Classification of EEG Signals during Mental Arithemetic tasks

In [45]:

```
import mne
import numpy as np
import pywt
import matplotlib.pyplot as plt
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
```

The following function gen_eegdata() generates the feature matrix of EEG data by the following steps :

1) It reads the EEG data from all channels for a particular subject during the time of mental arithemetic tasks. (Doesn't care about the EEG data before mental arithmetic task)

2) Using Discrete Wavelet Transformation (DWT) , using high pass and low pass filters it downsamples the timeseries data of all channels of a certain subject .

3) Wavelet energy fr each decomposition level is then calculated.

4) Total energy is the sum of all those energies.

5) Repeated the steps 1-4 for all subjects and return a 2-D numpy arrray as dataset.

In [46]:

```
def gen_eegdata():

    dataset = np.array([])

    for i in range(0,36):

        file = 'EEG_DATA/Subject{:02}_2.edf'.format(i)
        data = mne.io.read_raw_edf(file, verbose = False)
        raw_data = data.get_data()

        (A1,D1) = pywt.dwt(raw_data,'db4')
        (A2,D2) = pywt.dwt(A1,'db4')
        (A3,D3) = pywt.dwt(A2,'db4')
        (A4,D4) = pywt.dwt(A3,'db4')
        (A5,D5) = pywt.dwt(A4,'db4')

        ED1 = np.sum(np.abs(D1)**2,axis = 1,keepdims=True)
        ED2 = np.sum(np.abs(D2)**2,axis = 1,keepdims=True)
        ED3 = np.sum(np.abs(D3)**2,axis = 1,keepdims=True)
        ED4 = np.sum(np.abs(D4)**2,axis = 1,keepdims=True)
        ED5 = np.sum(np.abs(D5)**2,axis = 1,keepdims=True)

        EA5 = np.sum(np.abs(A5)**2,axis = 1,keepdims=True)

        E_total = EA5 + ED1+ED2+ED3+ED4+ED5
        E_total = E_total[0:20,:].T

        if len(dataset)==0:
            dataset = E_total
        else:
            dataset = np.concatenate((dataset,E_total))

    return dataset
```

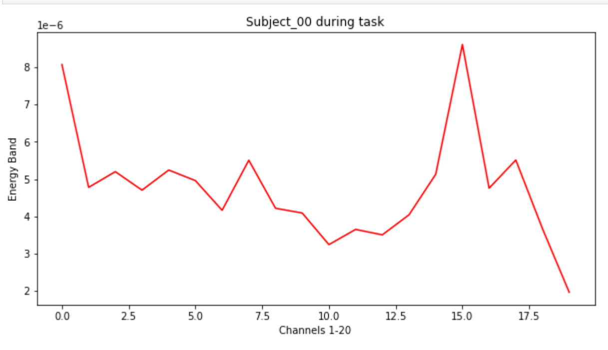
In [47]:

```
dataset = gen_eegdata()
```

Plotting the Total Energy band across all channels for two subjects

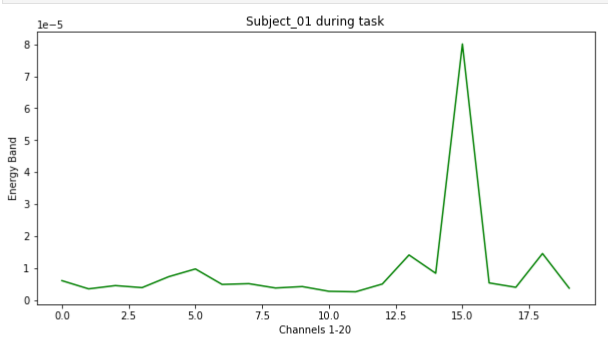
In [48]:

```
plt.figure(figsize=(10,5))
plt.plot(dataset[0,:],'r')
plt.xlabel('Channels 1-20')
plt.ylabel('Energy Band')
plt.title('Subject_00 during task')
plt.show()
```



In [49]:

```
plt.figure(figsize=(10,5))
plt.plot(dataset[1,:], 'g')
plt.xlabel('Channels 1-20')
plt.ylabel('Energy Band')
plt.title('Subject_01 during task')
plt.show()
```



In [50]:

```
dataset.shape
```

Out[50]:

```
(36, 20)
```

Since the dataset has 20 features after DWT so used Principle Component Analysis for feature selection and to reduce the dimensionality assuming that the 5 components carries the essence of Gamma, Beta, Alpha, Theta and Delta band energies.

In [51]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 5)
dataset = pca.fit_transform(dataset)
```

In [52]:

```
dataset.shape
```

Out[52]:

```
(36, 5)
```

From the subject information data, took the relevant features like Age,Number of subtractions, Gender and also the feature vector Count quality

In [53]:

```
subject_data = pd.read_csv('EEG_DATA/subject-info.csv')
subject_data = subject_data.drop(columns=['Subject','Recording year'])
subject_data = pd.get_dummies(subject_data, drop_first=True)
```

In [54]:

```
subject_data.head()
```

Out[54]:

	Age	Number of subtractions	Count quality	Gender_M
0	21	9.70	0	0
1	18	29.35	1	0
2	19	12.88	1	0
3	17	31.00	1	0
4	17	8.60	0	0

In [55]:

```
X = subject_data.drop(columns = ['Count quality']).to_numpy()
y =subject_data[['Count quality']].to_numpy()
```

Concatenated the above feature matrix with the EEG feature matrix.

In [56]:

```
X_new = np.concatenate((dataset,X),axis = 1)
```

Split the dataset into training set and test set

In [57]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_new,y,test_size = 0.25)
```

Scaled the dataset for better performance

In [58]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
sc.fit(X_train)
```

Out[58]:

```
StandardScaler()
```

In [59]:

```
X_train_sc = sc.transform(X_train)
X_test_sc  = sc.transform(X_test)
```

In [60]:

```
import tensorflow as tf
```

The dense neural network consists of 3 layers , two hiddden layers of 10 neurons each and an output layer with one neuron.

In [61]:

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(units = 10, activation='relu'))
model.add(tf.keras.layers.Dense(units = 10, activation='relu'))
model.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid'))
```

In [62]:

```
# adam optimizer and binary_crossentropy loss was found to be best match for the model.
model.compile(optimizer = 'adam', loss = 'binary_crossentropy')
```

In [63]:

```
model.fit(X_train_sc,y_train,batch_size = 1,epochs=100)
```

```
Epoch 1/100
27/27 [==============================] - 1s 2ms/step - loss: 0.6561
Epoch 2/100
27/27 [==============================] - 0s 2ms/step - loss: 0.6177
Epoch 3/100
27/27 [==============================] - 0s 2ms/step - loss: 0.5829
Epoch 4/100
27/27 [==============================] - 0s 2ms/step - loss: 0.5498
Epoch 5/100
27/27 [==============================] - 0s 2ms/step - loss: 0.5131
Epoch 6/100
27/27 [==============================] - 0s 2ms/step - loss: 0.4839
Epoch 7/100
27/27 [==============================] - 0s 2ms/step - loss: 0.4525
Epoch 8/100
27/27 [==============================] - 0s 2ms/step - loss: 0.4238
Epoch 9/100
27/27 [==============================] - 0s 2ms/step - loss: 0.3955
Epoch 10/100
27/27 [==============================] - 0s 1ms/step - loss: 0.3726
Epoch 11/100
27/27 [==============================] - 0s 2ms/step - loss: 0.3501
Epoch 12/100
27/27 [==============================] - 0s 2ms/step - loss: 0.3283
Epoch 13/100
27/27 [==============================] - ETA: 0s - loss: 0.169 - 0s 2ms/step - loss: 0.3094
Epoch 14/100
27/27 [==============================] - 0s 2ms/step - loss: 0.2925
Epoch 15/100
27/27 [==============================] - 0s 2ms/step - loss: 0.2761
Epoch 16/100
27/27 [==============================] - 0s 2ms/step - loss: 0.2638
Epoch 17/100
27/27 [==============================] - 0s 2ms/step - loss: 0.2511
Epoch 18/100
27/27 [==============================] - 0s 1ms/step - loss: 0.2387
Epoch 19/100
27/27 [==============================] - 0s 2ms/step - loss: 0.2289
Epoch 20/100
27/27 [==============================] - 0s 2ms/step - loss: 0.2176
Epoch 21/100
27/27 [==============================] - 0s 2ms/step - loss: 0.2095
Epoch 22/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1995
Epoch 23/100
27/27 [==============================] - 0s 1ms/step - loss: 0.1921
Epoch 24/100
27/27 [==============================] - 0s 1ms/step - loss: 0.1841
Epoch 25/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1755
Epoch 26/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1676
Epoch 27/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1625
Epoch 28/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1539
Epoch 29/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1489
Epoch 30/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1441
Epoch 31/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1376
Epoch 32/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1337
Epoch 33/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1310
Epoch 34/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1255
Epoch 35/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1224
Epoch 36/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1184
Epoch 37/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1116
Epoch 38/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1083
Epoch 39/100
27/27 [==============================] - 0s 2ms/step - loss: 0.1043
Epoch 40/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0997
Epoch 41/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0976
Epoch 42/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0937
Epoch 43/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0911
Epoch 44/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0883
Epoch 45/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0836
Epoch 46/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0824
Epoch 47/100
27/27 [==============================] - 0s 1ms/step - loss: 0.0789
Epoch 48/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0765
Epoch 49/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0754
Epoch 50/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0721
Epoch 51/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0699
Epoch 52/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0689
Epoch 53/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0671
Epoch 54/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0635
Epoch 55/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0615
Epoch 56/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0594
Epoch 57/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0593
Epoch 58/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0559
Epoch 59/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0553
Epoch 60/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0540
Epoch 61/100
27/27 [==============================] - 0s 1ms/step - loss: 0.0520
Epoch 62/100
27/27 [==============================] - 0s 1ms/step - loss: 0.0490
Epoch 63/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0487
Epoch 64/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0458
Epoch 65/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0455
Epoch 66/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0441
Epoch 67/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0419
Epoch 68/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0402
Epoch 69/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0403
Epoch 70/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0378
Epoch 71/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0369
Epoch 72/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0359
Epoch 73/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0342
Epoch 74/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0339
Epoch 75/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0330
Epoch 76/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0318
Epoch 77/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0308
Epoch 78/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0296
Epoch 79/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0294
Epoch 80/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0293
Epoch 81/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0268
Epoch 82/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0273
Epoch 83/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0266
Epoch 84/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0242
Epoch 85/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0242
Epoch 86/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0232
Epoch 87/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0227
Epoch 88/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0227
Epoch 89/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0213
Epoch 90/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0199
Epoch 91/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0192
Epoch 92/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0186
Epoch 93/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0189
Epoch 94/100
27/27 [==============================] - 0s 1ms/step - loss: 0.0180
Epoch 95/100
27/27 [==============================] - 0s 3ms/step - loss: 0.0174
Epoch 96/100
27/27 [==============================] - 0s 3ms/step - loss: 0.0177
Epoch 97/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0164
Epoch 98/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0159
Epoch 99/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0152
Epoch 100/100
27/27 [==============================] - 0s 2ms/step - loss: 0.0153
```

Out[63]:

```
&lt;keras.callbacks.History at 0x23eb465d8e0&gt;
```

In [64]:

```
y_pred = model.predict(X_test_sc)
y_pred = (y_pred>0.5)
```

From the confusion matrix it is seen that out of 9 predictions 8 were correct and with an accuracy score of 89%

In [65]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test,y_pred)
print(cm)
```

```
[[3 1]
 [0 5]]
```

In [66]:

```
acc = accuracy_score(y_test,y_pred)
print('Accuracy of prediction : {:.2f} %'.format(acc*100))
```

```
Accuracy of prediction : 88.89 %
```