluated on basis of normalized data and K-fold cross validation, which resulted in improvement of all three misclassification matrix. Higher and better performance was presented by this method as shown in figure M-6-5.2 below.



```
              precision    recall  f1-score   support

         0.0     0.9076    0.9176    0.9126   1230433
         1.0     0.9276    0.8987    0.9129     53659
         2.0     0.8864    0.8425    0.8639    316935
         3.0     0.8132    0.8146    0.8139    239191
         4.0     0.9292    0.9413    0.9352    260238

    accuracy                         0.8970   2100456
   macro avg     0.8928    0.8829    0.8877   2100456
weighted avg     0.8968    0.8970    0.8968   2100456
```

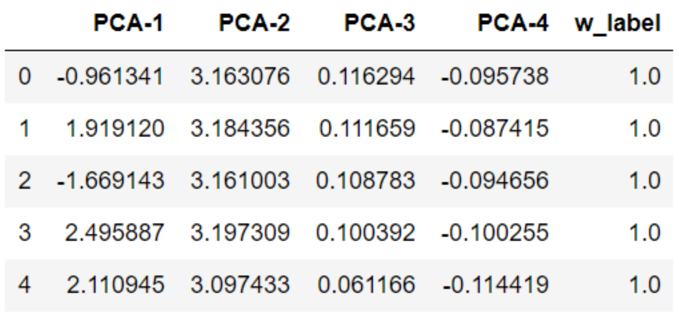
Figure-M-6.5.2: New misclassification matrix achieved by changing hyper parameters.

It should be noted that increasing maximum depth and total number of estimators to 15 and 70 respectively results in increasing complexity of the tree. Larger value of depth and estimators will result better handling of complex data, since more number of tree will handle complexity in an optimized way.

It is worth mentioning with larger depth of 15, more number of trees to 70 and small learning rate of 0.1 computational time taken to converge by this method is very large and results in very complex tree. Results can not be easily interpreted and this performance fails to match QDA.

<u>Method-2: Fine Tuning Feature Set via Dimension Reduction</u>

Principle component analysis is used in this project for dimension reduction. Since total number of features in data set are 8 thus maximum 4 components are allowed, where all 4 components are orthogonal to each other as shown in figure M-P.1 below.

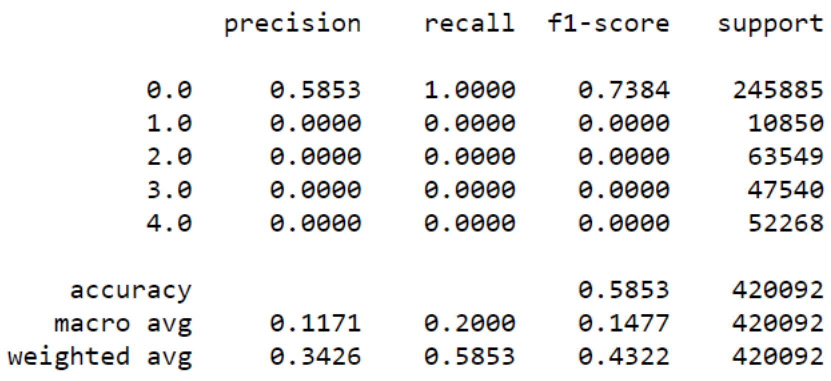


	PCA-1	PCA-2	PCA-3	PCA-4	w_label
0	-0.961341	3.163076	0.116294	-0.095738	1.0
1	1.919120	3.184356	0.111659	-0.087415	1.0
2	-1.669143	3.161003	0.108783	-0.094656	1.0
3	2.495887	3.197309	0.100392	-0.100255	1.0
4	2.110945	3.097433	0.061166	-0.114419	1.0

Figure M-P.1: All 4 principle components which are orthogonal to each other.

23) Model-P.1: XGBoost decision tree with ONE Principle Component

Optimal parameters suggested above for XGBoost decision tree are used with one principle component as input. Thus total features of the data set are reduced to only ONE feature by this method. Figure P.1.1 shows misclassification matrices for this method.



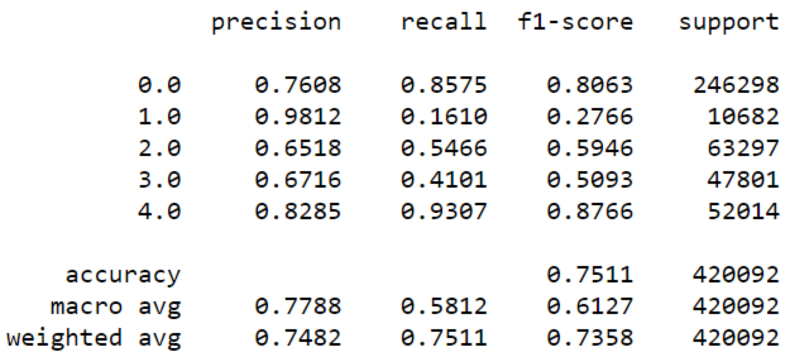
```
              precision    recall  f1-score   support

         0.0     0.5853    1.0000    0.7384    245885
         1.0     0.0000    0.0000    0.0000     10850
         2.0     0.0000    0.0000    0.0000     63549
         3.0     0.0000    0.0000    0.0000     47540
         4.0     0.0000    0.0000    0.0000     52268

    accuracy                         0.5853    420092
   macro avg     0.1171    0.2000    0.1477    420092
weighted avg     0.3426    0.5853    0.4322    420092
```

Figure-P.1.1: XGBoost with one principle component.

It is very evident that performance of XGBoost with best parameters completely spoils when only ONE principle component is used. This proves that data set was highly uncorrelated and PCA can not work with uncorrelated data.

24) Model-P.2: XGBoost decision tree with TWO Principle Components

Optimal parameters suggested above for XGBoost decision tree are used with TWO principle components as input. Thus total features of the data set are reduced to TWO features by this method. Figure P.2.1 shows misclassification matrices for this method.



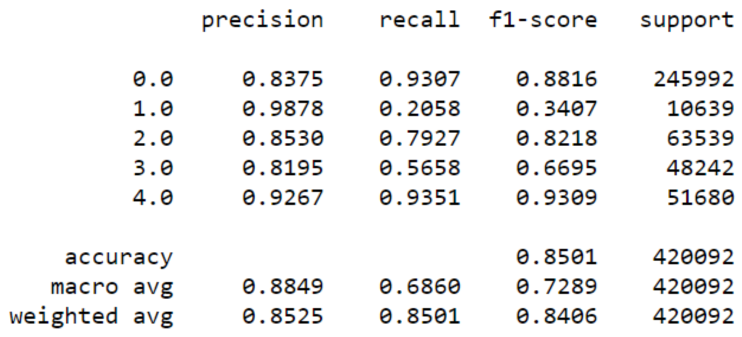
```
              precision    recall  f1-score   support

         0.0     0.7608    0.8575    0.8063    246298
         1.0     0.9812    0.1610    0.2766     10682
         2.0     0.6518    0.5466    0.5946     63297
         3.0     0.6716    0.4101    0.5093     47801
         4.0     0.8285    0.9307    0.8766     52014

    accuracy                         0.7511    420092
   macro avg     0.7788    0.5812    0.6127    420092
weighted avg     0.7482    0.7511    0.7358    420092
```

Figure-P.2.1: XGBoost with two principle components.

It is very evident that performance of XGBoost with best parameters completely spoils when only TWO principle components are used. This proves that data set was highly uncorrelated and PCA can not work with uncorrelated data.

25) Model-P.3: XGBoost decision tree with THREE Principle Components

Optimal parameters suggested above for XGBoost decision tree are used with THREE principle components as input. Thus total features of the data set are reduced to THREE features by this method. Figure P.3.1 shows misclassification matrices for this method.



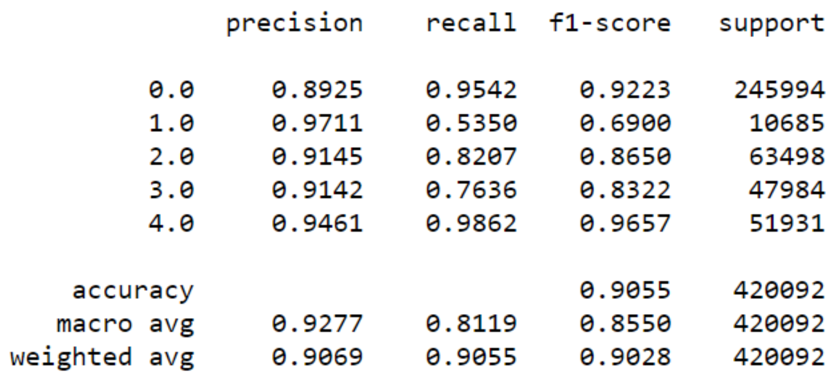
```
              precision    recall  f1-score   support

         0.0     0.8375    0.9307    0.8816    245992
         1.0     0.9878    0.2058    0.3407     10639
         2.0     0.8530    0.7927    0.8218     63539
         3.0     0.8195    0.5658    0.6695     48242
         4.0     0.9267    0.9351    0.9309     51680

    accuracy                         0.8501    420092
   macro avg     0.8849    0.6860    0.7289    420092
weighted avg     0.8525    0.8501    0.8406    420092
```

Figure-P.3.1: XGBoost with three principle components.

It is very evident that performance of XGBoost with best parameters completely spoils when only THREE principle components are used. This proves that data set was highly uncorrelated and PCA can not work with uncorrelated data. Although there is some improvement in precision matrix but FI-score and recall matrix are still very low. Which proves that PCA with 3 components is not able to optimize test performance.

26) Model-P.4: XGBoost decision tree with FOUR Principle Components

Optimal parameters suggested above for XGBoost decision tree are used with FOUR principle components as input. Thus total features of the data set are reduced to FOUR features by this method. Figure P.4.1 shows misclassification matrices for this method.



```
              precision    recall  f1-score   support

         0.0     0.8925    0.9542    0.9223    245994
         1.0     0.9711    0.5350    0.6900     10685
         2.0     0.9145    0.8207    0.8650     63498
         3.0     0.9142    0.7636    0.8322     47984
         4.0     0.9461    0.9862    0.9657     51931

    accuracy                         0.9055    420092
   macro avg     0.9277    0.8119    0.8550    420092
weighted avg     0.9069    0.9055    0.9028    420092
```

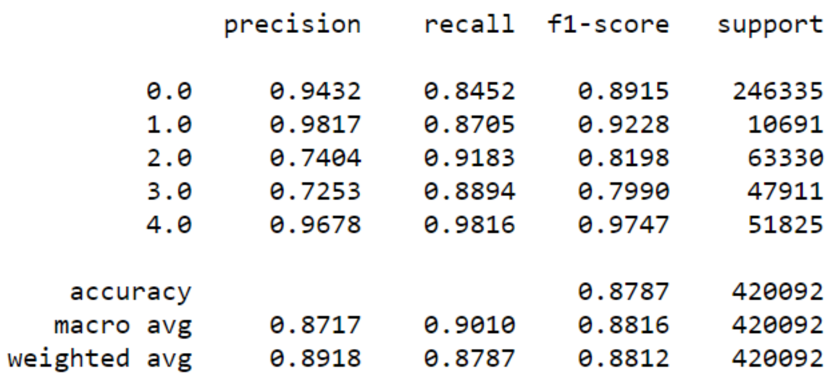
Figure-P.4.1: XGBoost with four principle components.

It is very evident that performance of XGBoost with best parameters slightly improves and becomes closer to performance attained by original data set with all the features. Thus it can be concluded that reducing features for this data set is not a good option. Since data set is highly uncorrelated thus feature removal will increase error and spoil overall accuracy.

It can be concluded that PCA with one, two, three and four principle components fails to perform better than originally attained performance by including all features of the data set.

27) Model-P.5: Simple decision tree with FOUR Principle Components

Optimal parameters suggested above for simple decision tree are used with FOUR principle components as input. Thus total features of the data set are reduced to FOUR features by this method. Figure P.5.1 shows misclassification matrices for this method.



```
              precision    recall  f1-score   support

         0.0     0.9432    0.8452    0.8915    246335
         1.0     0.9817    0.8705    0.9228     10691
         2.0     0.7404    0.9183    0.8198     63330
         3.0     0.7253    0.8894    0.7990     47911
         4.0     0.9678    0.9816    0.9747     51825

    accuracy                         0.8787    420092
   macro avg     0.8717    0.9010    0.8816    420092
weighted avg     0.8918    0.8787    0.8812    420092
```

Figure-P.5.1: Simple Decision Tree with four principle components.

It is very evident that performance of simple decision tree with best parameters slightly improves and becomes closer to performance attained by original data set with all the features. Thus it can be concluded that reducing features for this data set is not a good option. Since data set is highly uncorrelated thus feature removal will increase error and spoil overall accuracy.

It can be concluded that PCA with one, two, three and four principle components fails to perform better than originally attained performance by including all features of the data set. Moreover PCA based performance consumes 140 minutes to converge.

28) Model-P.6: KNN with K=5 and FOUR Principle Components

Optimal parameters suggested above for KNN are used with FOUR principle components as input. Thus total features of the data set are reduced to FOUR features by this method. Figure P.6.1 shows misclassification matrices for this method.

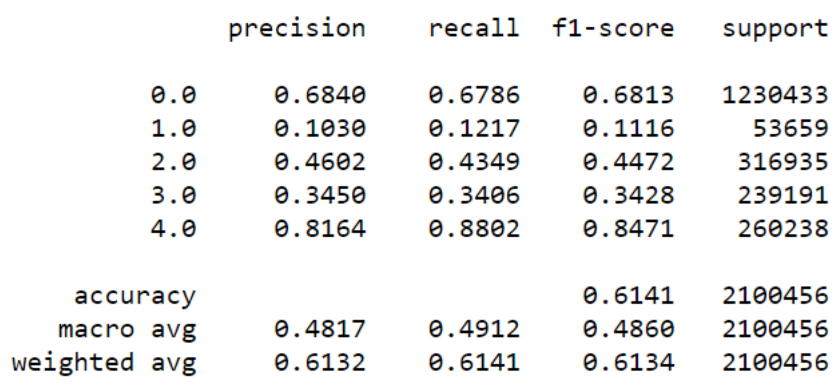

```
              precision    recall  f1-score   support

         0.0     0.6840    0.6786    0.6813   1230433
         1.0     0.1030    0.1217    0.1116     53659
         2.0     0.4602    0.4349    0.4472    316935
         3.0     0.3450    0.3406    0.3428    239191
         4.0     0.8164    0.8802    0.8471    260238

    accuracy                         0.6141   2100456
   macro avg     0.4817    0.4912    0.4860   2100456
weighted avg     0.6132    0.6141    0.6134   2100456
```

Figure-P.6.1: KNN with K=5 and four principle components.

It is very evident that performance of KNN with best parameters completely spoils when used with PCA having 4 principle components. This happens data is highly uncorrelated.

<u>Conclusion of doing PCA</u>

Thus PCA constantly performs bad results with KNN, Simple decision trees and XGBoost models. Hence feature reduction for this data set in order to optimize performance is a bad choice. Because PCA works with correlated data and WESAD data set is highly uncorrelated.

It can be concluded that PCA with one, two, three and four principle components fails to perform better than originally attained performance by including all features of the data set.

H) Performance of all models:

It should be noted that all misclassification matrices like Precision, Recall, Accuracy and F1-Score are given individually for each and every model in Section-G, where results attained are discussed individually.

Table H-1 shown below contains performance analysis for all models build till now, it should be noted that total of 28 models were made and tested already.

Please check **section G** which has individual performances and misclassification matrices clearly shown and discussed in depth. **Conclusion for all models built are also presented in section G.** In this section tabular representation of all models will be displayed and best model will be selected according to highest scores.

Table-H-1: Performance analysis via Precision, Recall, Accuracy and F1 scores of each model.

Index	Name	Precision	Recall	Accuracy	F1 Score	Time Taken
Base-1	Logistic Regression Base Model	0.83	0.72	0.8	0.75	30 minutes
Base-2	LDA	0.8	0.73	0.79	0.75	30 minutes
Base-3	QDA	0.89	0.95	0.92	0.92	5 minutes
Base-4	KNN + K=3 + CV=10	0.83	0.88	0.85	0.85	20 minutes
Base-5	Simple DT + NO CV	0.93	0.94	0.93	0.93	5 minutes
Base-6	XGB + NO CV	0.98	0.97	0.97	0.97	5 minutes
Base-7	SVC + LESS DATA + NO CV	0.78	0.6	0.74	0.64	1 day
Base-8	SVM + LESS DATA + NO CV	0.86	0.77	0.85	0.8	2 days
Modifications						
1	LR + L2	0.84	0.8	0.82	0.81	40 minutes
2	LDA + CV=5	0.8	0.74	0.79	0.75	40 minutes
3	QDA + CV=5	0.86	0.85	0.86	0.85	10 minutes

4	QDA + CV=10	0.88	0.92	0.9	0.9	10 minutes
5	QDA + Reg=0.5 + CV=10	0.78	0.79	0.78	0.78	15 minutes
6	QDA + Reg=0.7 + CV=10	0.76	0.744	0.76	0.74	15 minutes
7	QDA + Reg=0.9 + CV=10	0.74	0.67	0.73	0.69	15 minutes
8	QDA + CV=20	0.88	0.94	0.91	0.91	30 minutes
9	KNN + K=5 + CV=20	0.88	0.92	0.9	0.9	30 minutes
10	KNN + K=10 + CV=10	0.84	0.88	0.85	0.85	30 minutes
11	KNN + K=20 + CV=10	0.84	0.88	0.85	0.86	40 minutes
12	KNN + K=30 + CV=10	0.84	0.88	0.85	0.86	40 minutes
13	KNN + K=40 + CV=10	0.85	0.88	0.85	0.86	40 minutes
14	KNN + K=60 + CV=10	0.85	0.89	0.86	0.86	50 minutes
15	KNN + K=100 + CV=10	0.85	0.89	0.86	0.86	50 minutes
16	Simple DT + CV=10 + Depth=5	0.82	0.9	0.87	0.86	30 minutes
17	Simple DT + CV=10 + Depth=7	0.82	0.89	0.86	0.85	30 minutes
18	Simple DT + GINI + CV=10 + Depth=7	0.87	0.86	0.86	0.86	30 minutes
19	XGB+CV=5+Rate=0.9+D=5+N=20	0.83	0.86	0.89	0.84	30 minutes
20	XGB+CV=5+Rate=0.1+D=5+N=50	0.84	0.75	0.83	0.79	40 minutes
21	XGB+CV=10+Rate=0.1+D=10+N=50	0.88	0.85	0.89	0.86	40 minutes
22	XGB+CV=10+Rate=0.1+D=15+N=70	0.89	0.88	0.89	0.88	40 minutes
23	PCA-1 + XGBoost	0.1	0.2	0.5	0.1	40 minutes
24	PCA-2 + XGBoost	0.77	0.58	0.75	0.61	40 minutes
25	PCA-3 + XGBoost	0.88	0.68	0.85	0.72	40 minutes
26	PCA-4 + XGBoost	0.92	0.811	0.9	0.85	40 minutes
27	PCA-4 + Simple Decision Tree	0.87	0.9	0.87	0.88	40 minutes
28	PCA-4 + KNN K=5	0.47	0.49	0.61	0.48	40 minutes

Logistic regression could not perform well for given data set, since it assumes that data is linearly separable. L2 regularization helps in slightly improving the misclassification matrices.

SVC gave poorest performance, since it assumes that data is linearly separable.

SVM with radial bias kernel took 2 days to converge and did not appropriate results.

LDA performed better than logistic regression.

QDA gave outstanding results in smallest amount of time, it works better with uncorrelated data.

KNN with higher values of K like 30, 40, 60 or 100 give almost similar performances, which means data is not of degree 4 or 5. But it still can’t match QDA.

Similarly simple decision based trees and XGBoost models perform slightly below QDA because these models tend to overfit and have too many controlling parameters. Moreover they tend to perform badly with sequential data.

The results declared in table-H-1 confirm that QDA outperforms all other models, please visit section G for viewing snapshots of misclassification matrices for all 28 models. Results in this section are provided in tabular form, for performance analysis with better view.

Table-H-2: Best model QDA tested for 7 different subjects with overall performance analysis.

Name	BEST MODEL	Precision	Recall	Accuracy	F1-score	Time
Subject-2	QDA + CV=10 (BEST MODEL)	0.88	0.92	0.90	0.90	10 minutes
Subject-3	QDA + CV=10 (BEST MODEL)	0.91	0.91	0.93	0.91	10 minutes
Subject-4	QDA + CV=10 (BEST MODEL)	0.91	0.95	0.96	0.93	10 minutes
Subject-10	QDA + CV=10 (BEST MODEL)	0.88	0.94	0.93	0.9	10 minutes
Subject-11	QDA + CV=10 (BEST MODEL)	0.71	0.88	0.82	0.76	10 minutes
Subject-16	QDA + CV=10 (BEST MODEL)	0.81	0.89	0.88	0.85	10 minutes
Subject-17	QDA + CV=10 (BEST MODEL)	0.83	0.90	0.85	0.86	10 minutes
Overall	QDA + CV=10 (BEST MODEL)	0.86	0.90	0.89	0.87	10 minutes

Consider table-H-2 that provides performance analysis of my best model QDA for 7 different subjects available in WESAD data set. It should be noted that overall performance analysis of this QDA model provides excellent results for 10 GB large data set within 10 minutes of computational time.

It is worth mentioning that Recall score is constantly around 90% for each and every class. **Thus this means that 90% of the time correct predictions were made out of actual / true samples present in that class.**

I) Overall discussion of results & Conclusions

In this section 2 discussions are presented, overall summary of data set in section I-1 and overall discussions about the results with proper reasoning behind them in section I-2.

I-1) Summary of the data set and WESAD discussion:

A) WESAD data was not provided in form simple dictionary, structure of dictionary made it very difficult to be converted into pandas data frame.

B) Creating data frame was challenging, data had too many outliers and missing values.

C) Unbalanced data was collected from chest device and wrist device.

D) Unbalanced data was collected for all 4 different classes or labels.

E) Data cleaning, removal of outliers and data pre-processing took too much time.

F) Data set was huge and combining all subjects together resulted in hanging the laptop, thus models were build from one subjected and tested on 7 different subjects.

I-2) Discussion about the results:

1) In machine learning problems it is very difficult to guess the degree of data set, but in this project after developing 28 models it is evident that data is not very complex, degree of data is quadratic and data is highly uncorrelated.

2) QDA outperforms very simple and extremely complex models this proves that data is somewhere around degree 2. QDA assumes that data has gaussian distribution which is true in this case. QDA works well with data set having different variance, gaussian distribution and is more flexible than naïve Bayes classifier.

3) Complex models like KNN constantly provide same result with increasing number of K. As number of nearest neighbors are increased from 10, 20, 30, 40, 60 to 100 the accuracy, precision and recall values remain the constant and tend to touch QDA performance. This means that data set is not very complex.

4) KNN with small number of nearest neighbors tend to over fit the data set and performance is poor until higher values of K are achieved.

5) K-Fold cross validation helps in attaining stable results and fluctuations in misclassification matrices is strictly reduced. Similarly K=10 provides optimized results for most of the models.

6) Logistic regression assumes that data is not gaussian, which is completely false in this case. Thus it performs poorly. Moreover data is not linearly separable in this case.

7) LDA assumes data is gaussian but is unable to perform in this case because it assumes that data is linearly separable which is false in this case. QDA is more flexible than LDA.

8) L1 and L2 regularization improve results for all the models, thus it should be noted that coefficients of less important features do not contribute in predicting final labels. Thus regularization is very helpful in enhancing performance of the models.

9) SVM with RBF fails to converge early and does not provide good results. Major challenge with SVM is that it takes too much computational time and fine tuning this model is very difficult. In this case it took 2 days to provide few result. Since data is not very complex thus smaller value of gamma could be used, but this model could not be fine tuned.

10) Forward selection and Backward selection methods prove that all features in WESAD data set are equally important. The same results are confirmed by sns heat map and correlation matrix. Thus all features are included while making optimized models.

11) Simple decision trees are easy to interpret and converge quickly but fail to outperform QDA performance. This happens because of over fitting since depth and number of leaves can not be fixed with sequential data. Moreover for multi class sequential data classification problem of lower complexity, decision trees generally perform poorly since RBS algorithm is greedy in nature.

12) XGBoost decision trees use gradient boosting technique but have many controlling parameters with L1 and L2 regularization options. With multiclass sequential data XGBoost trees generally fail to optimize test error. Moreover hardcoding hyperparameters like learning rate, tree depth, number of leaves is not an optimized solution for sequential data.

13) Learning rate in XGBoost model is major cause for slow convergence, very small values take days to converge and result in over fitting. Whereas very large value of learning rate results in missing global minima and thus tend to over fit.

14) PCA works with uncorrelated data, in this data set all features are highly uncorrelated. Thus PCA constantly performs poorly with KNN, Simple Decision Tree and XGBoost models. As principle components are increased form 1, 2, 3 to 4 performance also improves. This confirms that data is uncorrelated and all features are extremely important.

Table-I-2: Average performance analysis of best model against 7 different subjects.

Name	MODEL	Precision	Recall	Accuracy	F1-Score	Time
Overall Average Performance for 7 subjects	QDA + CV=10 (BEST MODEL)	0.86	0.90	0.89	0.87	10 Min

15) Best QDA model was tested for 7 different subjects in WESAD data set and my QDA model provided excellent results as shown in table-I-2 above. It is worth mentioning that Recall score is constantly around 90% for each and every class. **Thus this means that 90% of the time correct predictions were made out of actual / true samples present in that class.**

<u>CONCLUSIONS</u>

1) We performed better than baseline paper provided by WESAD repository. For classifying baseline state, stress state and amusement state score of 80% was achieved by AdaBoost Decision Tree. **Whereas I obtained 90% score** for classifying 4 different states of baseline, stress, amusement and meditation using QDA model. Thus my best outperformed models presented by baseline paper.

2) Best model was created and verified after iterating 28 different models and 8 base models. **<u>Thus I made 36 different models</u>** for this project before finalizing my best model.

3) Actual complexity and degree of data can be understood via K-nearest Neighbor model. If high recall and precision values are available for low K value like 2,3,5 then data is complex. On contrary if K=30, 40, 100 provides better results then data set is of lower complexity or polynomial order.

4) XGBoost model is not the best classification model for sequential data and fine tuning more than 7 parameters including learning rate and depth are difficult to manage.

5) My QDA model provides precision of 86%, recall of 90%, accuracy of 89% and F1 score of 87% as an average performance analysis for 7 different subjects.

6) QDA works well with data set having different variance, gaussian distribution and is more flexible than naïve Bayes classifier.

References

I would like express my thanks coders and contributors mentioned below.

A) BradySheehan for his wesad_experiments project in github, his link is given below.

[1] https://github.com/BradySheehan/wesad_experiments

B) WJ Mathew for WESAD project on git hub

[2] https://github.com/WJMatthew/WESAD