Identification of Major Psychiatric Disorder from Resting State Electroencephalography (EEG) Using Machine Learning Approach

Group members:
Ayush Koirala (st122802)
Biplav Regmi (st123439)
Sagun Rupakheti (st123431)
Sandhya Lamichhane (st123497)
Sapna Thapa (st123473)
Sirikit Joshi (st123492)

Outline

01.

Introduction

Introduction

- Psychiatric disorder refers to a broad range of problems that disturb a person’s thoughts, feeling, behavior or mood.
- Electroencephalogram (EEG) is a test that measures electrical activity in the brain using small, metal discs (electrodes) attached to the scalp.
- There are different types of psychiatric disorders of which we have worked with 6 main disorders (schizophrenia, mood disorder, anxiety disorder, obsessive-compulsive disorder, trauma and stress-related disorder) and a healthy control set.

Motivation

- Mental health is one of the most pressing issues of our time.
- Researchers have estimated that **1/4** of us will experience a mental health problem in our lifetime and **1/6** report a common mental health problem such anxiety or depression each week
- After the catastrophic pandemic period, it was reported that the number of adults experiencing symptoms of depression has doubled whilst the number of people able to access both adult and child mental health had decreased.
- Hunt for the research papers started

Motivation

- After understanding how various ML techniques are applied on such research works, we looked for the dataset that aligned well with our plan.
- Finalized to work on the research papers by Park, S. M. (2021, August 16). EEG machine learning. Retrieved from osf.io/8bsvr - **Identification of Major Psychiatric Disorder from Resting State Electroencephalography (EEG) Using Machine Learning Approach**

02.

Data Study

Data Study

- This study aims to examine the efficiency of different machine learning models when analyzing EEG to detect and compare major psychiatric disorders.
- Research has found that symptoms-focused diagnosis limits the focus of treatment to symptom relief only. Therefore, data-driven approaches to study neural mechanisms are being used as a diagnostic aid.
- Advances in data and computational science are rapidly changing and use of ML here assesses the performance of predictions on unseen data thereby, providing individualized information and yielding results that may have high level of clinical translation.

Data Study

- The data set consists of medical records, intelligent quotient (IQ) scores from psychological assessments, and quantitative EEG (QEEG) at resting state assessment
- Data size: total sample is 945 (850-psychiatric disorder patients; 95-healthy control)
- Inclusion criteria:
  - Age of subjects - 18 to 70 years
  - Diagnosis which fall into 6 main disorders (one healthy control) and 9 specific disorders

03.

Data Science

EDA

Exploratory Data Analysis

DATA FEATURES

Demographic Data

- Age
- Sex
- Education
- IQ

Power Spectral Density (PSD)

- Absolute Power
- No. of Features = 19
- Bands = 6

Functional Connectivity (FC)

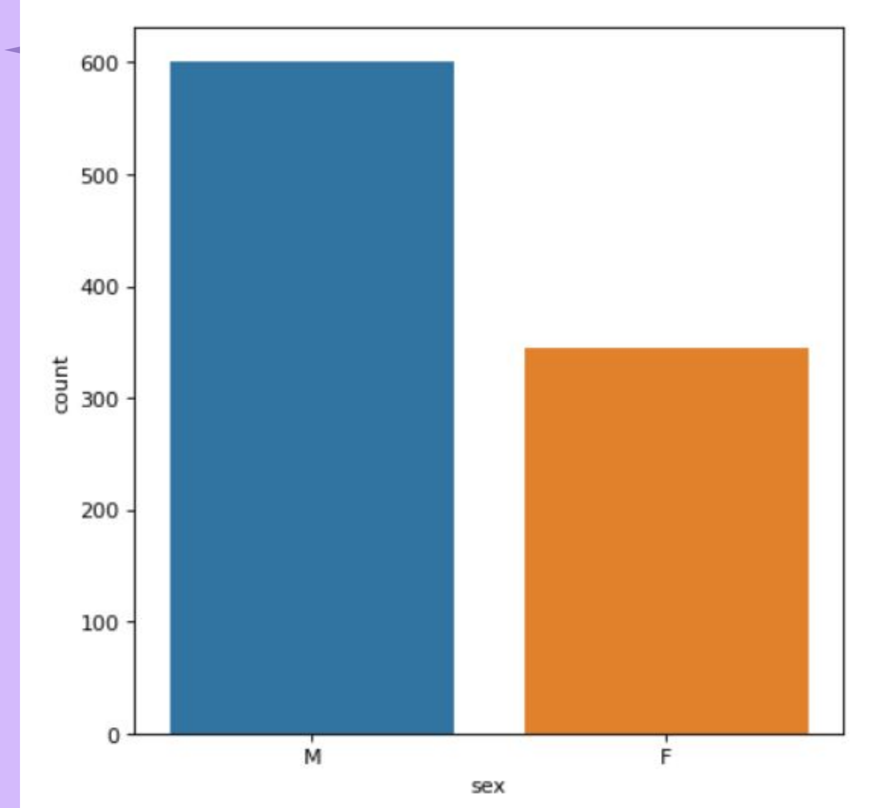
- Coherence
- Combination of channels (19C2)
- No. of Features = 171
- Bands = 6

Data Exploration

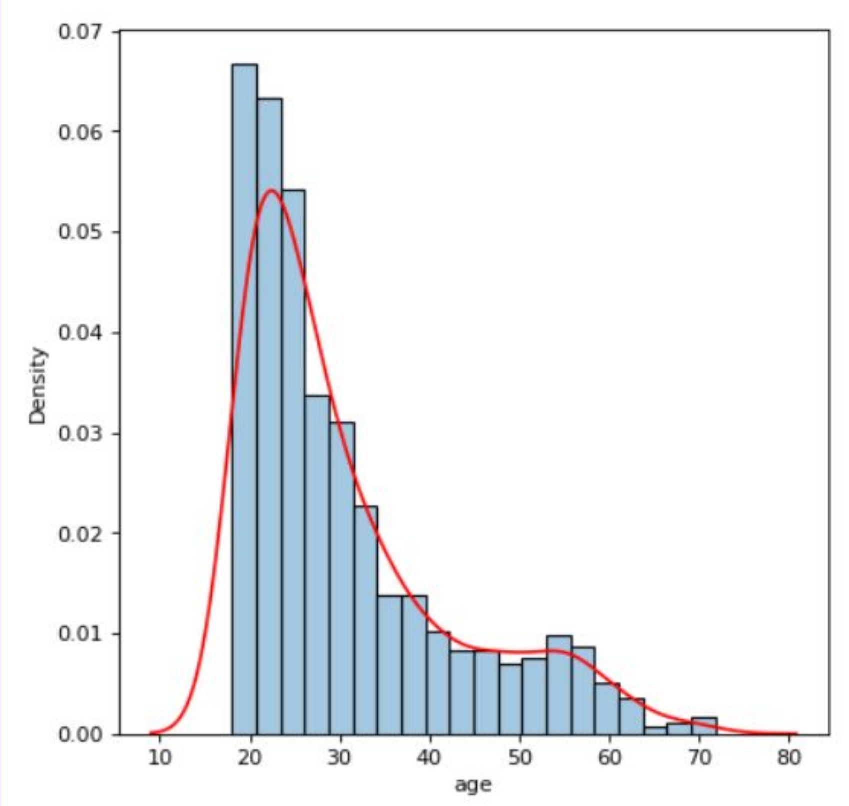
	no.	sex	age	eeg.date	education	IQ	main.disorder	specific.disorder	AB.A.delta.a.FP1	AB.A.delta.b.FP2	...	COH.F.gamma.o.Pz.p.P4
0	1	M	57.0	2012.8.30	NaN	NaN	Addictive disorder	Alcohol use disorder	35.998557	21.717375	...	55.989192
1	2	M	37.0	2012.9.6	6.0	120.0	Addictive disorder	Alcohol use disorder	13.425118	11.002916	...	45.595619
2	3	M	32.0	2012.9.10	16.0	113.0	Addictive disorder	Alcohol use disorder	29.941780	27.544684	...	99.475453
3	4	M	35.0	2012.10.8	18.0	126.0	Addictive disorder	Alcohol use disorder	21.496226	21.846832	...	59.986561
4	5	M	36.0	2012.10.18	16.0	112.0	Addictive disorder	Alcohol use disorder	37.775667	33.607679	...	61.462720

5 rows × 1149 columns

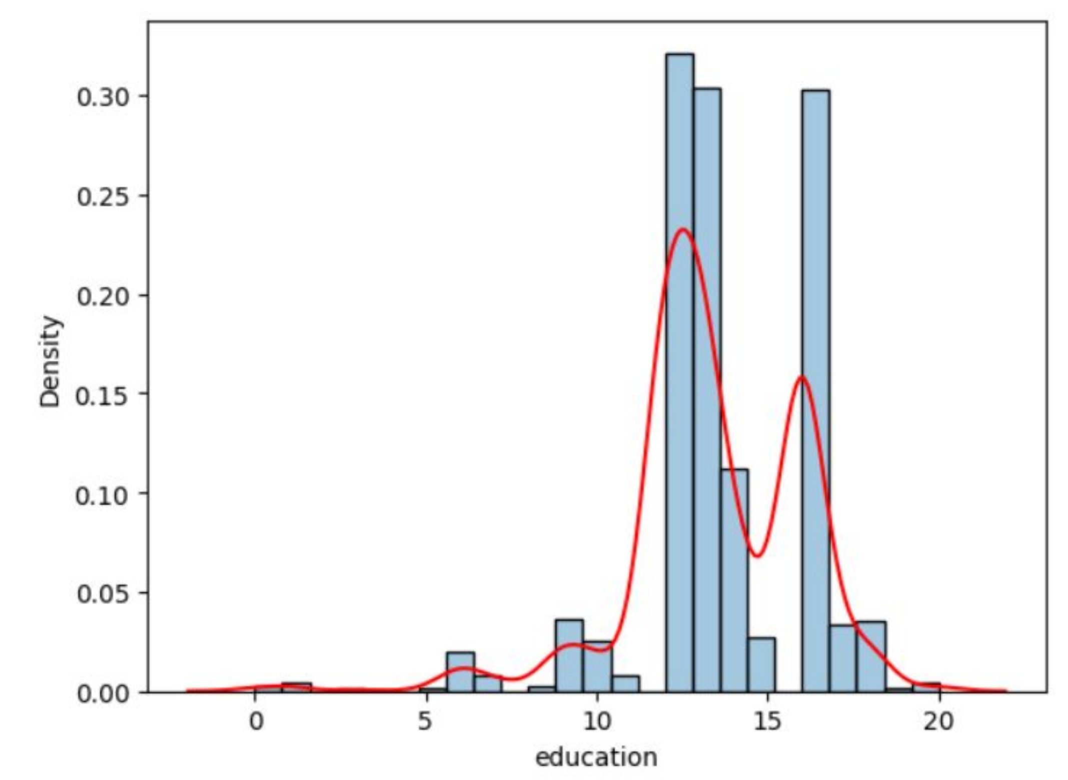
Gender Distribution



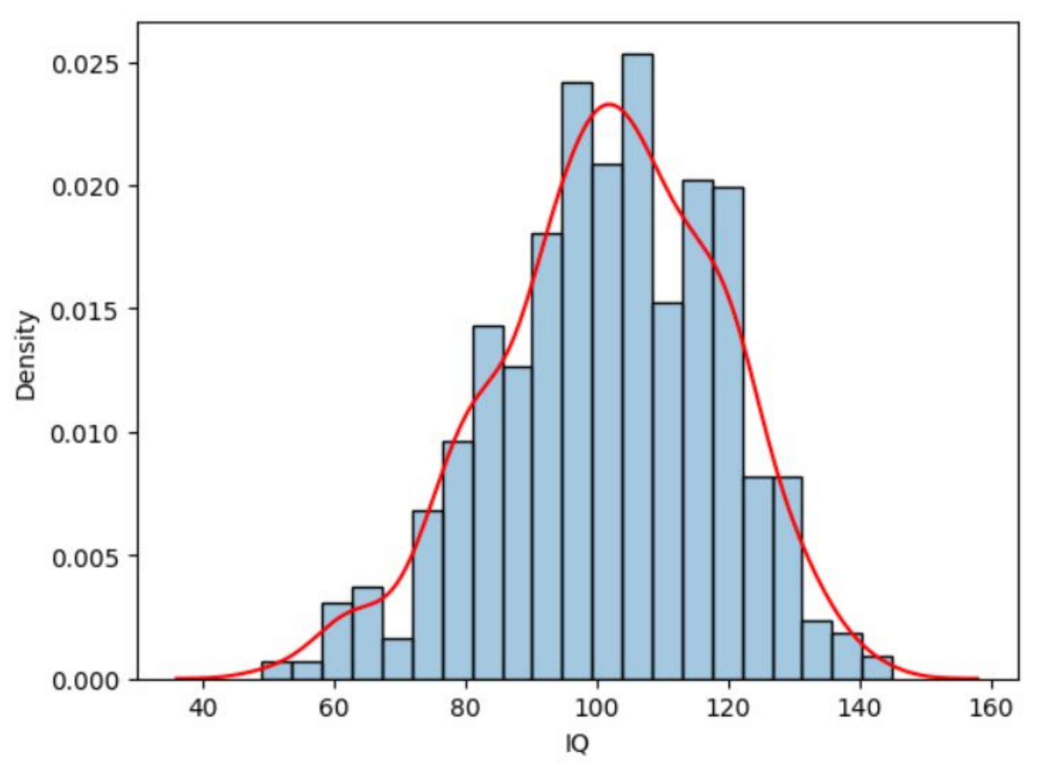
Age Distribution



Education Distribution

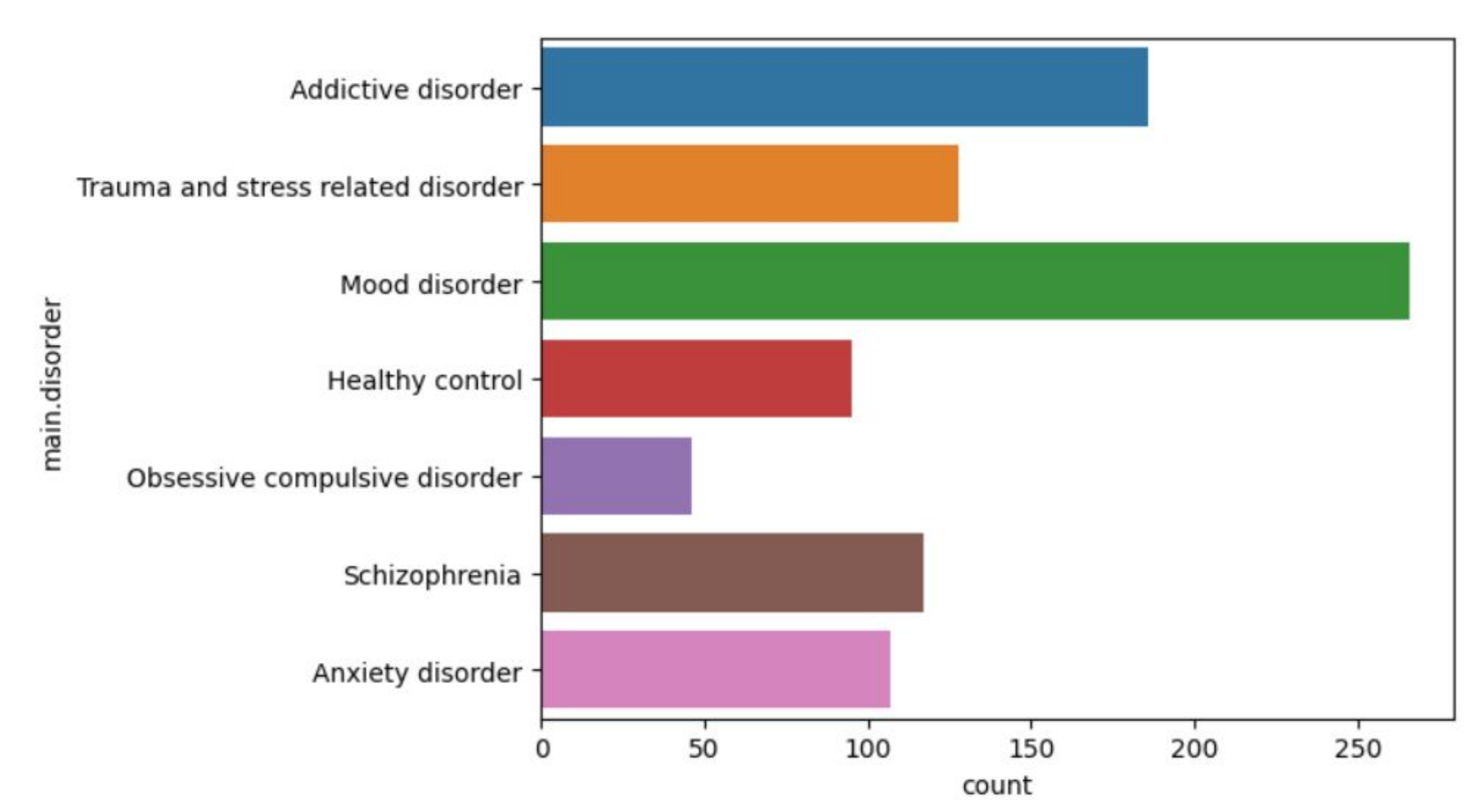


IQ Distribution

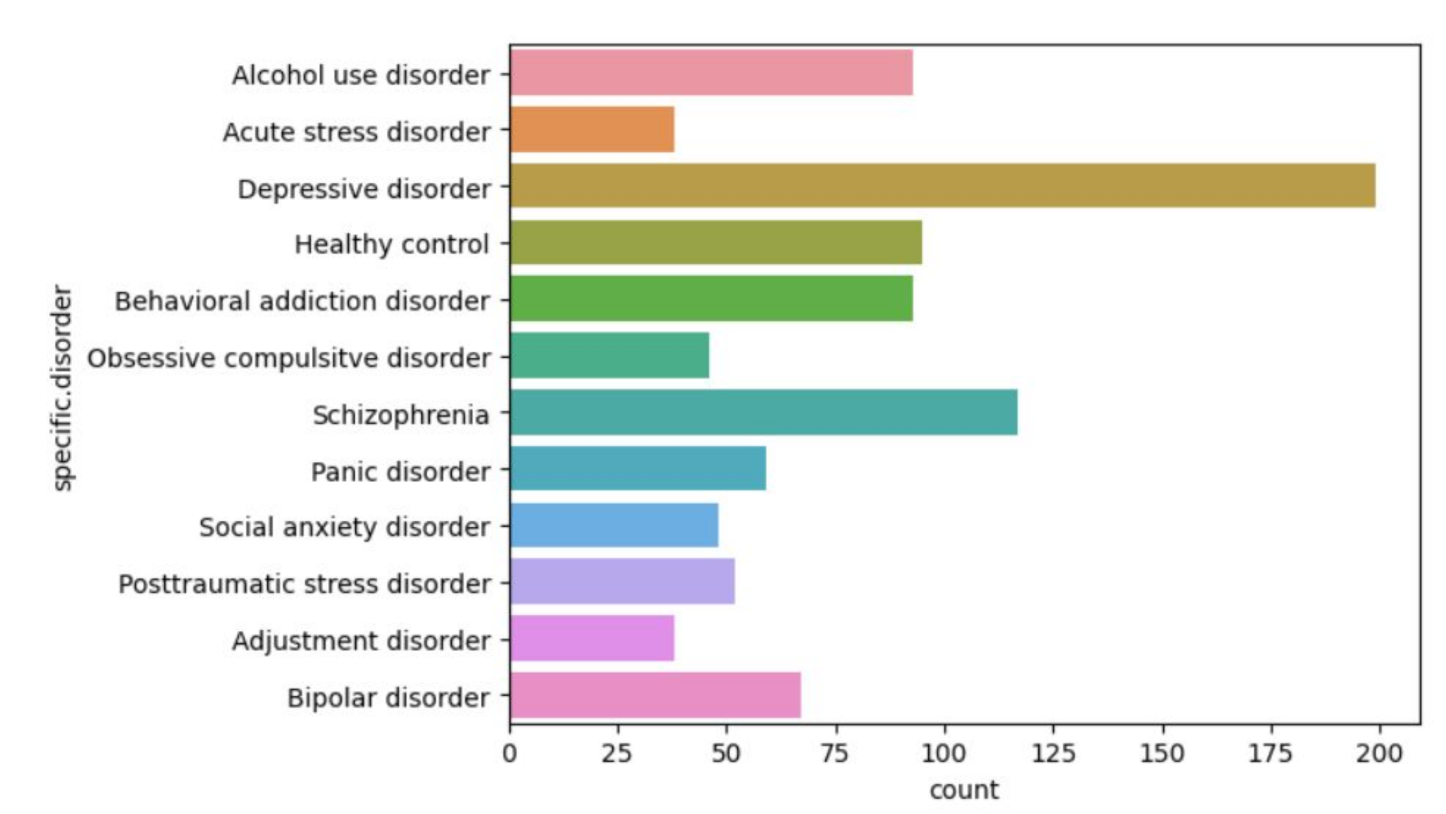


Main Disorder

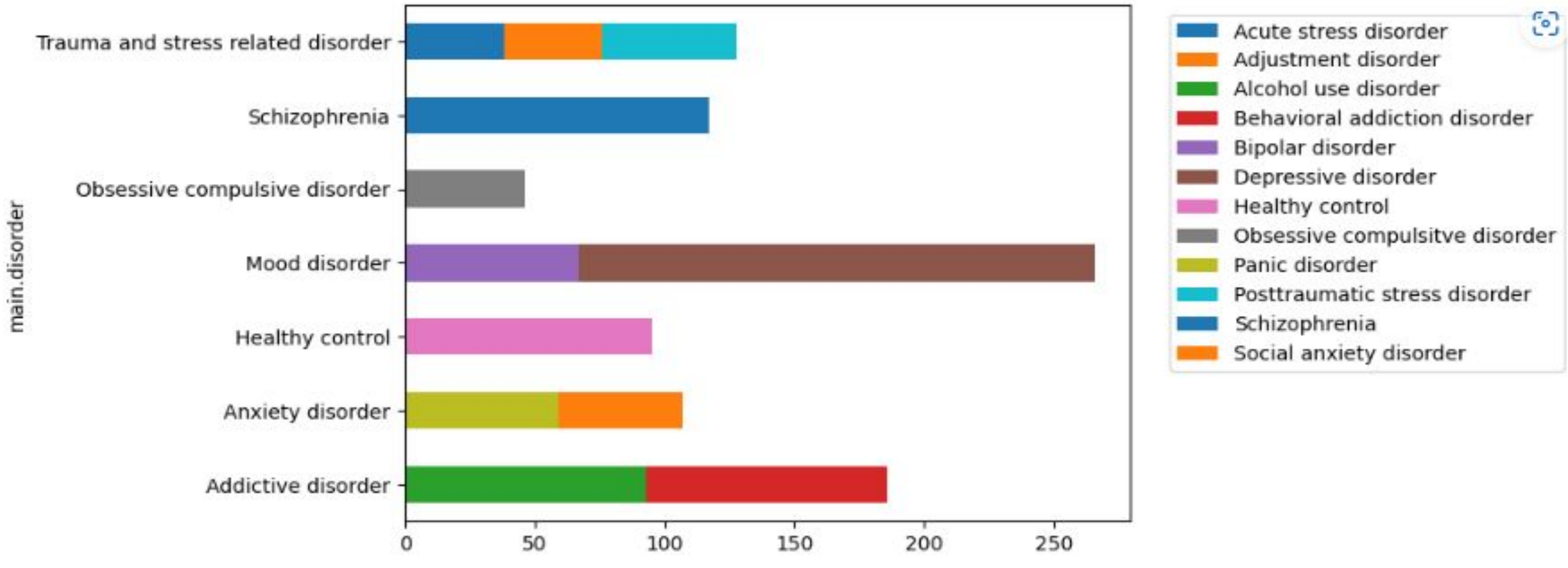
TARGET



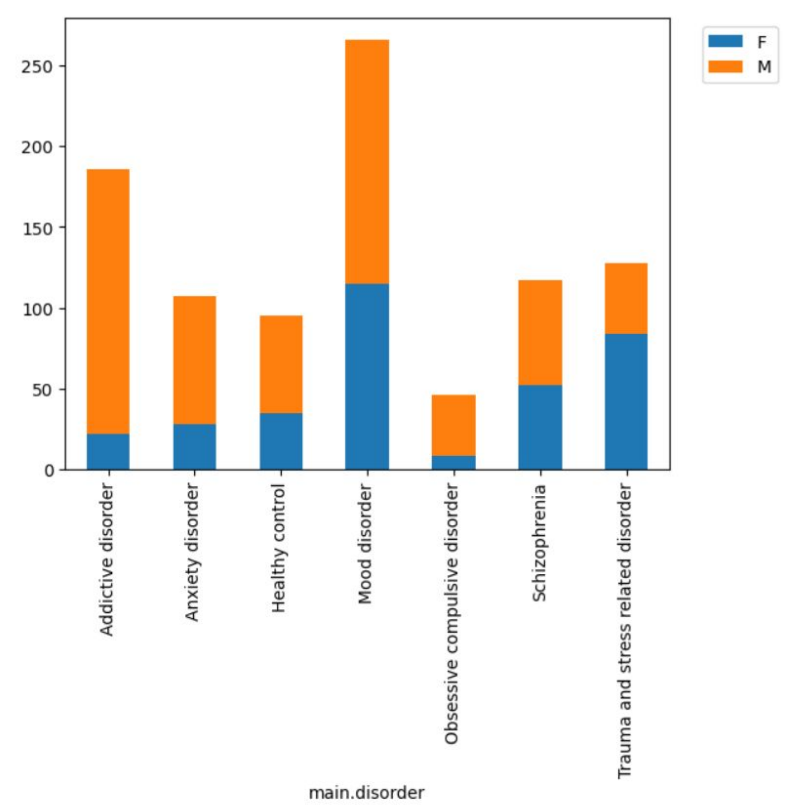
Specific Disorder



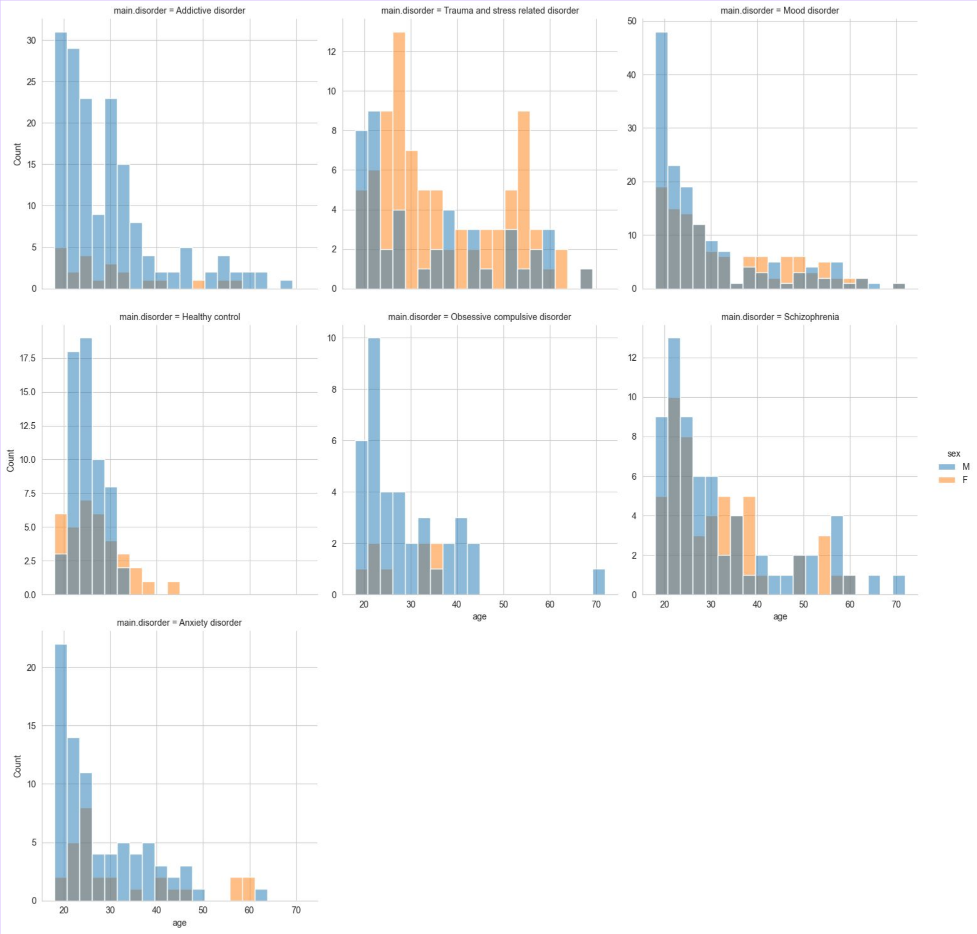
Specific Disorders within Main Disorder



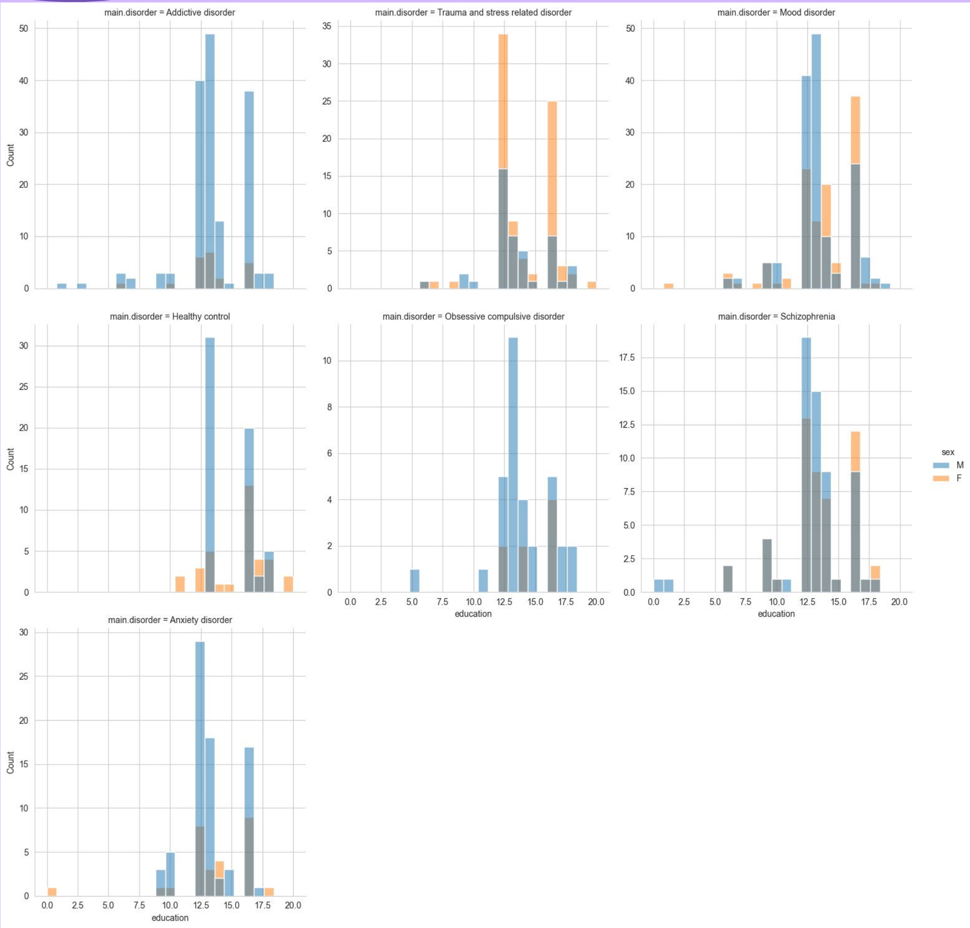
Gender VS Main Disorder



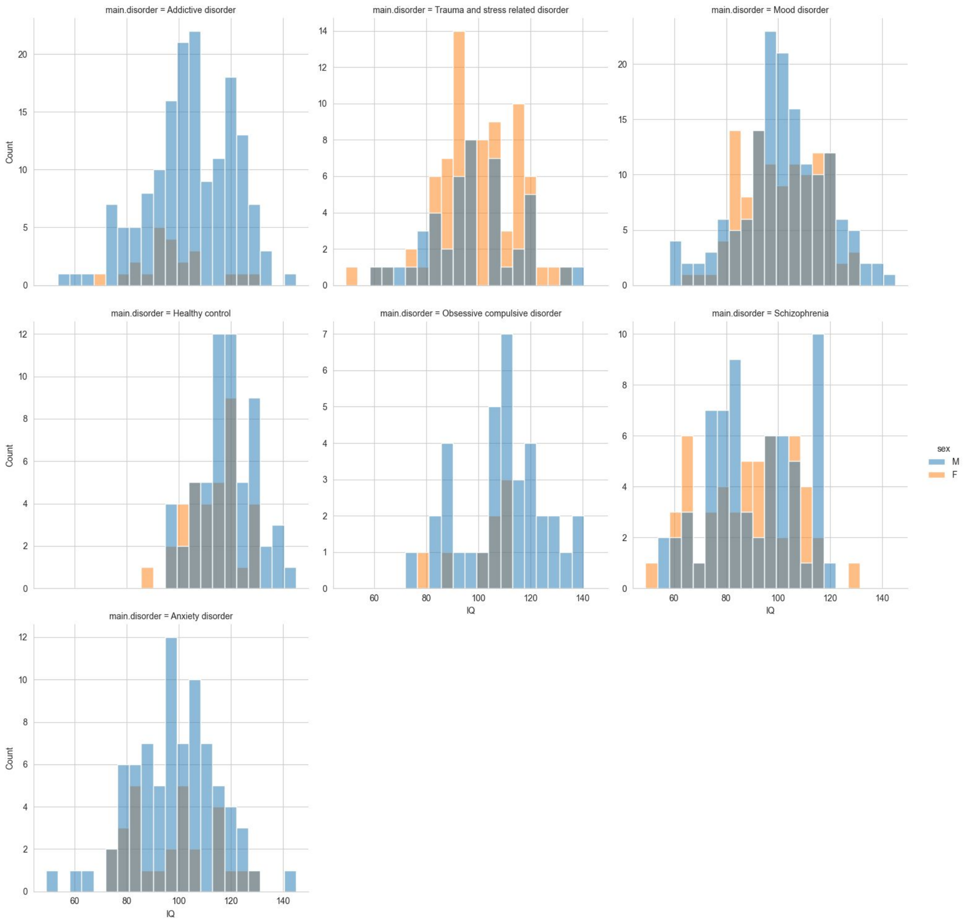
Age VS Main Disorder



Education VS Main Disorder

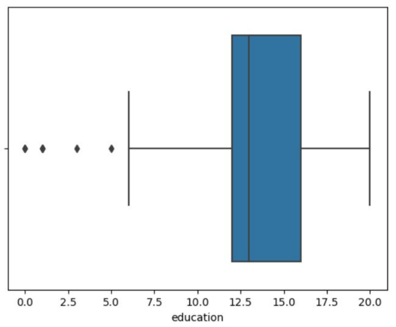


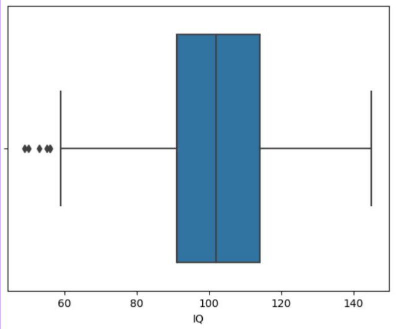
IQ VS Main Disorder



Null Values and Filling

```
education null values: 15
IQ null values: 13
Unnamed: 122 null values: 945
```



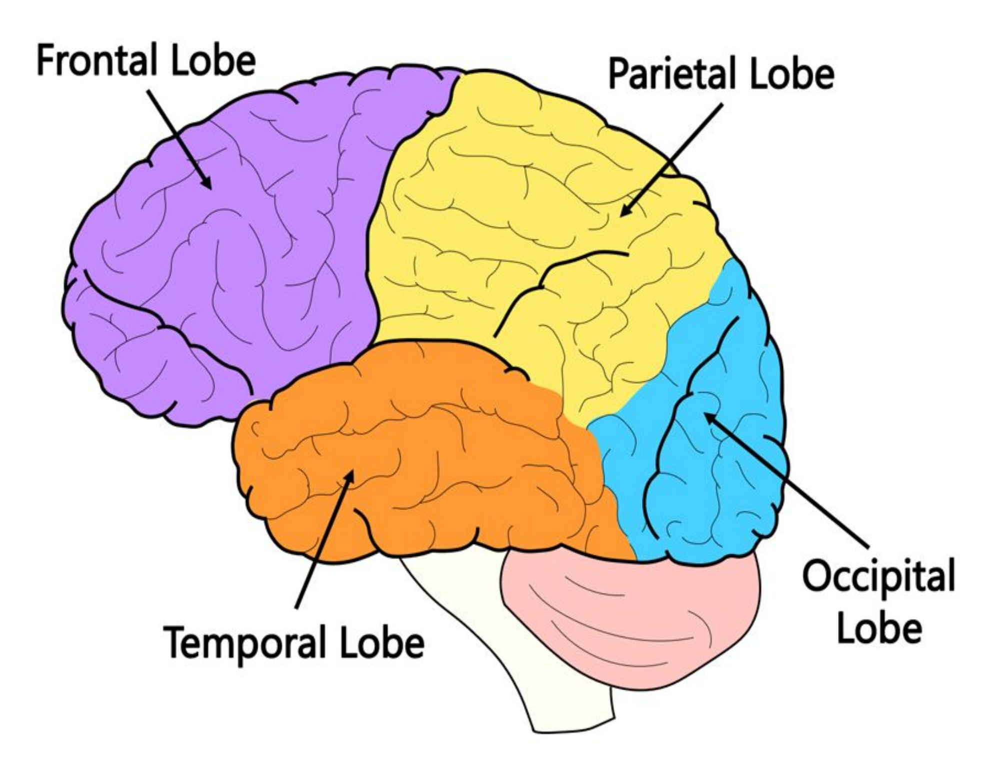


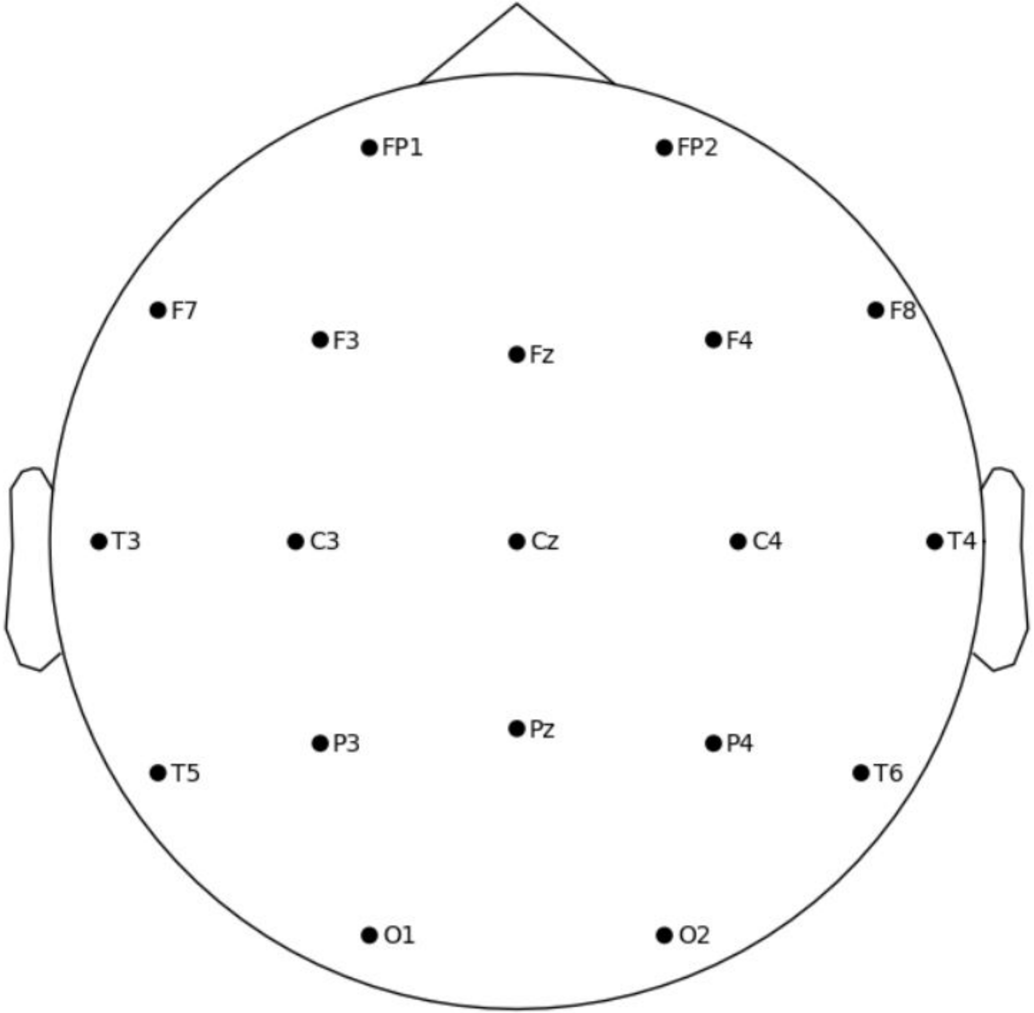
	no.	sex	age	eeg.date	education	IQ	main.disorder	specific.disorder	AB.A.delta.a.FP1	AB.A.delta.b.FP2	...	COH.F.gamma.o.Pz.p.P4
0	1	M	57.0	2012.8.30	13.0	102.0	Addictive disorder	Alcohol use disorder	35.998557	21.717375	...	55.989192
1	2	M	37.0	2012.9.6	6.0	120.0	Addictive disorder	Alcohol use disorder	13.425118	11.002916	...	45.595619
2	3	M	32.0	2012.9.10	16.0	113.0	Addictive disorder	Alcohol use disorder	29.941780	27.544684	...	99.475453
3	4	M	35.0	2012.10.8	18.0	126.0	Addictive disorder	Alcohol use disorder	21.496226	21.846832	...	59.986561
4	5	M	36.0	2012.10.18	16.0	112.0	Addictive disorder	Alcohol use disorder	37.775667	33.607679	...	61.462720

5 rows × 1148 columns

Electrode Labelling and Channels

F - Frontal - F7, F3, FZ, F4, F8
P - Parietal - P3, PZ, P4
T - Temporal - T4, T5, T6
C - Central - C3, Cz, C4
O - Occipital - o1, o2
Fp - Frontal Parietal - FP1, FP2



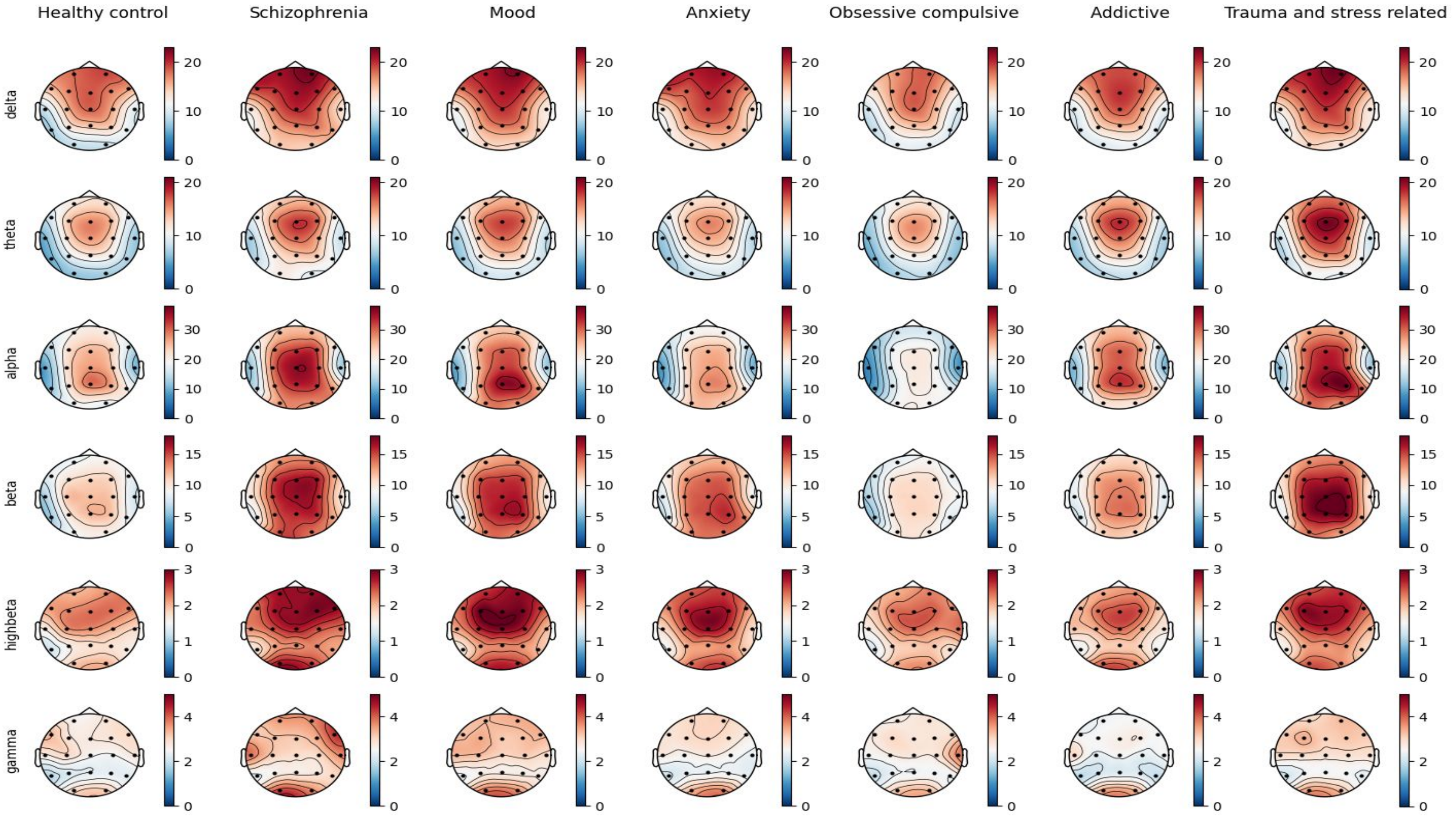

FP1
FP2
F7
F3
Fz
F4
F8
T3
C3
Cz
C4
T4
Pz
P3
P4
T5
T6
O1
O2

FREQUENCY BANDS

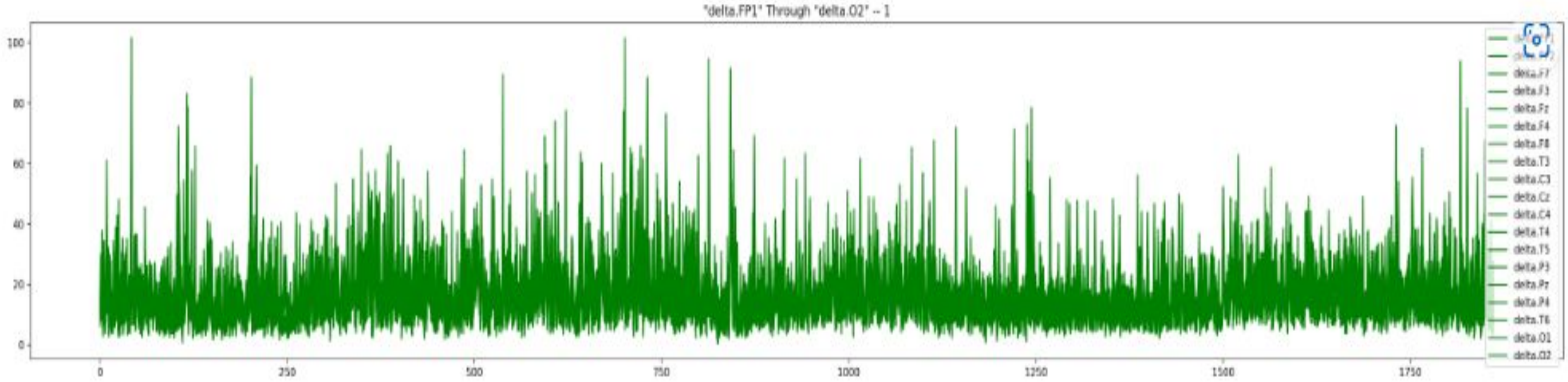
DELTA	1-4 Hz
THETA	4-8 Hz
ALPHA	8-12 Hz
BETA	12 -25 Hz
HIGH BETA	25-30 Hz
GAMMA	30-40 Hz

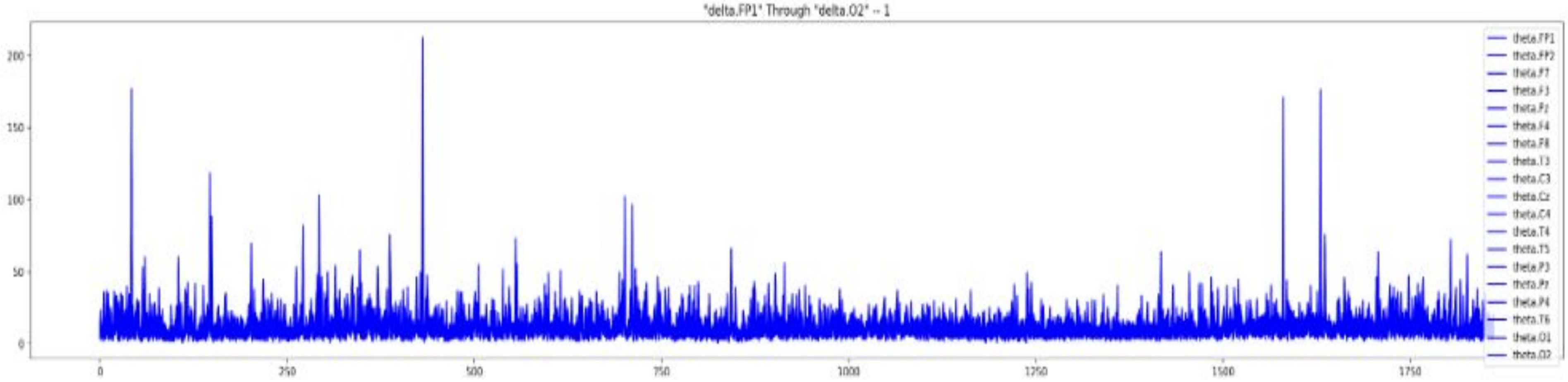
Pre processed data

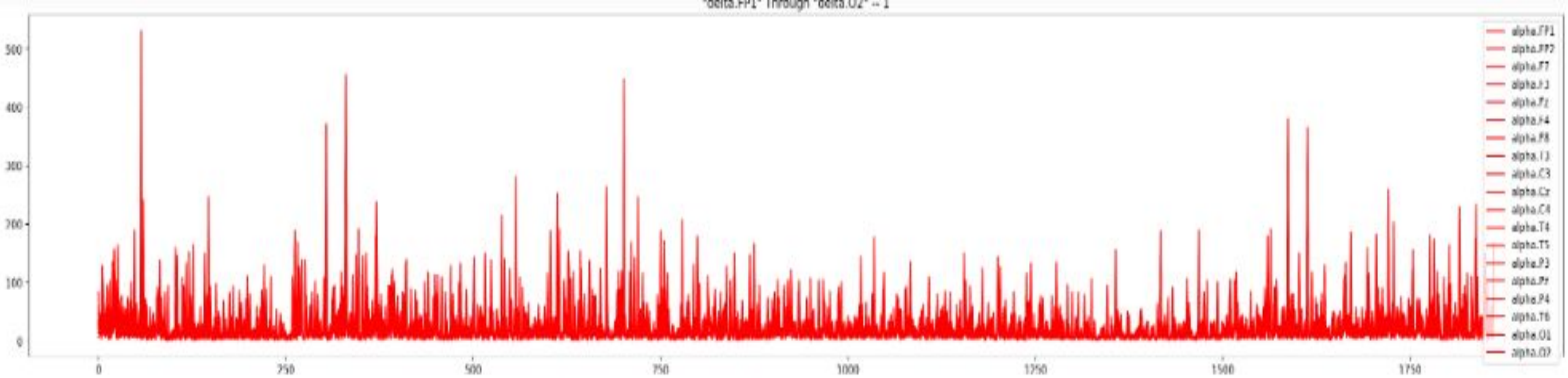
- EEG data transformed to Frequency domain using fast Fourier transformation (epoch =2s, sampling rate = 0.5–40Hz, resolution = 0.5Hz)
- Artifact rejection (Signals recorded by EEG that might mimic seizures but generated from outside the brain)
- $$coherence\,(f) = \frac{\left(\sum_N (a(x)u(y) + b(x)v(y))\right)^2 + \left(\sum_N (a(x)v(y) + b(x)u(y))\right)^2}{\sum_N (a(x)^2 + b(x)^2)\sum_N (u(y)^2 + v(y)^2)}$$

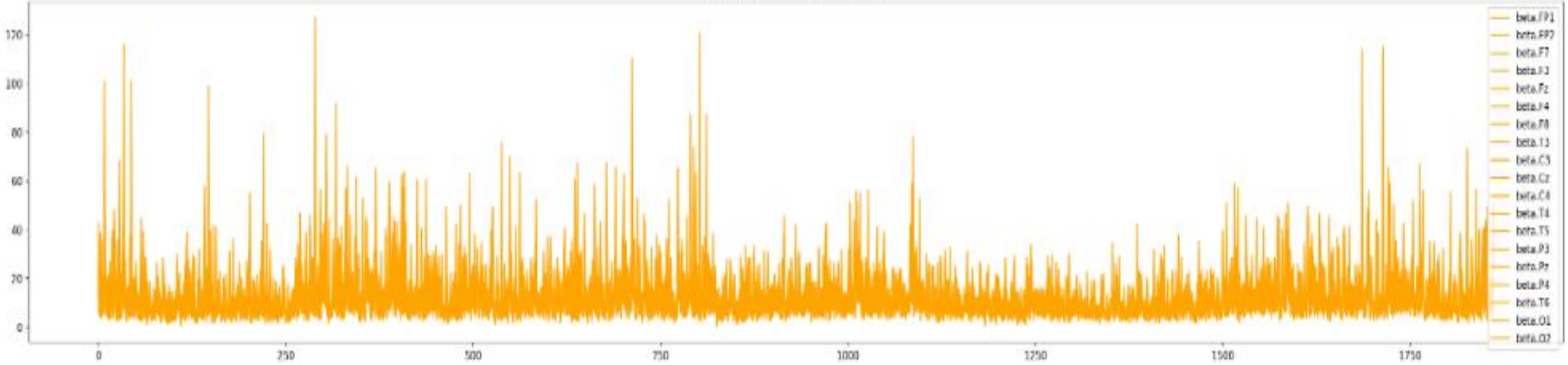


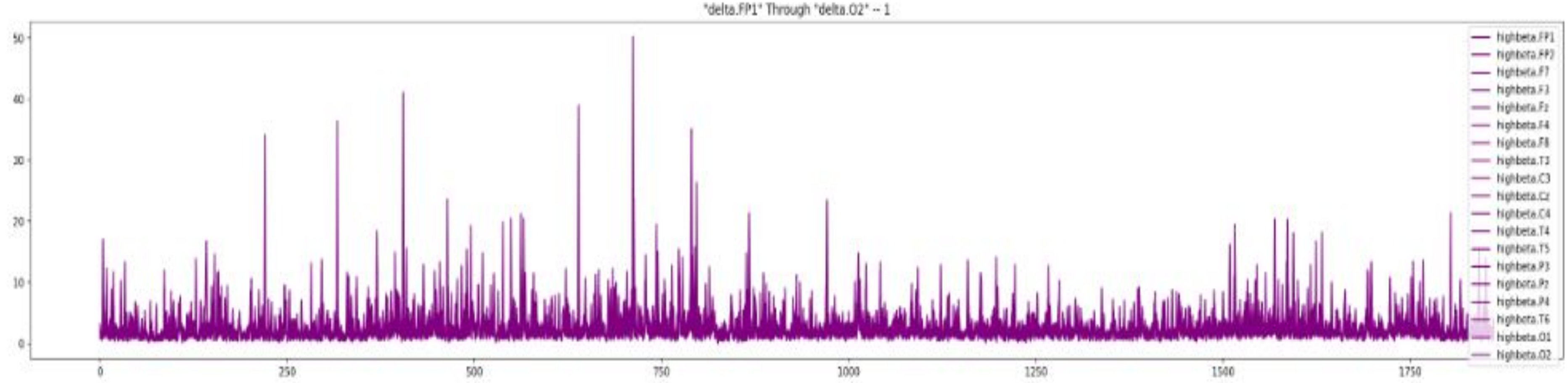
Healthy control
Schizophrenia
Mood
Anxiety
Obsessive compulsive
Addictive
Trauma and stress related
delta
theta
alpha
beta
highbeta
gamma
0
10
20
30
5
15
1
2
3
4


"delta.FP1" Through "delta.O2" -- 1
0
20
40
60
80
100
0
250
500
750
1000
1250
1500
1750
delta.F7
delta.F3
delta.Fz
delta.F4
delta.F8
delta.T3
delta.C3
delta.Cz
delta.C4
delta.T4
delta.T5
delta.P3
delta.Pz
delta.P4
delta.T6
delta.O1
delta.O2

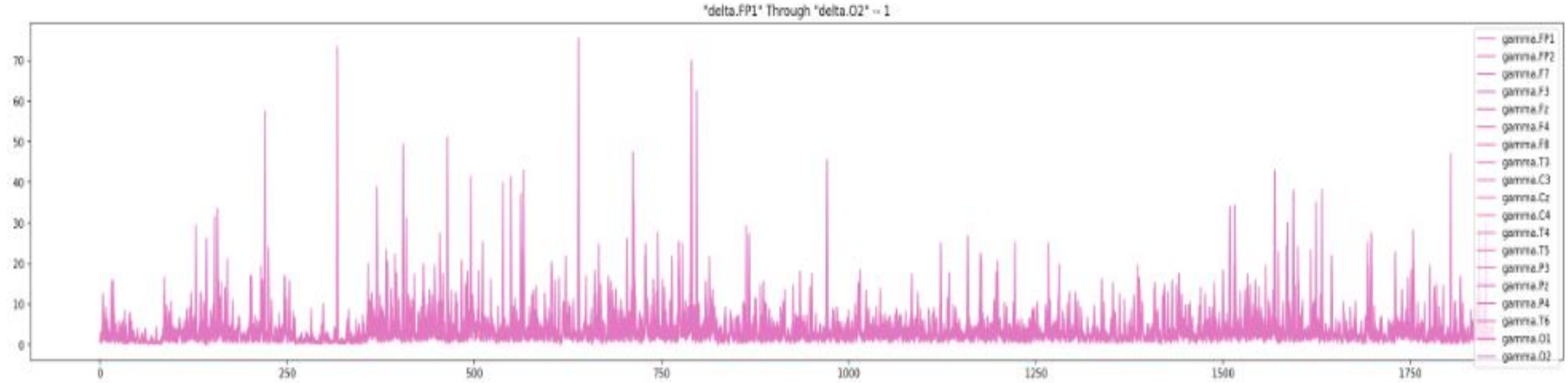

"delta.FP1" Through "delta.O2" -- 1
0
50
100
150
200
0
250
500
750
1000
1250
1500
1750
theta.FP1
theta.FP2
theta.F7
theta.F3
theta.Fz
theta.F4
theta.F8
theta.T3
theta.C3
theta.Cz
theta.C4
theta.T4
theta.T5
theta.P3
theta.Pz
theta.P4
theta.T6
theta.O1
theta.O2


"delta.FP1" Through "delta.O2" -- 1
alpha.FP1
alpha.FP2
alpha.F7
alpha.F3
alpha.Fz
alpha.F4
alpha.F8
alpha.T3
alpha.C3
alpha.Cz
alpha.C4
alpha.T4
alpha.T5
alpha.P3
alpha.Pz
alpha.P4
alpha.T6
alpha.O1
alpha.O2


"delta.FP1" Through "delta.O2" -- 1
beta.FP1
beta.FP2
beta.F7
beta.F3
beta.Fz
beta.F4
beta.F8
beta.T3
beta.C3
beta.Cz
beta.C4
beta.T4
beta.T5
beta.P3
beta.Pz
beta.P4
beta.T6
beta.O1
beta.O2



"delta.FP1" Through "delta.O2" -- 1
highbeta.FP1
highbeta.FP2
highbeta.F7
highbeta.F3
highbeta.Fz
highbeta.F4
highbeta.F8
highbeta.T3
highbeta.C3
highbeta.Cz
highbeta.C4
highbeta.T4
highbeta.T5
highbeta.P3
highbeta.Pz
highbeta.P4
highbeta.T6
highbeta.O1
highbeta.O2
0
10
20
30
40
50
250
500
750
1000
1250
1500
1750



"delta.FP1" Through "delta.O2" -- 1
gamma.FP1
gamma.FP2
gamma.F7
gamma.F3
gamma.Fz
gamma.F4
gamma.F8
gamma.T3
gamma.C3
gamma.Cz
gamma.C4
gamma.T4
gamma.T5
gamma.P3
gamma.Pz
gamma.P4
gamma.T6
gamma.O1
gamma.O2
0
10
20
30
40
50
60
70
250
500
750
1000
1250
1500
1750

Final Dataset

✓	Rename columns Train test split -> 10 data for each 7 classes for test data = 70
✓	Tackle Class Imbalance using SMOTE (oversample)
✓	Data with COH and without COH
✓	Complete Dataset

```
from imblearn.over_sampling import SMOTE
sm1=SMOTE(random_state = 2)
X_train_res, y_train_res = sm1.fit_resample(X_train, y_train.ravel())
```

	delta.FP1	delta.FP2	delta.F7	delta.F3	delta.Fz	delta.F4	delta.F8	delta.T3	delta.C3	delta.Cz	...	COH.gamma.Pz.T6	COH.gamma.Pz.O1
0	13.425118	11.002916	11.942516	15.272216	14.151570	12.456034	8.436832	9.975238	14.834740	10.950564	...	17.510824	26.777368
1	29.941780	27.544684	17.150159	23.608960	27.087811	13.541237	16.523963	12.775574	21.686306	18.367666	...	70.654171	39.131547
2	21.496226	21.846832	17.364316	13.833701	14.100954	13.100939	14.613650	8.063191	11.015078	11.639560	...	63.822201	36.478254
3	37.775667	33.607679	21.865556	21.771413	22.854536	21.456377	15.969042	9.434306	15.244523	17.041979	...	59.166097	51.465531
4	13.482096	14.095855	12.854630	11.727480	13.128924	11.627138	14.978119	6.899770	9.751346	14.141171	...	82.302355	83.938567
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1787	12.122552	12.279012	15.760783	12.450432	11.878292	11.353933	9.752290	13.445496	11.643723	11.886123	...	54.957847	63.642583
1788	19.248523	20.804489	27.454779	19.082305	19.648852	21.364365	17.278478	15.513716	19.290149	20.666848	...	70.722075	66.995461
1789	22.086091	21.417607	21.525632	21.311974	23.012317	21.271466	17.912482	13.813082	18.670484	21.926205	...	51.646214	54.720352
1790	11.324536	11.986965	11.874343	17.349002	16.150498	15.744348	9.514742	11.388541	10.305956	11.136205	...	64.279798	63.618312
1791	13.617149	25.411044	13.189382	17.272278	11.490694	11.879488	11.390569	9.697276	10.128936	11.595368	...	39.336023	31.119681

1792 rows × 1141 columns

04.

Machine Learning Approach

Algorithm

✓	RandomForestClassifier
✓	SupportVectorMachine
✓	LogisticRegression
✓	KNeighborsClassifier
✓	DecisionTreeClassifier
✓	SGDClassifier
✓	Perceptron
✓	GradientBoostingClassifier

Our Data

Data 'X' - Features

Normalization = MinMaxScaler

Labeled using Labeled Encoding

	delta.FP1	delta.FP2	delta.F7	delta.F3	delta.Fz	delta.F4	delta.F8	delta.T3	delta.C3	delta.Cz	...	gamma.Cz	gamma.C4	gamma.T4	g
0	0.225947	0.255878	0.324159	0.279571	0.255055	0.359052	0.310743	0.222554	0.344432	0.251227	...	0.439863	0.136631	0.199082	
1	0.118103	0.138452	0.125176	0.170737	0.154731	0.175230	0.232981	0.127276	0.159395	0.150392	...	0.161557	0.054611	0.115212	
2	0.421424	0.449684	0.221270	0.244298	0.184795	0.283808	0.299571	0.128850	0.247582	0.198764	...	0.119624	0.038502	0.228711	
3	0.283667	0.241900	0.258844	0.329630	0.234146	0.304259	0.282722	0.206390	0.351401	0.273289	...	0.099861	0.036480	0.059893	
4	0.443502	0.387614	0.369259	0.336499	0.263586	0.361706	0.429265	0.268025	0.431020	0.307567	...	0.086332	0.027065	0.044928	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1607	0.122076	0.121355	0.119557	0.151187	0.117311	0.146483	0.147213	0.109484	0.180386	0.171746	...	0.089026	0.027179	0.035509	
1608	0.073433	0.077306	0.052745	0.122505	0.111101	0.112257	0.096705	0.045604	0.182974	0.204321	...	0.077499	0.016654	0.015544	
1609	0.139985	0.133593	0.079051	0.182854	0.175706	0.204229	0.145008	0.098337	0.230444	0.236947	...	0.044336	0.011348	0.011032	
1610	0.183996	0.211869	0.149639	0.188011	0.193092	0.194965	0.198315	0.104013	0.205520	0.187964	...	0.118602	0.037055	0.066248	
1611	0.250958	0.227139	0.270682	0.294095	0.222577	0.248936	0.250903	0.193568	0.315480	0.297083	...	0.048373	0.017897	0.025575	

1612 rows × 114 columns

```
for model in models:
    scores.append(cross_val_score(model, Xmd_train, y, cv=5,
                                  scoring='accuracy', ))

print(scores)
```

```
[array([0.6183844 , 0.56824513, 0.68156425, 0.72067039, 0.68715084]), array([0.40111421, 0.39554318, 0.50558659, 0.530
72626, 0.54469274]), array([0.32311978, 0.32869081, 0.37430168, 0.4273743 , 0.41899441]), array([0.23119777, 0.2534818
9, 0.29050279, 0.24860335, 0.23184358]), array([0.48467967, 0.46518106, 0.54189944, 0.50837989, 0.52234637]), array
([0.42339833, 0.41504178, 0.49441341, 0.44692737, 0.47765363]), array([0.3091922 , 0.31197772, 0.32402235, 0.34078212,
0.39385475]), array([0.2729805 , 0.17270195, 0.17039106, 0.20391061, 0.29050279]), array([0.54038997, 0.5097493 , 0.59
497207, 0.63687151, 0.6424581 ])]
```

```
print(np.mean(scores,axis=1))
```

```
[0.655203   0.47553259 0.3744962  0.25112588 0.50449728 0.4514869
 0.33596583 0.22209738 0.58488819]
```

RandomForestClassifier

RandomForestClassifier

```
GridSearchCV(cv=5, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['entropy'], 'max_depth': [4, 7, 8],
                         'max_features': ['sqrt', 'log2'],
                         'n_estimators': [100, 200]},
             scoring='accuracy')
```

```
grid.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 8,
 'max_features': 'sqrt',
 'n_estimators': 200}
```

```
accuracy_score(y_test, yhat)
```

```
0.12857142857142856
```

KNeighborsClassifier

```
GridSearchCV(cv=5, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['ball_tree'], 'leaf_size': [30, 50],
                         'n_neighbors': [5, 8],
                         'weights': ['uniform', 'distance']},
             scoring='accuracy')
```

```
grid.best_params_
```

```
{'algorithm': 'ball_tree',
 'leaf_size': 30,
 'n_neighbors': 5,
 'weights': 'distance'}
```

```
accuracy_score(y_test, yhat)
```

```
0.2
```

05.

Deep Learning Approach

Train-Valid Split

```python
from sklearn.model_selection import train_test_split
def train_test_dataset(X,y, split):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=split, random_state=42)
    return X_train, X_test, y_train, y_test

X_train, X_test, y_train, y_test = train_test_dataset(X,Y, split = 0.10)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((1612, 1140), (180, 1140), (1612, 7), (180, 7))
```

```python
# Make sure that in 10% split we will have all the classes in valid and train set.

assert len(y_test.sum(axis=0)) == len(y_train.sum(axis=0))
y_test.sum(axis=0), y_train.sum(axis=0)
```

```
(Addictive disorder                  25
 Anxiety disorder                    23
 Healthy control                     19
 Mood disorder                       34
 Obsessive compulsive disorder       28
 Schizophrenia                       26
 Trauma and stress related disorder  25
 dtype: int64,
 Addictive disorder                  231
 Anxiety disorder                    233
 Healthy control                     237
 Mood disorder                       222
 Obsessive compulsive disorder       228
 Schizophrenia                       230
 Trauma and stress related disorder  231
 dtype: int64)
```

10% valid test set to evaluate model while training

Our Target Data and Normalization in feature set

	Addictive disorder	Anxiety disorder	Healthy control	Mood disorder	Obsessive compulsive disorder	Schizophrenia	Trauma and stress related disorder
1013	0	1	0	0	0	0	0
583	0	0	0	0	0	1	0
787	0	0	0	1	0	0	0
30	0	0	0	0	0	0	1
1560	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...
1130	0	0	1	0	0	0	0
1294	0	0	0	0	1	0	0

Labeled using One Hot Encoding

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_train = pd.DataFrame(scaler.fit_transform(X_train),columns=X.columns)
X_test = pd.DataFrame(scaler.transform(X_test),columns=X.columns)
```

Preparing Custom DATASET:

```python
#Prepare manual dataset that will return target and features to our model
torch.manual_seed(555)
datasets = {}
class EEGDataset(Dataset):
    def __init__(self, X,y):
        self.X = torch.tensor(X.values)
        self.y = torch.tensor(y.values)

    def __getitem__(self, idx):
        preds = self.X[idx].float()
        trgts = self.y[idx].float()

        return preds, trgts

    def __len__(self):
        return len(self.X)

datasets['train'] = EEGDataset(X_train,y_train)
datasets['test'] = EEGDataset(X_test,y_test)
```

Data Loader

```python
train_loader = DataLoader(datasets['train'], shuffle=True)
test_loader = DataLoader(datasets['test'],shuffle=True)
```

Model- 1 "S"

```
Small Model parameters
    96
    32
254912
     7
______
255047
```

```python
class eegConv1d(nn.Module):
    def __init__(self, input_size = 1, hidden_size=32, out_size=7):
        super().__init__()
        self.conv1d   = nn.Conv1d(input_size, hidden_size, kernel_size = 3,stride =1)
        # nn.init.normal_(self.conv1d.weight, mean=0.0, std=1)
        self.linear = nn.Linear(36416, out_size)

    def forward(self, seq):
        #seq = seq.unsqueeze(dim=0)
        out = self.conv1d(seq)
        out = out.reshape(-1,36416)
        out = self.linear(out)
        return out
```

```
optimizer, criterion
```

```
(Adam (
 Parameter Group 0
     amsgrad: False
     betas: (0.9, 0.999)
     capturable: False
     differentiable: False
     eps: 1e-08
     foreach: None
     fused: False
     lr: 0.0005
     maximize: False
     weight_decay: 0
 ),
 CrossEntropyLoss())
```

Model- 2 “L”

```
class CNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv1d(1,64,6,stride = 1)
        # nn.init.normal_(self.conv1.weight, mean=0.0, std=1)
        self.conv2 = nn.Conv1d(64,8,6,stride = 1)
        # nn.init.normal_(self.conv2.weight, mean=0.0, std=1)
        self.linear1 = nn.Linear(9040,1024) #
        self.linear2 = nn.Linear(1024, 7)
        self.leakyrelu = nn.LeakyReLU()
        self.dropout = nn.Dropout(p = 0.25)

        #self.softmax = nn.Softmax()
    def forward(self, X):
        #X = X.unsqueeze(dim=0)
        X = self.leakyrelu(self.conv1(X))
        X = self.leakyrelu(self.conv2(X))
#           X = self.dropout(X)
#           X = self.dropout(X)
        X = X.view(-1, 9040)
        X = self.leakyrelu(self.linear1(X))
        X = self.dropout(X)
        X = self.leakyrelu(self.linear2(X))

        return X
```

```
Large Model parameters
   384
    64
  3072
     8
9256960
  1024
  7168
     7
_______
9268687
```

```
optimizer, criterion
```

```
(Adam (
 Parameter Group 0
     amsgrad: False
     betas: (0.9, 0.999)
     capturable: False
     differentiable: False
     eps: 1e-08
     foreach: None
     fused: False
     lr: 0.0005
     maximize: False
     weight_decay: 0
 ),
 CrossEntropyLoss())
```

Select the required model and parameters through user input:

```python
print("Please select from above models... 0 for simple model, 1 for deep model")
mod = int(input())
if mod == 0:
    CNNmodel = CNNmodel_s
    mo = "Simple"
elif mod == 1:
    CNNmodel = CNNmodel_l
    mo = "Deep"
```

```python
criterion = nn.CrossEntropyLoss()
print("Choose optimizer. Enter 0 for Adam and 1 for SGD")
op = input()
if int(op) == 0:
    optim = 'ADAM'
    print("Enter the learning rate")
    learning_rate = float(input())
    optimizer = torch.optim.Adam(CNNmodel.parameters(), lr=learning_rate)

elif int(op) == 1:
    optim = 'SGD'
    print("Enter the learning rate")
    learning_rate = float(input())
    print("Enter the momentum value.. (0-1)")
    m = float(input())
    optimizer = torch.optim.SGD(CNNmodel.parameters(), lr=learning_rate, momentum = m)
```

Training approaches

```
for i in range(epochs):
    start_time = time.time()
    confusion_matrix = np.zeros((7,7))
    train_correct = 0
    train_total = 0
    val_total   = 0
    train_acc   = 0
    val_acc     = 0
    val_correct = 0
    CNNmodel.train()
    for X_train, y_train in train_loader:
        y_train = y_train.to(device)
        X_train = X_train.to(device)
        y_pred = CNNmodel(X_train)
        predicted = torch.max(y_pred, 1)[1]
        train_total += y_train.size(0)
        train_correct += (predicted == torch.argmax(y_train,1)).sum().item()
        train_acc = 100 * (train_correct / train_total)
        train_loss = criterion(y_pred, y_train)
        optimizer.zero_grad()
        train_loss.backward()
        optimizer.step()
    train_losses.append(train_loss)
    train_accuracy.append(train_acc)
```

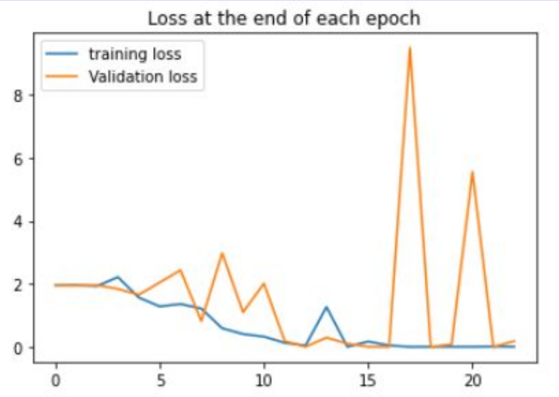
Validation and model saving approach:

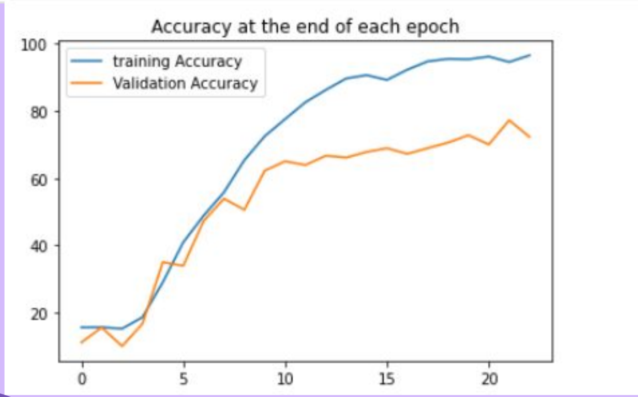
```
# Run the validation batches
with torch.no_grad():
    for b, (X_val, y_val) in enumerate(test_loader):
        y_val = y_val.to(device)
        X_val = X_val.to(device)
        yhat_val = CNNmodel(X_val)
        _, predicted = torch.max(yhat_val.data, 1)
        val_total += y_val.size(0)  #keep track of total
        actual = torch.argmax(y_val,1)
        val_correct += (predicted == actual).sum().item()  #.item() give the raw number
        confusion_matrix[predicted][actual] +=1
        val_acc = 100 * (val_correct / val_total)
        val_loss = criterion(yhat_val, y_val)
val_losses.append(val_loss)
val_accuracy.append(val_acc)
end_time = time.time()
print(f"Epoch {i} train_loss:{train_loss:10.3f} train_acc:{train_acc:10.4f} val_acc: {val_acc:10.3f} val_loss:{val_loss:10.4f}")

if val_loss > best_val_loss:
    j = j+1
    if j==10:
        print(f"................Early stoping...................at Epoch :{i+1}" )
        print(confusion_matrix)
        break
elif best_val_loss > val_loss:
    j = 0
    best_val_loss = val_loss
    xyz = 'models/with_COH_epoch_' + str(i) + '_model_' + str(mo) +'_' +str(optim) + '.pth.tar'
    torch.save(CNNmodel.state_dict(), xyz)
```

Result - Model- L with COH

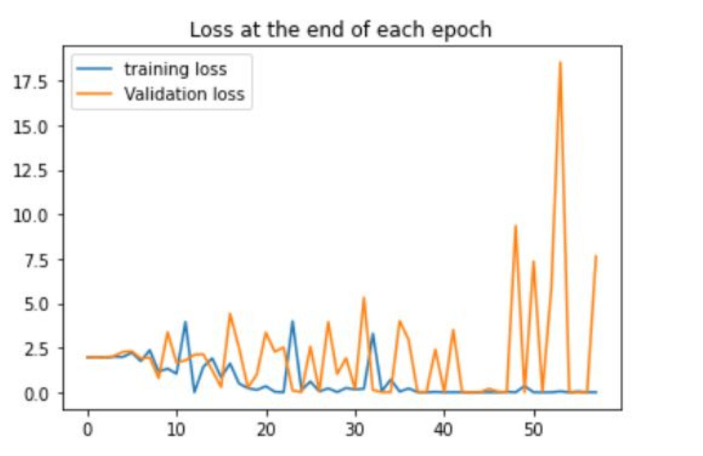
```
Epoch 16 train_loss:    0.043 train_acc:   92.2457 val_acc:     67.222 val_loss:    0.0001
Epoch 17 train_loss:   -0.000 train_acc:   94.7270 val_acc:     68.889 val_loss:    9.4820
Epoch 18 train_loss:    0.004 train_acc:   95.4715 val_acc:     70.556 val_loss:    0.0001
Epoch 19 train_loss:    0.004 train_acc:   95.3474 val_acc:     72.778 val_loss:    0.0792
Epoch 20 train_loss:    0.003 train_acc:   96.1538 val_acc:     70.000 val_loss:    5.5507
Epoch 21 train_loss:    0.012 train_acc:   94.5409 val_acc:     77.222 val_loss:    0.0002
Epoch 22 train_loss:    0.007 train_acc:   96.5261 val_acc:     72.222 val_loss:    0.1798
................Early stoping...................at Epoch :23
[[14.  1.  0.  6.  0.  0.  0.]
 [ 1. 19.  0.  7.  0.  0.  2.]
 [ 2.  0. 16.  0.  0.  1.  0.]
 [ 3.  1.  0. 13.  0.  1.  3.]
 [ 1.  1.  0.  0. 28.  0.  0.]
 [ 4.  1.  3.  4.  0. 21.  1.]
 [ 0.  0.  0.  4.  0.  3. 19.]]
```

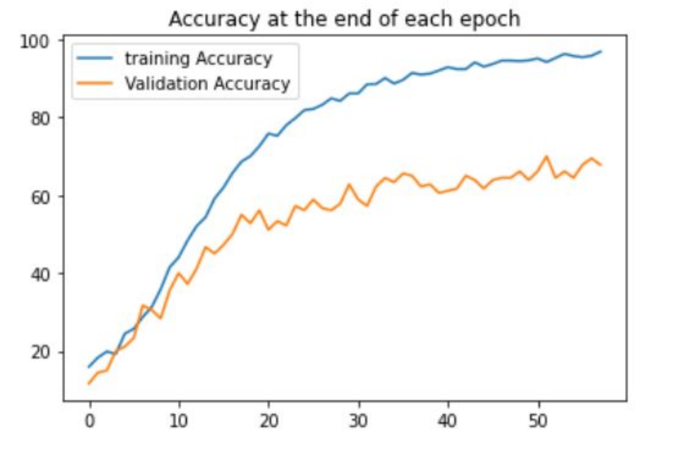




Result - Model- L for data with no COH data

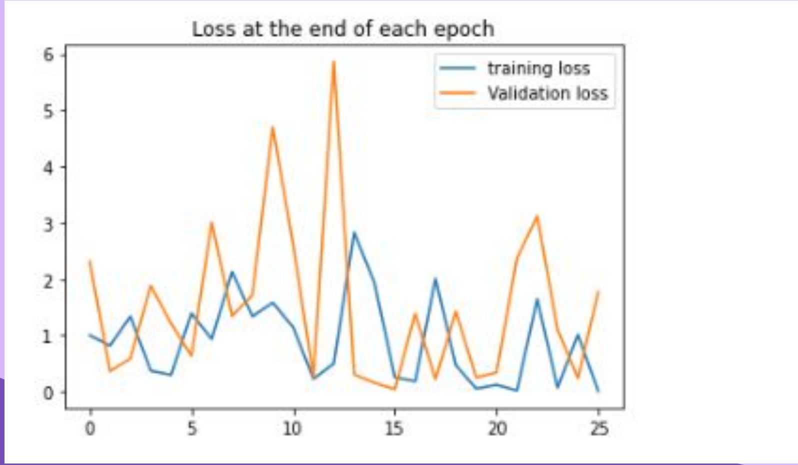
```
Epoch 56 train_loss:     0.008 train_acc:   95.7816 val_acc:     69.444 val_loss:     0.0000
Epoch 57 train_loss:     0.000 train_acc:   96.8362 val_acc:     67.778 val_loss:     7.6265
................Early stoping...................at Epoch :58
[[18.  2.  2.  8.  0.  0.  0.]
 [ 0. 14.  0.  3.  0.  1.  1.]
 [ 3.  1. 16.  3.  1.  2.  2.]
 [ 2.  6.  0. 12.  0.  0.  4.]
 [ 0.  0.  0.  0. 27.  0.  0.]
 [ 1.  0.  0.  6.  0. 20.  3.]
 [ 1.  0.  1.  2.  0.  3. 15.]]
```

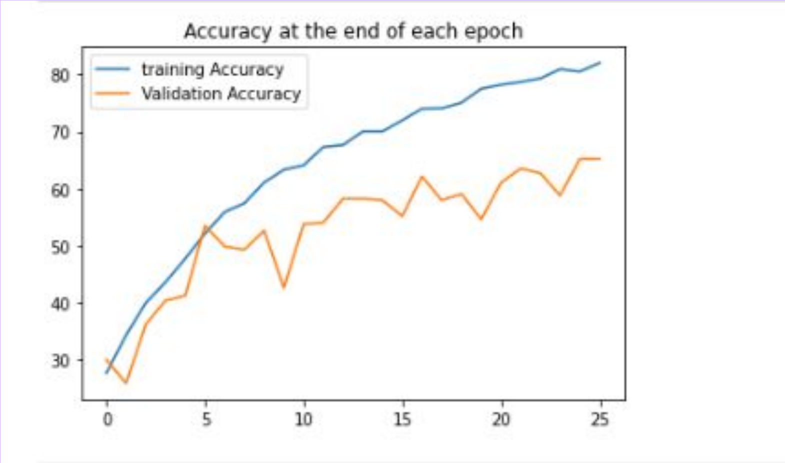




Small Model 1 ‘S’ with COH

```
Epoch 11 train_loss:    0.097 train_acc:  88.1368 val_acc:    70.474 val_loss:    3.0340
Epoch 12 train_loss:    0.024 train_acc:  87.7879 val_acc:    70.195 val_loss:    0.1047
Epoch 13 train_loss:    0.090 train_acc:  87.7879 val_acc:    67.688 val_loss:    0.6508
Epoch 14 train_loss:    0.068 train_acc:  89.6022 val_acc:    72.423 val_loss:    0.4997
.................Early stoping....................at Epoch :15
[[24.  1.  0.  8.  0.  0.  3.]
 [ 7. 31.  0. 11.  0.  4.  1.]
 [ 2.  0. 41.  2.  0.  1.  0.]
 [ 4.  5.  0. 21.  1.  3.  1.]
 [ 1.  0.  0.  4. 61.  0.  2.]
 [ 1.  2.  4.  8.  0. 38.  3.]
 [ 6.  4.  0.  8.  0.  2. 44.]]
```





06.

Comparison Study

Machine Learning Approach

Random Forest Classifier

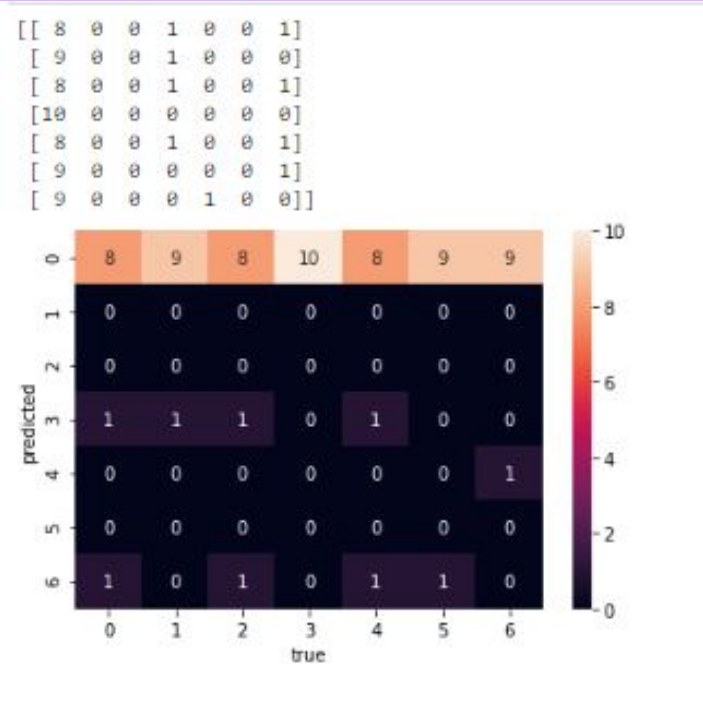
	precision	recall	f1-score	support
0.0	0.14	0.90	0.24	10
1.0	0.00	0.00	0.00	10
2.0	0.00	0.00	0.00	10
3.0	0.00	0.00	0.00	10
4.0	0.00	0.00	0.00	10
5.0	0.00	0.00	0.00	10
6.0	0.00	0.00	0.00	10
accuracy			0.13	70
macro avg	0.02	0.13	0.03	70
weighted avg	0.02	0.13	0.03	70

KNeighborsClassifier

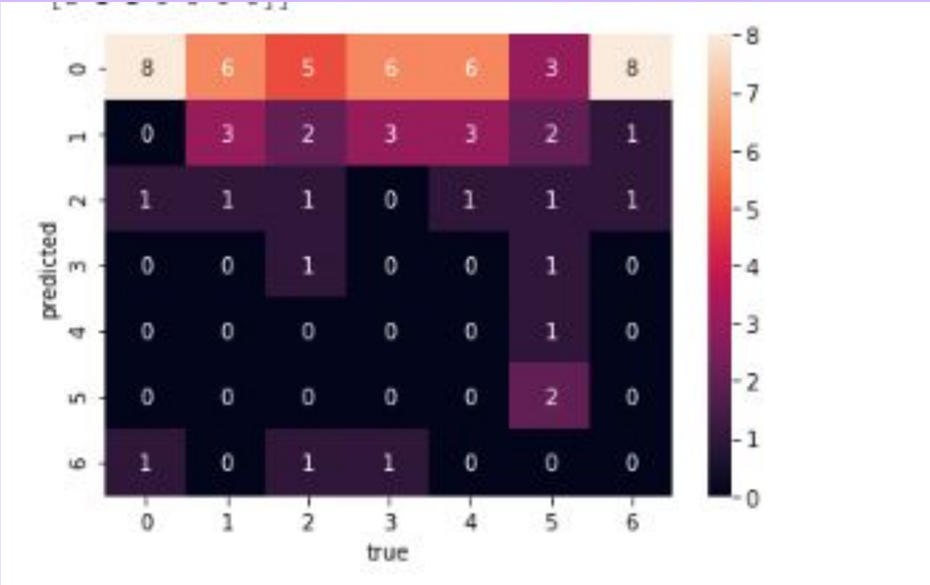
```
]: print(classification_report(y_test, yhat))
```

	precision	recall	f1-score	support
0.0	0.19	0.80	0.31	10
1.0	0.21	0.30	0.25	10
2.0	0.17	0.10	0.12	10
3.0	0.00	0.00	0.00	10
4.0	0.00	0.00	0.00	10
5.0	1.00	0.20	0.33	10
6.0	0.00	0.00	0.00	10
accuracy			0.20	70
macro avg	0.22	0.20	0.15	70
weighted avg	0.22	0.20	0.15	70

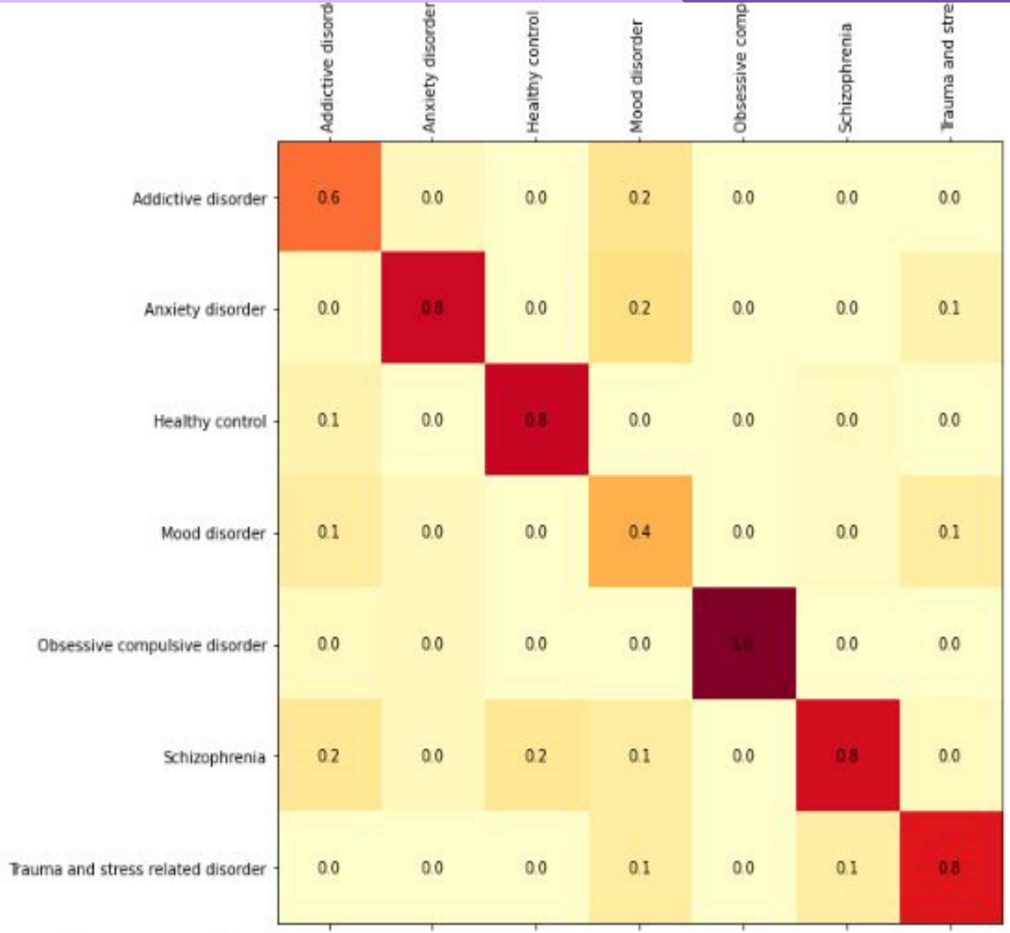
Confusion matrix on RandomForest Classifier on Test dataset



Confusion matrix on KNeighborsClassifier



Confusion matrix
For Large model on
Validation dataset
For all data

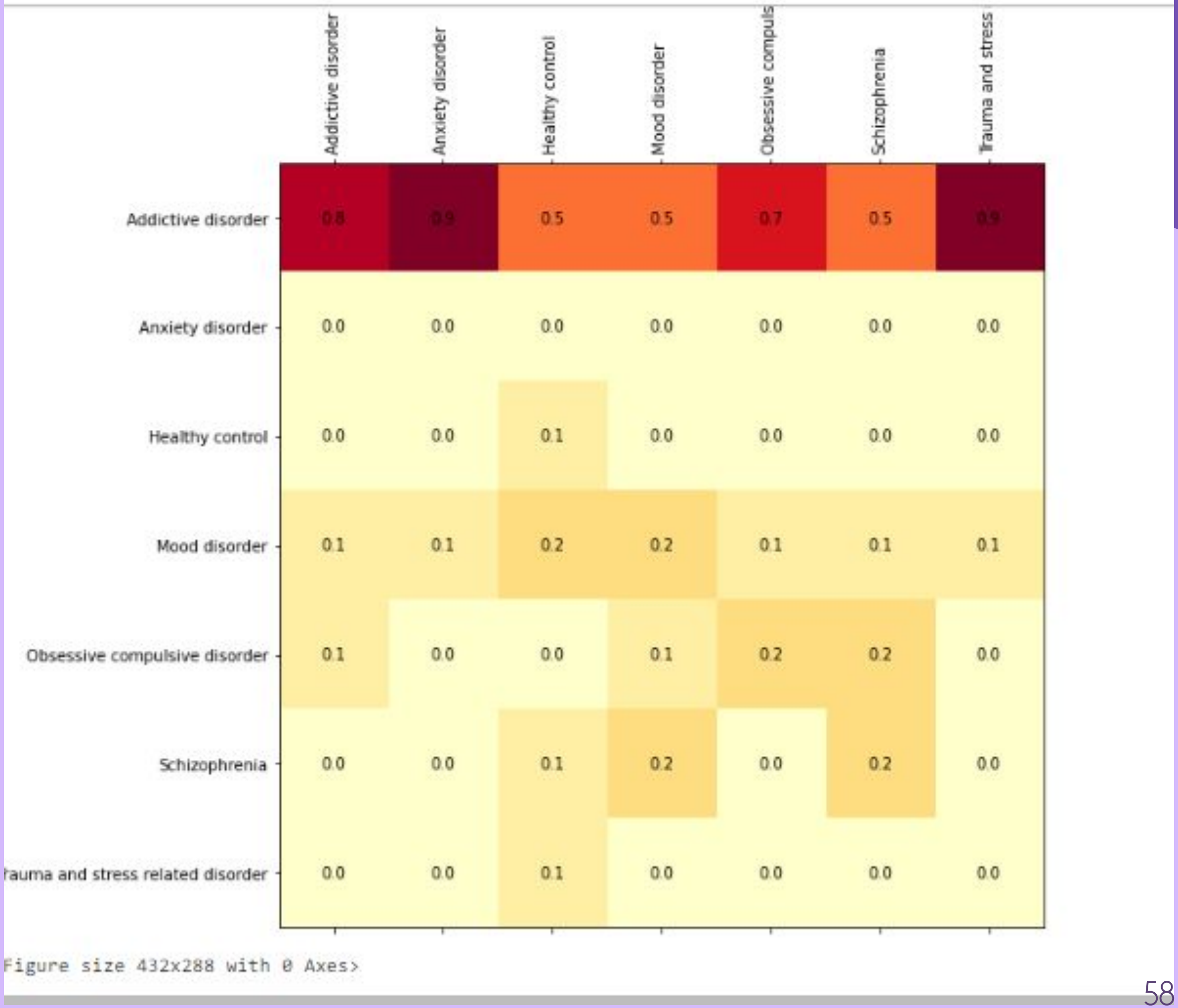


<Figure size 432x288 with 0 Axes>

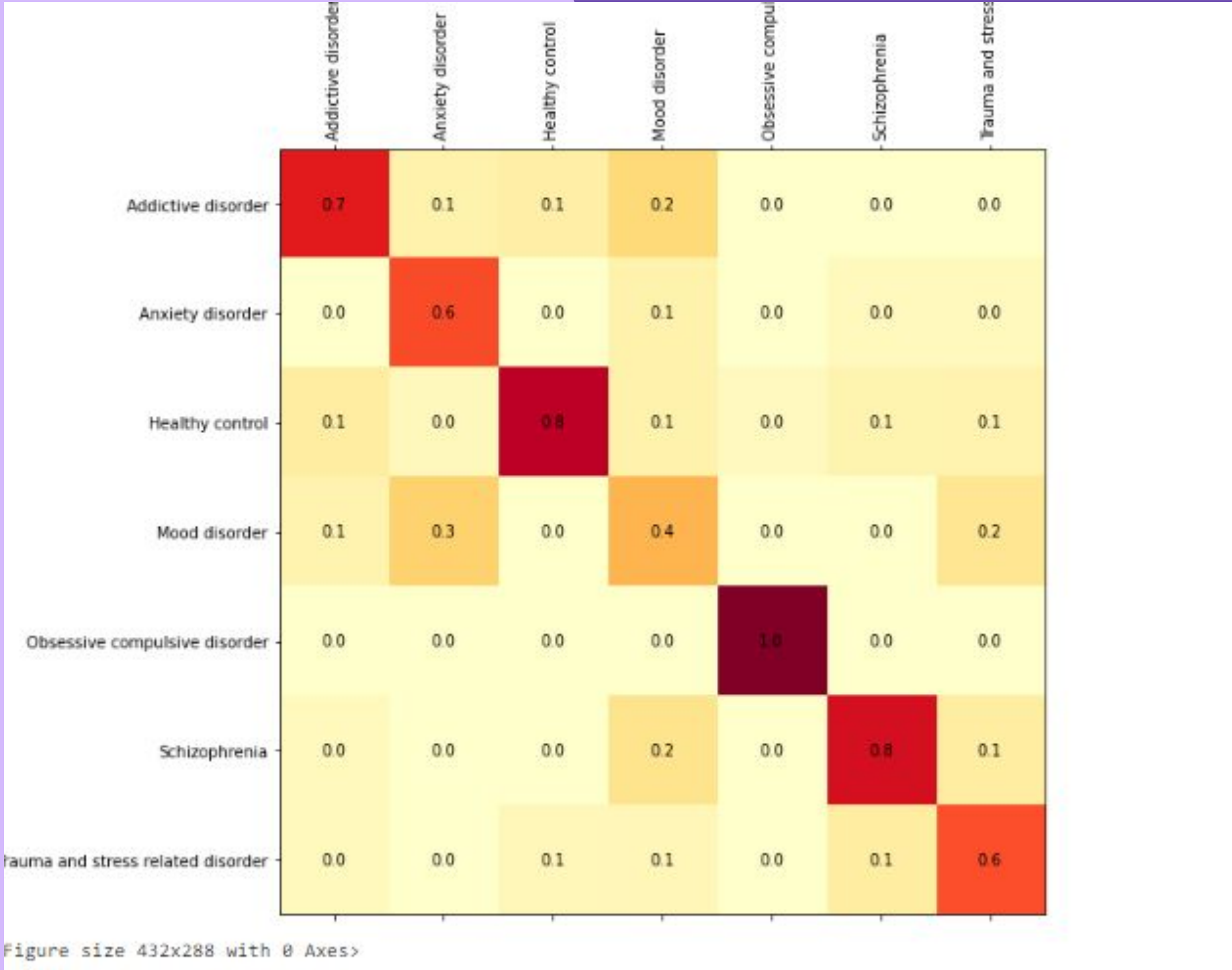
Confusion matrix For Large model on Test dataset For all data

```
print(f"Test_Dataset_Accuracy: {test_accuracy[0]}")
```

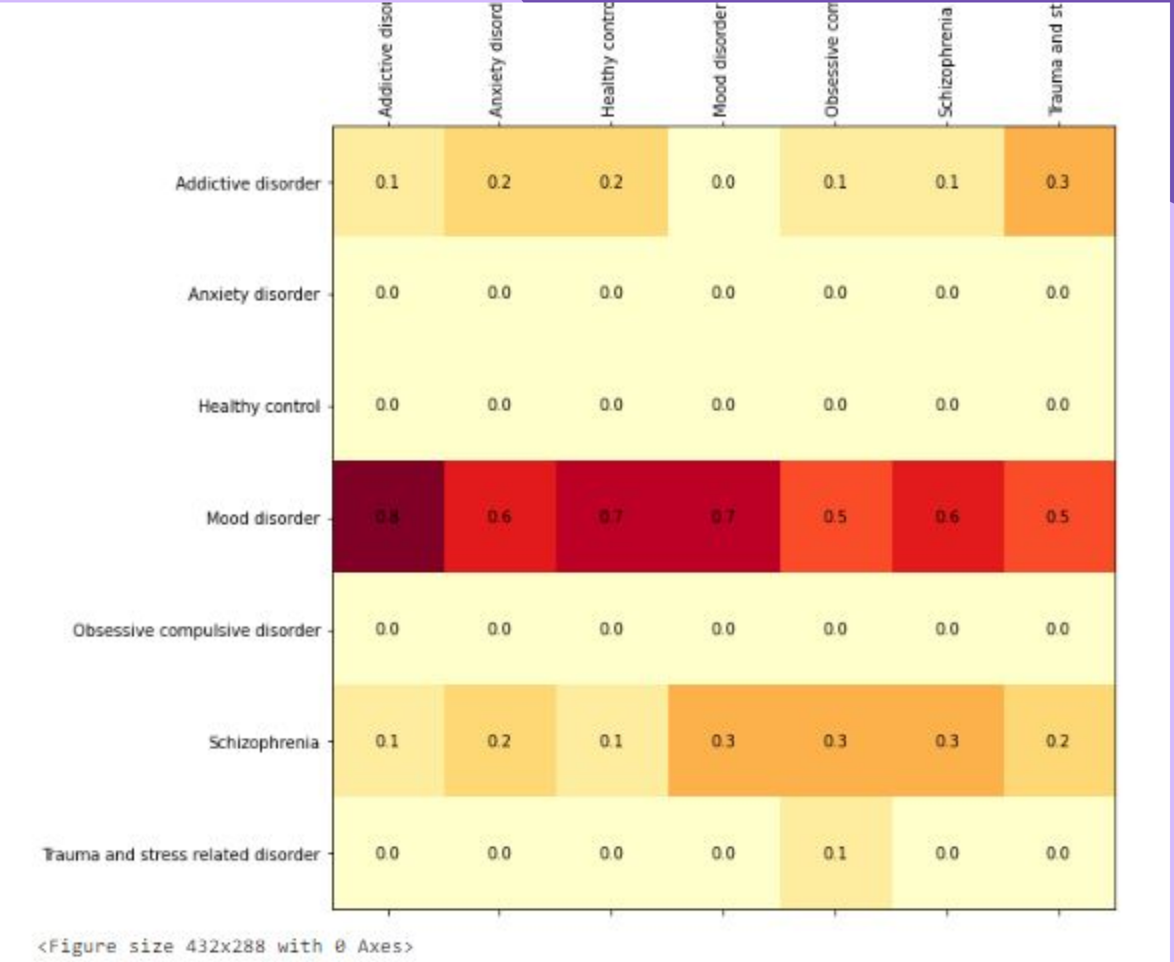
```
Test_Dataset_Accuracy: 21.428571428571427
```



Confusion Matrix on Large model on Validation dataset For data without COH



Confusion matrix of Large model on Test dataset for data without COH




```
print(f"Test_Dataset_Accuracy: {test_accuracy[0]}")
```

```
Test_Dataset_Accuracy: 15.714285714285714
```

07.

Result and Conclusion

- Machine Learning model did not help us to classify on training and testing dataset.
- Deep Learning approach seems to be little promising than machine learning approach due to More number of features that can be learnt through a convolutional and Linear layer.
- Still deep learning model suffers from less dataset.

08.

Future Work

- More experiment can be done on making Deep Learning Approach

- Experiment on data sampling can be done

- Limit the scope of classification to fewer classes.

Reference

- Park S. M, Jeong B, Oh D. Y, Choi CH, Jung H. Y, Lee JY, Lee D, Choi JS. Identification of Major Psychiatric Disorders From Resting-State Electroencephalography Using a Machine Learning Approach. Frontiers in Psychiatry. (2021). doi: 10.3389/fpsyt.2021.707581

THANK YOU