# IIVP Project Topic:

# Mental State Recognition using EEG Signal


**Group No. 1**

**IIT2018114 -** Harsh Goyal

**IIT2018144 -** Aaditya Gadhave

**IIT2018149 -** Sourabh Gupta

**IIT2018158 -** Meet Singh Gambhir

**IIT2018159 -** Tushar Atrey

**Branch -** Information Technology

**Semester -** V


| | |
|---|---|
| **Instructor:** | **Dr. Anupam Agarwal, Gopal Chandra Jana** |
| **Course:** | **IIVP** |
| **Date:** | **15 - 11 - 2020** |


**Indian Institute of Information Technology, Allahabad, Prayagraj.**

# Table of Contents

## *Abstract :*

This work presents an image classification approach to EEG brainwave classification. The proposed method is based on the representation of temporal and statistical features as a 2D image, which is then classified using **Support vector machines**. A three-class mental state problem is investigated, in which subjects experience either relaxation, concentration, or neutral states. Using publicly available EEG data from a Muse Electroencephalography headband, a large number of features describing the wave are extracted of 4 subjects, and subsequently reduced based using 5 different methods. These selected features are then normalised and reshaped, which can be expressed as a grayscale image. A **Support vector machine** is then trained on this data in order to classify the mental state of subjects.



**Keywords:** Machine Learning,Support vector machines, Image Recognition, Mental State Classification


## *Introduction and Motivation :*

Human-machine interaction is often considered a mirror of the human experience; sound and visuals constitute voice recognition, human activity classification, facial recognition, sentiment analysis and so on. Though, with the avail- ability of sensors to gather data that the human body cannot, interaction with machines can often exceed the abilities of the natural human experience. An example of this is the consideration of electroencephalographic brainwaves. The brain, based on what a person is thinking, feeling, or doing, has a unique pat- tern of electrical activity that emerges as a consequence of the aggregate firing patterns of billions of individual neurons [1, 2]. These electrical signals can, in principle, be detected and processed to infer the state of the brain and, by extension, the mental state of a given subject. Besides clinical applications, this possibility is also useful, e.g., for brain-machine interfacing. This work focuses on the process of feature extraction, selection and format- ting in order to achieve improved classification accuracy of EEG signals. More specifically, the main contribution is a framework to perform classification of these signals, based on (i) the extraction of a large number of static statistical features of the data, followed by (ii) automated feature selection and (iii) repre- sentation of the selected attributes as a 2D matrix. The resulting matrices are (iv) interpreted as grayscale images, which allows the leveraging of the state-of- the-art performance of convolutional neural networks [5, 6] as image classifiers.

# *Problem Definition & Objectives :*

**Project Title :** State Recognition using EEG Signal

**Objective :** Our main motive is to predict the three-class mental state of any subject in which they experience either relaxation, concentration, or neutral states. Therefore we first extract features from the pre-processed EEG signals data, then we select the important features from it. Then we make it pass through a SVM model which allows us to classify the mental state of the subject.

# *Literature Review :*

| Sr. No | Author Name | Paper Title | Conference/Journal Paper with year | Method/approach used | Application domain | Dataset used | Achieved Performance | Advantages and Disadvantages | Feature scope |
|---|---|---|---|---|---|---|---|---|---|
| 1. | Jodie Ashford Jordan J. Bird Felipe Campelo Diego R. Faria | Classification of EEG Signals Based on Image Representation of Statistical Features | UK Workshop on Computational Intelligence, 30 August 2019 | Convolutional Neural Networks, Electroencephalography | EEG-based | DEAP dataset | CNN achieved 81.41% accuracy for valence and 73.35% for arousal | Advantage: The extracted features were reduced to optimum number of features Disadvantage:EEG recorded during a cognitive task is relatively easy to separate from a baseline EEG than | Extending the application of the proposed method to various image recognition system for predicting emotions |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | EEG signals recorded in two different cognitive tasks | |
| 2. | Jordan J. Bird , Diego R. Faria, Luis J. Manso, Anikó Ekárt, and Christopher D. Buckingham | A Deep Evolutionary Approach to Bioinspired Classifier Optimisation for Brain-Machine Interaction | *School of Engineering and Applied Science, Aston University, Birmingham B4 7ET, UK* | Deep learning ,Adaptive Boosted LSTM, Adaptive Boosted DEvo MLP , nonboosted DEvo MLP | Deep Evolutionary Approach , EEG-Based | The EEG MindBigData digit ,Muse EEG headband | Adaptive Boosted LSTM can achieve an accuracy of 84.44%, 97.06% , Adaptive Boosted DEvo MLP reaching 31.35%, DEvo MLP was of 79.81% | The wavelet transforms have certain advantages compared to Fourier transform since they do not require the use of fixed data windows and are localized in both time and frequency, whereas the Fourier transform is only localized in frequency | The developed system can help neurophysiologists identify EEG patterns in epilepsy diagnostic tasks. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| 3. | Sander Koelstra, Mohammad Soleymani, Ashkan Yazdani, Anton Nijholt | DEAP: A Database for Emotion Analysis Using Physiological Signals | IEEE TRANSACTIONS ON AFFECTIVE COMPUTING | Emotion classification, EEG, physiological signals, signal processing, pattern classification, affective computing. | EEG signal reconstruction and classification | DEAP dataset | Best classification accuracy achieved is 67.67% which is for AO dataset.<br><br>Best reconstructed data accuracy is 73.63% for GAN dataset. | Advantage - The datasets are easily available and high accuracy classification can be achieved using these.<br>Disadvantage - The memory required is very high for storing datasets and a single dataset can not be used for accuracy in all fields. | Focus on reconstruction signal amplitude ranges of EEG signals with different sensitivity |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| 4. | Sunhee Hwang, Kibeom Hong, Guiyoung Son, Hyeran Byun | Mental Emotional Sentiment Classification with an EEG-based Brain-machine Interface | The International Conference on Digital Image & Signal Processing (DISP'19) | Gated Recurrent Unit (GRU), Generative Adversarial Network (GAN), Zero-Shot Learning (ZSL) | Emotion Classification, Brain-Machine Interface, Machine Learning. | OneR, Bayes Network, Information Gain, and Symmetrical Uncertainty. | Overall accuracy of around 97.89%, outperforming the current state of the art by 2.99 percentage points. The best single classifier was a deep neural network with an accuracy of 94.89%. | Advantage is that the proposed approach can be generalized to various research fields of BCI. | Zero-Shot EEG classification method can be extended for untrained EEG classes as well. It has achieved an accuracy of 39.65% for the ten untrained EEG classes. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 5. | Rajeev Sharm Ram Bilas Pachori | Classification of epileptic seizures in EEG signals based on phase space representation of intrinsic mode functions | Fourth International Conference | Linear prediction (LP) error energy and fractional linear prediction (FLP) error | center of pressure (COP) signal analysis (Pachori et al., 2008, Pachori et al., 2009), gear fault diagnosis (Parey and Pachori, 2012, Parey and Pachori, 2014), speech signal analysis (Huang & Pan, 2006), electrocardiogram (ECG) signal analysis (Pal & Mitra, | Dataset used in this study is a benchmark dataset one can cipher what was said. | Shows the effectiveness of the proposed method for classification of epileptic seizure and seizure-free EEG signals. | | Features were extracted from PSR of IMFs of EEG signals. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | 2012), electromyogram (EMG) signal analysis (Xie & Wang, 2006) etc. | | | | |
| 6. | Wei-Long Zheng, Jia- Yi Zhu,Yong Peng, Bao-Liang Lu | EEG-Based emotion classification using deep belief networks | IEEE International Conference on Multimedia and Expo (2014) | Differential Entropy Feature Extraction,Classification with GLEM,DBN and HMM | Neuroscience, psychology and computer science | Some emotional movie clips were chosen which helped subjects to elicit their own emotion states.EEG was recorded using an ESI NeuroScan System at a sampling rate of 1000 Hz for dataset | High frequency-band (beta and gamma) features are more related to emotion recognition, which is consistent with previous work, proposing higher frequency brain activity reflecting emotional and cognitive processes | | To demonstrate the capacity of DBNs as a classification platform for multichannel EEG data. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 7. | NAN-YING LIANG, PARAMASIVAN SARATCHANDRAN, GUANG-BIN HUANG and NARASIMHAN SUNDARARAJANa | CLASSIFICATION OF MENTAL TASKS FROM EEG SIGNALS USING EXTREME LEARNING MACHINE | International Journal of Neural Systems | Backpropagation Neural Network (BPNN) classifier and also Support Vector Machines (SVMs). | EEG-based Pattern Recognition | Deap Dataset | Results show that ELM needs an order of magnitude less training time compared with SVMs and two orders of magnitude less compared with BPNN. | | The study showed that smoothing of the classifiers' outputs can significantly improve their classification accuracies. |
| 8. | Kang Li,Xiaoyi Li,Yuan Zhang, Aidong Zhang | Affective state recognition from EEG with deep belief networks | IEEE International Conference on Bioinformatics and Biomedicine(2013) | Deep Feature Extraction,Critical Channel Selection, | | DEAP Data set | Experiments on a real world data set validated that the proposed method significantly outperforms five baselines by 11.5% to 24.4%. | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 9. | Yuanfang Ren, Yan Wu | Convolutional deep belief networks for feature extraction of EEG signal | International Joint Conference on Neural Networks(2014) | Band power,multivariate adaptive autoregressive(MVAAR),common spatial pattern (CSP),independent component analysis (ICA) | | Dataset III in 2003 BCIC II,Iva in 2005 BCIC III,Dataset III in 2005 BCIC III | CDBN performs well in the feature learning of EEG signals especially when there is a large amount of unlabeled data. | | |
| 10 | Key Gregor Hartmann,Robin Tibor Schirrmeister,Tonio Ball | Generative adversarial networks for electroence paleographic (EEG) brain signals | Journal:ArXiv (2018) | WGANs with gradi-ent penalty (WGAN-GP),Inception score, Frechet inception distance and sliced Wasserstein distance | | The dataset was collected by recording the hand movement of subjects using 128-electrode EEG System. | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 11 | Nhan Duy Truong ,Levin Kuhlmann,Mohammad Reza Bonyadi,Damein Querlioz,Lupng Zhou, Omid Kavehei | Epileptic Seizure Forecasting With Generative Adversarial Networks | Journal:IEEE Access (2019) | Short-Time Fourier Transform (STFT),Generative Neural Network (GAN) architecture with three de-convolution layers,Area under the receiver operating characteristics curve (AUC) | | CHB-MIT,Freiburg Hospital dataset,EPILEPSIAE, | Feature extrac-tion for seizure forecasting can be done using unsupervised deep learning or GAN particularly.Female patients under thirty years<br><br>old with focal aware seizure type would benefit the most from such a seizure forecasting system. | | |

***Methodology and FlowChart :***

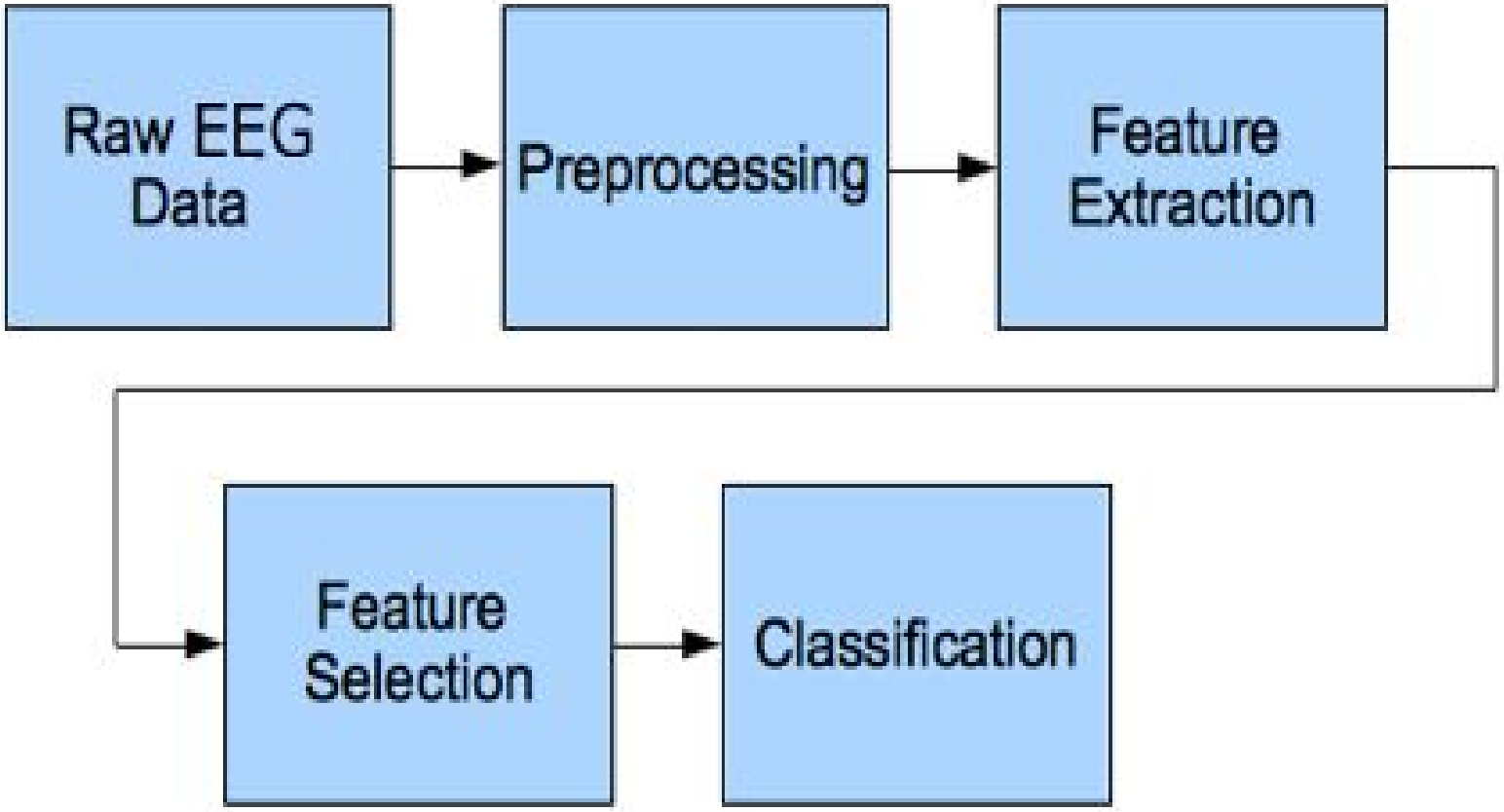



*image 1:* ***Flowchart***

***Activity Chart :***

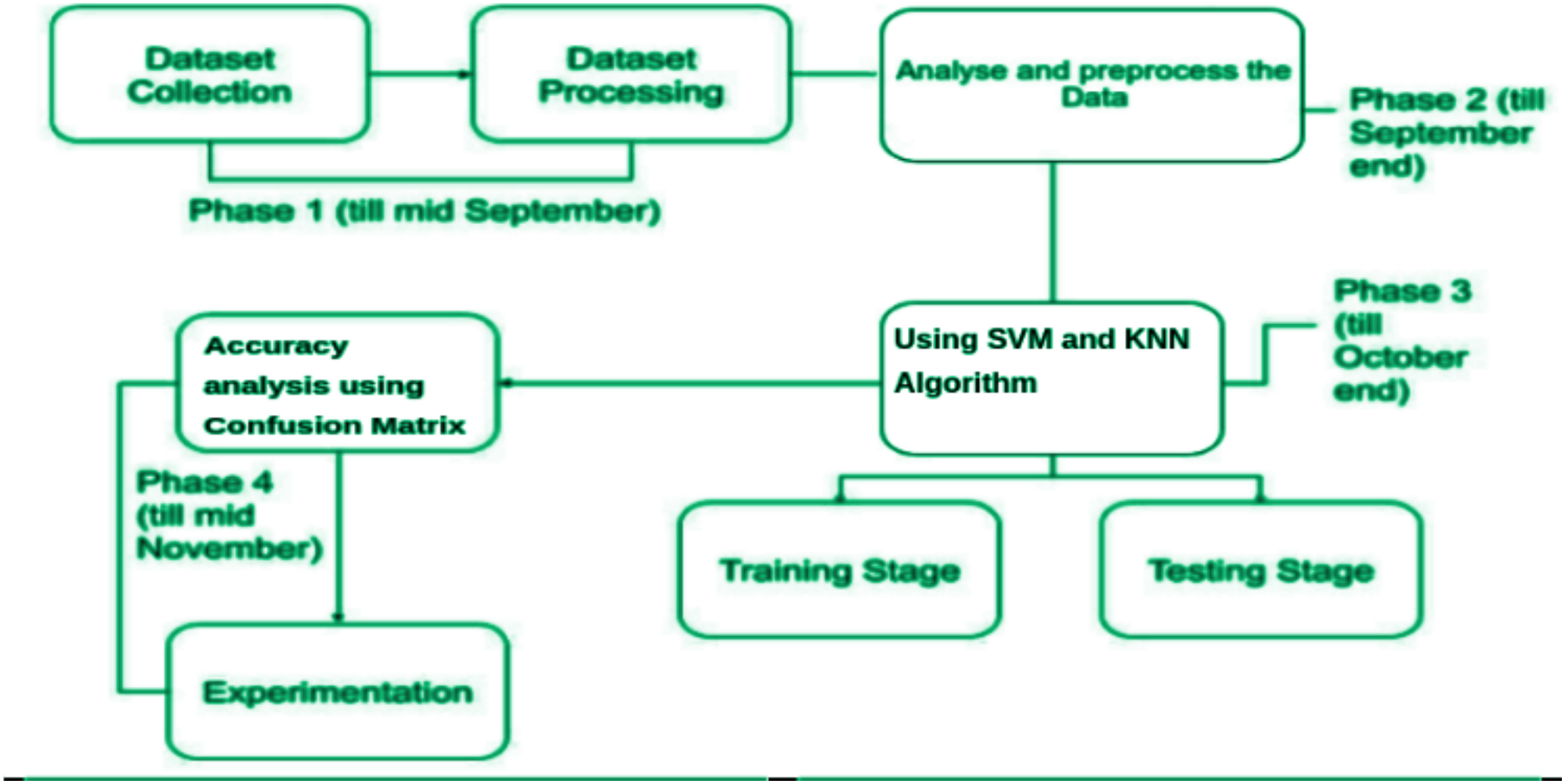



*image 2*

## *Software and Hardware Requirements :*

### *1. Experimental Dataset :*

The main dataset to be used is :

**1) EEG-Feature -Generation**

https://github.com/jordan-bird/eeg-feature-generation/tree/master/dataset/original_data

***Timestamps :- is that instance of me where the data is collected***

***TP9,AF7,AF8,TP10, :- These are the four sensors used up for collecting dataset***

***Right Aux :- RAW brainwaves for the auxiliary USB sensor (MU-02 and MU-03 only)***

```
timestamps,TP9,AF7,AF8,TP10,Right AUX
1533222559.839,59.105,28.320,15.137,12.207,54.199
1533222559.843,62.012,30.273,43.945,11.719,79.102
1533222559.847,44.922,30.273,-97.656,11.230,32.715
1533222559.851,28.809,27.832,-110.352,9.277,29.785
1533222559.855,36.156,28.809,-73.242,11.230,50.781
1533222559.859,57.617,36.133,-17.090,16.113,37.109
1533222559.862,74.219,38.574,42.480,27.832,20.020
1533222559.866,57.129,33.691,-30.273,34.180,28.320
1533222559.870,23.949,29.785,-101.074,13.184,64.941
1533222559.874,20.020,33.203,-92.285,9.277,40.527
1533222559.878,41.504,33.203,-24.902,15.625,-0.488
1533222559.882,51.758,29.785,68.359,14.160,37.598
1533222559.886,47.852,34.180,-29.297,21.973,55.176
1533222559.890,34.180,36.133,-122.070,17.578,59.570
```

*Image 3*

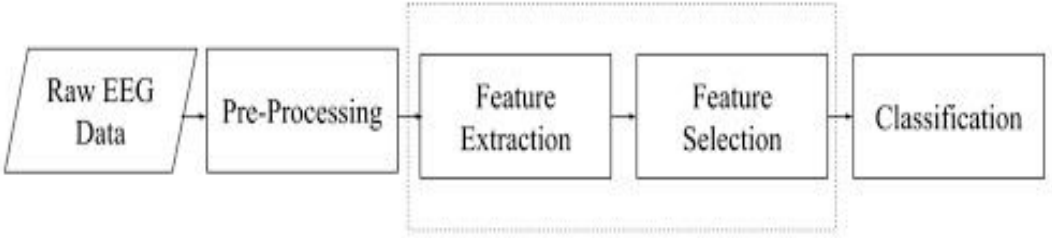
### *2. Language / tools for implementation :*

The language of choice for us will be python due to the rich support of data science libraries available. Scikit and scipy are some examples of such popular libraries.

## *Implementation* :

### 1. Feature Preprocessing :

Firstly, an available training set of EEG signals is preprocessed. The data is assumed to contain the time series related to one or more electrodes, within a given experimental time frame, labelled in terms of **three distinct mental states (relaxed, concentrating, and neutral)** that the subjects were keeping during data collection . From these signals a number of statistical features are extracted , resulting in a high dimensional attribute space.




### 2. Feature Extraction :

Due to the temporal, auto-correlated nature of the EEG waves, single-point features cannot generally provide enough information for good rules to be generated by machine learning models. In this work we follow the approach of extracting statistical features based on sliding time windows [3, 4]. More specifically, the

EEG signal is divided into a sequence of windows of length one second, with consecutive windows overlapping by 0.5 seconds, e.g., [(0s − 1s), [0.5s − 1.5s), [1s − 2s), . . . ]).

**The following statistical features were generated for each time window and for this we have implemented the following functions:**

## Functions used :

- feature_mean : Takes in the matrix and returns the mean value of each signal for the full time window
- Feature_mean_d : Takes in two matrices denoting first half and second half of any time slice and computes the change in the means (backward difference) of all signals between the first and second half-windows, i.e mean(arg1)-mean(arg2)
- feature_mean_q : Takes in four matrices denoting four quarters of any time slice and computes the mean values of each signal for each quarter-window, plus the paired differences of means of each signal for the quarter-windows i.e Σ(feature_mean(arg(i)) - (feature_mean(arg(j)) for all 1<=i,j<=4
- feature_stddev : Takes in the matrix and returns the standard deviation of each signal for the full time window
- feature_stddev_d :Takes in two matrices denoting first half and second half of any time slice and computes the change in standard deviation (backward difference) of all signals between the first and second half-windows, i.e std(arg1)-std(arg2)
- feature_moments : Takes in the matrix and computes the 3rd and 4th standardised moments about the mean (i.e., skewness and kurtosis) of each signal, for the full time window.
- feature_max : Takes in the matrix and returns the max value of each signal for the full time window
- Feature_max_d : Takes in two matrices denoting first half and second half of any time slice and computes the change in the max value (backward difference) of all signals between the first and second half-windows, i.e mean(arg1)-mean(arg2).
- feature_max_q : it will take four quarter windows as input,computes the maximum value for each quarter window and then paired the difference of maximum values for the quarter-windows.
- feature_min : Returns the minimum value of each signal for the full time window.
- feature_min_d: Computes the change in min values of all signals between the first and second half-windows, min(h2) - min(h1)

- feature_min_q: Computes the min values for each quarter-window and paired differences of min values for the quarter-windows.
- feature_covariance_matrix: Computes the lower triangular elements of the covariance matrix of the signals because the covariance matrix is symmetric.
- feature_eigenvalues : Computes the eigenvalues of the covariance matrix.
- feature_logcov: Computes the matrix logarithm of the covariance matrix
- calc_feature_vector: Calculates all previously defined features and concatenates everything into a single feature vector.
- generate_feature_vectors_from_samples: Reads data from CSV file and extracts statistical features for each time window of width "period".
- Gen_training_matrix: This is the main function which calls on all the calculating features and generate the final matrix

### 3. Feature Selection:

Based on all the features we get, we select only the relevant ones. Scikit-learn has made it pretty much easy for us to make the feature selection. There are a lot of ways in which we can think of feature selection, but most feature selection methods can be divided into three major buckets

- ***Filter based:*** We specify some metric and based on that filter features. An example of such a metric could be correlation/chi-square.
- ***Wrapper-based:*** Wrapper methods consider the selection of a set of features as a search problem. Example: Recursive Feature Elimination
- ***Embedded:*** Embedded methods use algorithms that have built-in feature selection methods. For instance, Lasso and RF have their own feature selection methods.

So we used 5 methods for feature selection and we then have chosen the one with highest accuracy :

1) **Pearson Correlation :**This is a filter-based method.We check the ***absolute value of the Pearson's correlation*** between the target and numerical features in our dataset. We keep the top n features based on this criterion.

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2 \sum(y-\bar{y})^2}}$$

```
def cor_selector(X, y, num_feats):
    cor_list = []
    feature_name = X.columns.tolist()
    # calculate the correlation with y for each feature
    for i in X.columns.tolist():
        cor = np.corrcoef(X[i], y)[0, 1]
        cor_list.append(cor)
    # replace NaN with 0
    cor_list = [0 if np.isnan(i) else i for i in cor_list]
    # feature name
    cor_feature = X.iloc[:, np.argsort(
        np.abs(cor_list))[-num_feats:]].columns.tolist()
    # feature selection? 0 for not select, 1 for select
    cor_support = [True if i in cor_feature else False for i in feature_name]
    return cor_support, cor_feature
```

2) **Chi-Squared :** This is another filter-based method.In this method, we calculate the chi-square metric between the target and the numerical variable and only select the variable with the maximum chi-squared values. We therefore calculate the chi-squared value. To do this, we first find out the values we would expect to be falling in each bucket if there was indeed independence between the two categorical variables. We then multiply the row sum and the column sum for each cell and divide it by total observations. We use the following formula :

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import chi2
```

```
from sklearn.feature_selection import SelectKBest
chi_selector = SelectKBest(chi2, k=num_feats)
chi_selector.fit(X_norm, y)
chi_support = chi_selector.get_support()
chi_feature = X.loc[:, chi_support].columns.tolist()
```

3) **Recursive Feature Elimination :** This is a wrapper based method. As I said before, wrapper methods consider the selection of a set of features as a search problem. The goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and the importance of each feature is obtained either through a `coef_` attribute or through a `feature_importances_` attribute. Then, the least important features are pruned from the current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE
rfe_selector = RFE(estimator=LogisticRegression(),
                   n_features_to_select=num_feats, step=30, verbose=5)
rfe_selector.fit(X_norm, y)
rfe_support = rfe_selector.get_support()
rfe_feature = X.loc[:, rfe_support].columns.tolist()
```

4) **Tree-based: SelectFromModel :** This is an Embedded method.We can also use RandomForest to select features based on feature importance.We calculate feature importance using node impurities in each decision tree. In Random forest, the final feature importance is the average of all decision tree feature importance.

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
embeded_rf_selector = SelectFromModel(
   RandomForestClassifier(n_estimators=100), max_features=num_feats)
embeded_rf_selector.fit(X, y)

embeded_rf_support = embeded_rf_selector.get_support()
embeded_rf_feature = X.loc[:, embeded_rf_support].columns.tolist()
```

5) **LightGBM :** We have also used a LightGBM as it has a feature_importances_ attribute.

```
from lightgbm import LGBMClassifier
from sklearn.feature_selection import SelectFromModel

lgbc = LGBMClassifier(n_estimators=500, learning_rate=0.05, num_leaves=32, colsample_bytree=0.2,
                      reg_alpha=3, reg_lambda=1, min_split_gain=0.01, min_child_weight=40)

embeded_lgb_selector = SelectFromModel(lgbc, max_features=num_feats)
embeded_lgb_selector.fit(X, y)

embeded_lgb_support = embeded_lgb_selector.get_support()
embeded_lgb_feature = X.loc[:, embeded_lgb_support].columns.tolist()
```

## 4. 1. Support vector machine

**Fundamental reason for using the support vector machine is to classify/predict , .i.e. Firstly using the training data (combination of sample input and output ) , in order to train the svm and then , for some set of input , get the output predicted technically also known as the supervised learning.**

**The SVM is used to get classification among the different classes among the output and thus is the discrete classification and not the continuous one.** Saying that there are **n features** in the input and then the main objective of the **svm algorithm is to find a plane in n dimensional that classifies the data points , which are themself n dimensional. The classification is taken as either the data point belongs inside or outside of the plane.**

In order to separate the classes of data , there can be many possible different planes that can be used in doing so. Task comes to get that plane that has the maximum margin , so the large margin classifier, that is  distance between data points of both classes should be maximum w.r.t the plane .As we are maximizing marginal distance so as to get some extra reinforcement so that on the future data points we will be getting less error. And thus will be classified with more amount of confidence.

**The Scikit learn library provides the implementation for the same and just call the related functions to implement the SVM model as it is even more reliable and accurate as well as implementing ourselves.**

```
eeg_svc = SVC(C=1.0, kernel="linear")
# print(type(eeg_svc))

eeg_svc.fit(X_train, y_train.ravel())
print("Training Accuracy:", (eeg_svc.score(X_train, y_train.ravel()))*100, "%")

eeg_svc.predict(X_test)
print("Test Accuracy:", (eeg_svc.score(X_test, y_test.ravel()))*100, "%")
matrix = plot_confusion_matrix(eeg_svc,
                               X_test,
                               y_test,
                               cmap=plt.cm.Blues,
                               normalize='true')
plt.title('Confusion matrix for our classifier')

plt.show()
```

**After the generation of the out.csv and getting on the features from it , using the already mentioned feature extraction algorithms , the task comes on to train and then predict the state of an individual using the same features , which is quite basic when using the internal implementation.**
X_train -> it is the set of all features for the whole of the training data which gets the corresponding output as the state of an individual.

Y_train -> it has the label , the answer for the same training data.

X_test -> it is the set of all features for whole of the testing data, being used to get the output predicted

Y_test -> it has the label , the answer for the same testing data.

**Distribution of the data**

We have used 70% of the dataset inorder to train the model and the remaining 30% of the data available to us , to get the testing accuracy , as done in the base paper also.

**Confusion matrix represents the ratio of number of predictions in a particular class w.r.t total value which have this class as true . This can also be verified by checking as summation of values of each row comes out to be 1.**

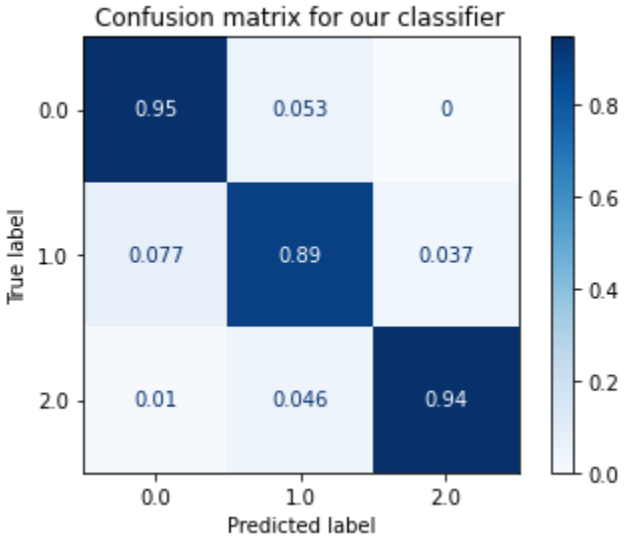



*Image 4*

**The dark blue background (.i.e. The values on the main diagonal of the confusion matrix ) represents the correct prediction of our implemented svm whereas others represent the wrong prediction.**

## 2. KNN

K- Nearest Neighbors is a Supervised machine learning algorithm as the target variable is known Non parametric as it does not make an assumption about the underlying data distribution pattern Lazy algorithm as KNN does not have a training step. All data points will be used only at the time of prediction. With no training step, prediction step is costly. An eager learner algorithm eagerly learns during the training step. It is used for both Classification and Regression. It uses feature similarity to predict the cluster that the new point will fall into.

## *Comparison and Result :*

### Results for SVM

1. **Pearson Correlation :**

```
---------------------------------------- Pearson Correlation ----
(2479, 850)
2479
Training Accuracy: 100.0 %
Test Accuracy: 90.34289713086075 %
          TP      FP       TN       FN

0.0       417      54       913      45
1.0       424      59       869      77
2.0       450      25       938      16

Classification Report

                precision     recall   f1-score    support

          0.0        0.90       0.89       0.89        471
          1.0        0.85       0.88       0.86        483
          2.0        0.97       0.95       0.96        475

     accuracy                             0.90       1429
    macro avg        0.90       0.90       0.90       1429
weighted avg        0.90       0.90       0.90       1429
```

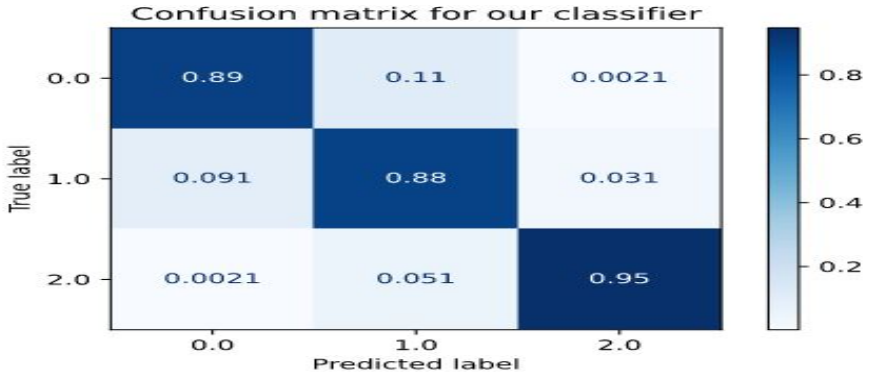



*Image 5*

2. **Chi - Squared :**

```
---------------------------------------- Chi-Squared ------------------------
(2479, 850)
2479
Training Accuracy: 100.0 %
Test Accuracy: 91.46256123163052 %
          TP     FP      TN      FN

0.0      421     50      919     39
1.0      428     55      881     65
2.0      458     17      936     18

Classification Report

              precision    recall  f1-score   support

         0.0       0.92      0.89      0.90       471
         1.0       0.87      0.89      0.88       483
         2.0       0.96      0.96      0.96       475

    accuracy                           0.91      1429
   macro avg       0.92      0.91      0.91      1429
weighted avg       0.91      0.91      0.91      1429
```

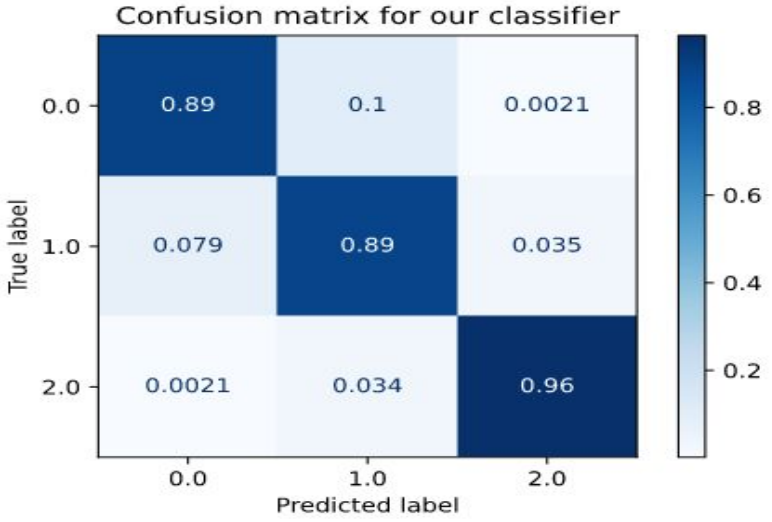



*Image 6*

### 3. Recursive Feature Elimination :

```
------------------------------------ Recursive Feature Elimination -----------------
(2479, 850)
2479
Training Accuracy: 100.0 %
Test Accuracy: 90.76277116864941 %
         TP     FP      TN      FN

0.0      428     43      909     49
1.0      417     66      884     62
2.0      452     23      933     21

Classification Report

              precision    recall  f1-score   support

         0.0       0.90      0.91      0.90       471
         1.0       0.87      0.86      0.87       483
         2.0       0.96      0.95      0.95       475

    accuracy                           0.91      1429
   macro avg       0.91      0.91      0.91      1429
weighted avg       0.91      0.91      0.91      1429
```

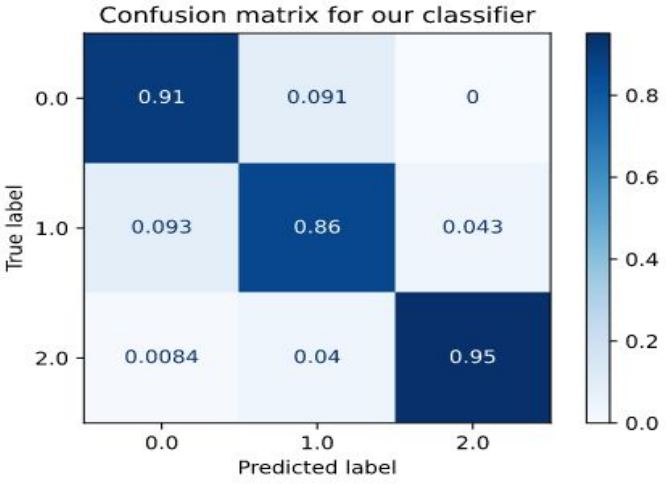



*Image 7*

### 4. Random Forest :

```
---------------------------------------- Random Forest -----------
(2479, 206)
2479
Training Accuracy: 98.4747378455672 %
Test Accuracy: 90.83275017494752 %
          TP      FP       TN       FN

0.0       436      35       909      49
1.0       417      66       890      56
2.0       445      30       928      26

Classification Report

              precision    recall  f1-score   support

         0.0       0.90      0.93      0.91       471
         1.0       0.88      0.86      0.87       483
         2.0       0.94      0.94      0.94       475

    accuracy                           0.91      1429
   macro avg       0.91      0.91      0.91      1429
weighted avg       0.91      0.91      0.91      1429
```

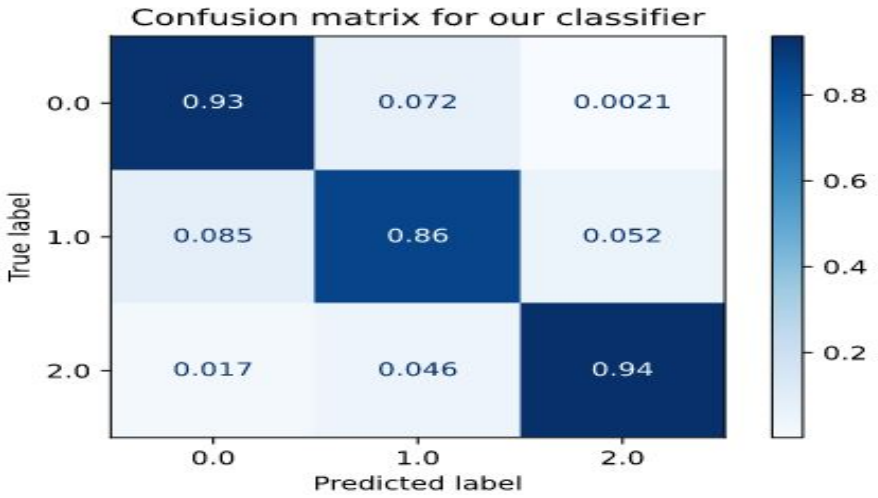



*Image 8*

5. **Light GBM :**

```
(2479, 199)
2479
Training Accuracy: 97.04480457578646 %
Test Accuracy: 88.52344296710987 %
```

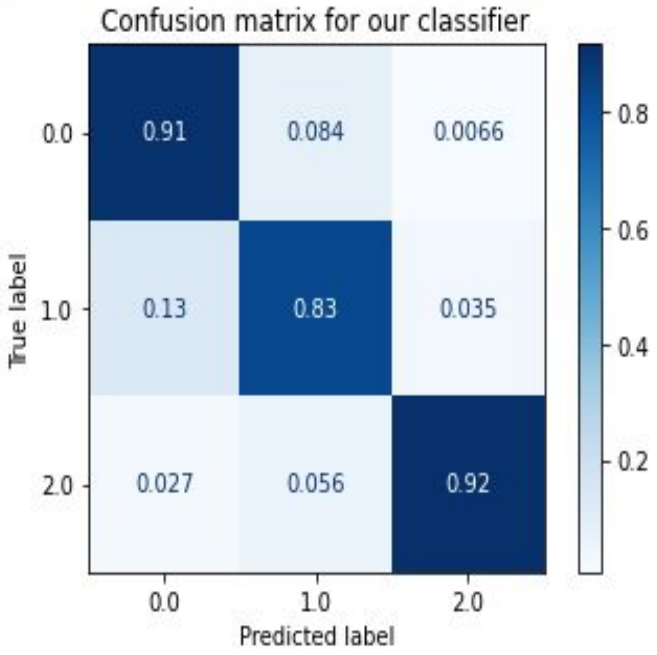



*Image 9*

**Results for KNN:**

1. **Pearson Correlation :**

```
------------------------------------ Pearson Correlation --------------
Test Accuracy: 95.16129032258065 %
         TP     FP      TN      FN

0.0      232     14      480     18
1.0      218     22      491     13
2.0      258     0       481     5

Classification Report

              precision    recall  f1-score   support

         0.0       0.93      0.94      0.94       246
         1.0       0.94      0.91      0.93       240
         2.0       0.98      1.00      0.99       258

    accuracy                           0.95       744
   macro avg       0.95      0.95      0.95       744
weighted avg       0.95      0.95      0.95       744
```

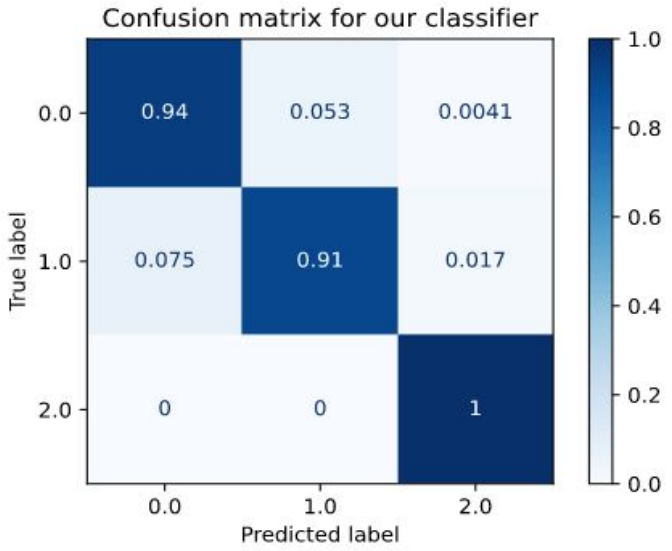



*Image 10*

**2. Chi - Squared :**

```
---------------------------------------- Chi-Squared -----------------------
Test Accuracy: 90.18817204301075 %
          TP      FP       TN       FN

0.0       224      20       458      42
1.0       208      50       464      22
2.0       239      3        493      9

Classification Report

              precision    recall  f1-score   support

         0.0       0.84      0.92      0.88       244
         1.0       0.90      0.81      0.85       258
         2.0       0.96      0.99      0.98       242

    accuracy                           0.90       744
   macro avg       0.90      0.90      0.90       744
weighted avg       0.90      0.90      0.90       744
```

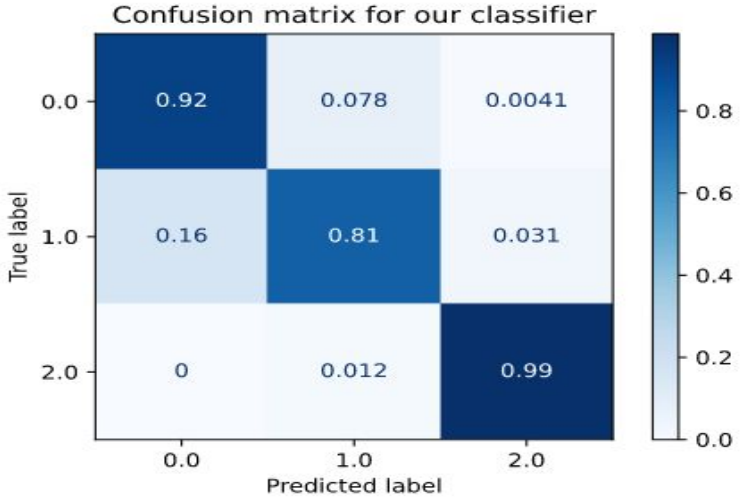



*Image 11*

**3. Recursive Feature Elimination :**

```
------------------------------------ Recursive Feature Elimination -----------------
Test Accuracy: 90.18817204301075 %
         TP     FP      TN      FN

0.0      217     14      462     51
1.0      206     57      466     15
2.0      248     2       487     7

Classification Report

              precision    recall  f1-score   support

         0.0       0.81      0.94      0.87       231
         1.0       0.93      0.78      0.85       263
         2.0       0.97      0.99      0.98       250

    accuracy                           0.90       744
   macro avg       0.90      0.90      0.90       744
weighted avg       0.91      0.90      0.90       744
```

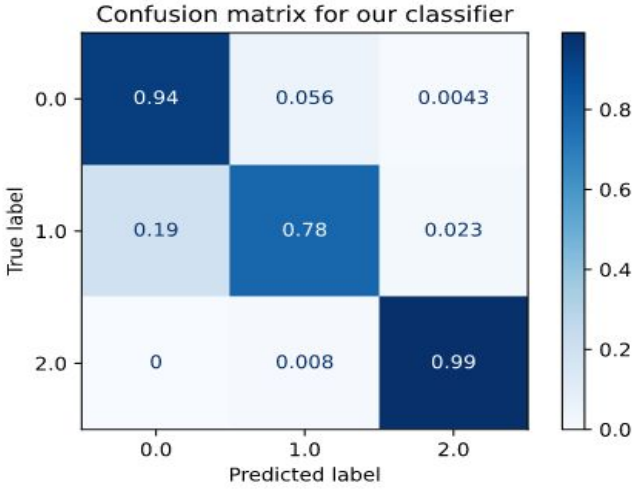



*Image 12*

## 4. Random Forest :

```
------------------------------------ Random Forest --------------------------
Test Accuracy: 95.16129032258065 %
         TP     FP      TN      FN

0.0      244     10      474     16
1.0      222     24      487     11
2.0      242     2       491     9

Classification Report

              precision    recall  f1-score   support

         0.0       0.94      0.96      0.95       254
         1.0       0.95      0.90      0.93       246
         2.0       0.96      0.99      0.98       244

    accuracy                           0.95       744
   macro avg       0.95      0.95      0.95       744
weighted avg       0.95      0.95      0.95       744
```

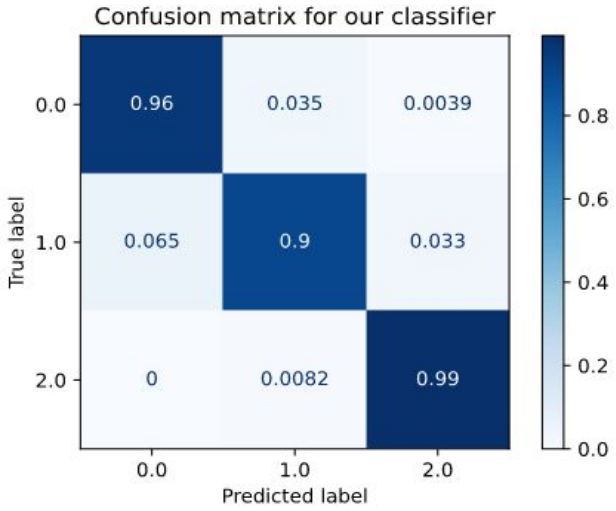



*Image 13*

## 5. Light GBM :

```
------------------------------------ LightGBM -----------------------------
Test Accuracy: 95.96774193548387 %
         TP     FP      TN      FN

0.0      237     6       482     19
1.0      224     22      490     8
2.0      253     2       486     3

Classification Report

              precision    recall  f1-score   support

         0.0       0.93      0.98      0.95       243
         1.0       0.97      0.91      0.94       246
         2.0       0.99      0.99      0.99       255

    accuracy                           0.96       744
   macro avg       0.96      0.96      0.96       744
weighted avg       0.96      0.96      0.96       744
```

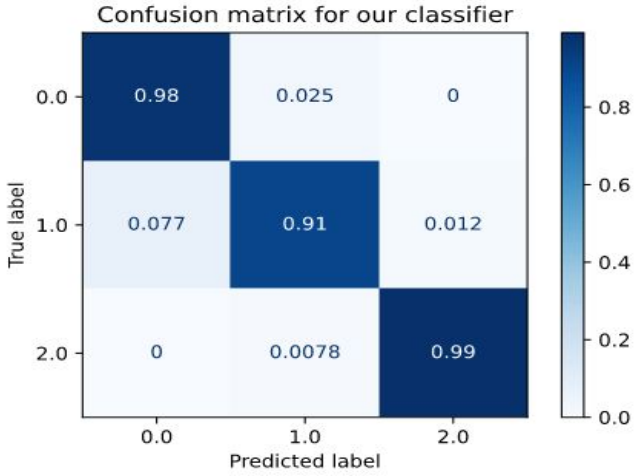



*Image 14*

**The above efforts and the attached proof shows the accuracy for both training and testing data from the validation split. As we get the highest frequency in Recursive Feature Elimination, we have then opted this method for feature selection.**

| Study | Method | Validation | Focus | Accuracy |
|---|---|---|---|---|
| This study | Inf. Gain Selection, CNN | 70/30 Split | Accuracy | 89.38% [86.94, 91.50] |
| [3] | OneR Selection, Random Forest | 10-fold | Accuracy | 87.2% [85.7, 88.6] |
| [35] | Evol. Selection, DEvoMLP | 5-fold | Accuracy, Resource Usage | 79.8% [78.1, 81.5] |

Fig from the base paper

| FeatureSelection\Classifier | SVM (%) | KNN (%) |
|---|---|---|
| | | |
| Pearson Correlation | 91.39258223 | 93.01075269 |
| Chi-Squared | 91.81245626 | 92.06989247 |
| Recursive Feature Elimination | 92.51224633 | 90.99462366 |
| Random Forest | 91.1126662 | 93.9516129 |
| LightGBM | 88.52344297 | 95.56451613 |

Comparison Table for SVM and KNN for different Feature Selection Algorithms.

**Best accuracy of the base paper :- 89.38%, and what we have achieved is 92.51% for SVM and 95.56% for KNN which is quite an amazing thing.**

## *Conclusion :*

In this project , a new approach for classification of EEG signals has been presented, based on the sequential application of statistical feature extraction and selection, normalisation and subsequent projection of the selected features and classification based on a SVM .
**The results obtained for this method have been shown to be very competitive with the best known results to date for the available dataset and also the accuracy by using SVM is better than the one done in the research paper. Therefore SVM is the best method for the given dataset.**

## *References:*


[1] Mental Emotional Sentiment Classification with an EEG-based Brain-machine Interface by Jordan J.Bird , Anikó Ekárt , Christopher D. Buckingham and Diego R. Faria

[2] DEAP: A Database for Emotion Analysis Using Physiological Signals Sander Koelstra, Student Member, IEEE, Christian Mühl, Mohammad Soleymani, Student Member, IEEE, Jong-Seok Lee, Member, IEEE, Ashkan Yazdani, Touradj Ebrahimi, Member, IEEE, Thierry Pun, Member, IEEE, Anton Nijholt, Member, IEEE, and Ioannis (Yiannis) Patras, Senior Member, IEEE

[3] A Deep Evolutionary Approach to Bioinspired Classifier Optimisation for Brain-Machine Interaction Jordan J. Bird , Diego R. Faria, Luis J. Manso, Anikó Ekárt, and Christopher D. Buckingham

[4] Classification of EEG Signals Based on Image Representation of Statistical Features Jodie Ashford * , Jordan J. Bird * , Felipe Campelo, and Diego R. Faria

[5] Amin HU, Mumtaz W, Subhani AR, Saad MNM and Malik AS (2017) Classification of EEG Signals Based on Pattern Recognition Approach. *Front. Comput. Neurosci*. 11:103. doi: 10.3389/fncom.2017.00103

[6]Amorim, P., Moraes, T., Fazanaro, D., Silva, J. and Pedrini, H., 2017. Electroencephalogram signal classification based on shearlet and contourlet transforms. Expert Systems with Applications, 67, pp.140-147.

[7] Luo Tian-jian, Fan Yachao, Chen Lifei, Guo Gongde, Zhou Changle. (2020). EEG Signal Reconstruction Using a Generative Adversarial Network With Wasserstein Distance and Temporal-Spatial-Frequency Loss. Frontiers in Neuroinformatics, doi: 10.3389/fninf.2020.00015

[8] S. Hwang, K. Hong, G. Son and H. Byun, "EZSL-GAN: EEG-based Zero-Shot Learning approach using a Generative Adversarial Network," 2019 7th International Winter Conference on Brain-Computer Interface (BCI), Gangwon, Korea (South), 2019, pp. 1-4, doi: 10.1109/IWW-BCI.2019.8737322.

[9] Classification of epileptic seizures in EEG signals based on phase space representation of intrinsic mode functions
Author links open overlay panel RajeevSharmaRam BilasPachori

[10] W. Zheng, J. Zhu, Y. Peng and B. Lu, "EEG-based emotion classification using deep belief networks," 2014 IEEE International Conference on Multimedia and Expo (ICME), Chengdu, 2014, pp. 1-6, doi: 10.1109/ICME.2014.6890166.

[11]Classification Of Mental Tasks From Eeg Signals Using Extreme Learning Machine Nan-ying Liang, Paramasivan Saratchandran, Guang-bin Huang And Narasimhan Sundararajan

[12] K. Li, X. Li, Y. Zhang and A. Zhang, "Affective state recognition from EEG with deep belief networks," 2013 IEEE International Conference on Bioinformatics and Biomedicine, Shanghai, 2013, pp. 305-310, doi: 10.1109/BIBM.2013.6732507.

[13] Y. Ren and Y. Wu, "Convolutional deep belief networks for feature extraction of EEG signal," 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, 2014, pp. 2850-2853, doi: 10.1109/IJCNN.2014.6889383.

[14] Hartmann, Kay Gregor et al. “EEG-GAN: Generative adversarial networks for electroencephalographic (EEG) brain signals.” *ArXiv* abs/1806.01875 (2018): n. pag.

[15] N. D. Truong, L. Kuhlmann, M. R. Bonyadi, D. Querlioz, L. Zhou and O. Kavehei, "Epileptic Seizure Forecasting With Generative Adversarial Networks," in IEEE Access, vol. 7, pp. 143999-144009, 2019, doi: 10.1109/ACCESS.2019.2944691.

**Reference Links used for Code Implementation and understanding of the technicalities:**

https://github.com/nadzeri/Realtime-EEG-Based-Emotion-Recognition

https://github.com/Raghav714/EEG-Emotion-classification

https://github.com/Daisybiubiubiu/EEG-Emotion-Recognition

https://github.com/nasoboleva/EEG-Emotion-Recognition

https://github.com/jordan-bird/eeg-feature-generation

https://github.com/anumitgarg/Emotion-Classification-using-EEG

https://github.com/rishirdua/emotion-classification

https://github.com/Piyush-Bhardwaj/EEG-based-emotion-analysis-using-DEAP-dataset-for-Supervised-Machine-Learning

https://github.com/mahesh147/Support-Vector-Machine

https://towardsdatascience.com/andrew-ngs-machine-learning-course-in-python-support-vector-machines-435fc34b7bf9

https://medium.com/data-science-group-iitr/support-vector-machines-svm-unraveled-e0e7e3ccd49b

https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

https://github.com/Piyush-Bhardwaj/EEG-based-emotion-analysis-using-DEAP-dataset-for-Supervised-Machine-Learning

https://github.com/robi56/emotion_classification

https://github.com/junmoan/eeg-feeling-emotions-LSTM

https://github.com/Mister5ive/emotion_classification

https://github.com/tongdaxu/EEG_Emotion_Classifier_DEAP

https://github.com/dweidai/DEAP-JRP-Emotion-Classification

https://github.com/yerzhan7orazayev/EEG-based-Emotion-Classification

https://github.com/PratikshyaM/Emotion-Recognition-From-EEG-brain-waves-