In this file I tried to pick the best 12 features and train my model on those and see if the results which I will get are good enough or not. The reason to do this is : 1)It helps to train model faster and Keeping the pilot's real time data processing in mind if we can reduce the number of features and still get best result then that will be very beneficial. 2)It reduces model complexity. 3)It improves model accuracy if proper features are chosen.

In [ ]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (X11; Li
nux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36" --
header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-GB,en-
US;q=0.9,en;q=0.8" --header="Referer: https://www.kaggle.com/" "https://storage.googleapi
s.com/kaggle-competitions-data/kaggle-v2/11835/224935/compressed/train.csv.zip?GoogleAcce
ssId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1596623187&Signature=doLtiDv5
Dhc5VY2IffVsHC14bXwc%2B82Kt9BPSLU%2B3MsBqmC3C5Bx93D%2FNkZ4DhkbkwodBG3wEGRfR4aYcl2oxTaz2dX
VQ4D5a3H3dIkddAXBj554IN4%2F0sWol8CtZrdxIVTzYiyPjTsjhw%2FZu0okgowsXCLZ1lIlxp5g%2BTEiJTxJnQ
bwKzCO4kWmbRnpoQCWN1FUJZ2veJPPISPrF0FluI%2BUKzGqwIHnM3A6rOQ6H2Oi0oazUBps9KcZI4moL6Qs0Lbv4
dsFLE%2BtsBpa3IHGxPJIUfqPDfqx0hFMk875%2FB3N8sBvs2IYdYMzZZpR0WN559bpgu3%2Fq%2Bx7DPcs4mf7Og
%3D%3D&response-content-disposition=attachment%3B+filename%3Dtrain.csv.zip" -c -O 'train.
csv.zip'
```

```
--2020-08-04 10:53:00--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-v
2/11835/224935/compressed/train.csv.zip?GoogleAccessId=web-data@kaggle-161607.iam.gservic
eaccount.com&Expires=1596623187&Signature=doLtiDv5Dhc5VY2IffVsHC14bXwc%2B82Kt9BPSLU%2B3Ms
BqmC3C5Bx93D%2FNkZ4DhkbkwodBG3wEGRfR4aYcl2oxTaz2dXVQ4D5a3H3dIkddAXBj554IN4%2F0sWol8CtZrdx
IVTzYiyPjTsjhw%2FZu0okgowsXCLZ1lIlxp5g%2BTEiJTxJnQbwKzCO4kWmbRnpoQCWN1FUJZ2veJPPISPrF0Flu
I%2BUKzGqwIHnM3A6rOQ6H2Oi0oazUBps9KcZI4moL6Qs0Lbv4dsFLE%2BtsBpa3IHGxPJIUfqPDfqx0hFMk875%2
FB3N8sBvs2IYdYMzZZpR0WN559bpgu3%2Fq%2Bx7DPcs4mf7Og%3D%3D&response-content-disposition=att
achment%3B+filename%3Dtrain.csv.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.28.128, 74.125.142.12
8, 74.125.195.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.28.128|:443... conne
cted.
HTTP request sent, awaiting response... 200 OK
Length: 456337398 (435M) [application/zip]
Saving to: ‘train.csv.zip’

train.csv.zip       100%[===================>] 435.20M   299MB/s    in 1.5s

2020-08-04 10:53:02 (299 MB/s) - ‘train.csv.zip’ saved [456337398/456337398]
```

In [ ]:

```
!unzip train.csv.zip
```

```
Archive:  train.csv.zip
  inflating: train.csv
```

In [ ]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (X11; Li
nux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36" --
header="Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9" --header="Accept-Language: en-GB,en-
US;q=0.9,en;q=0.8" --header="Referer: https://www.kaggle.com/" "https://storage.googleapi
s.com/kaggle-competitions-data/kaggle-v2/11835/224935/compressed/test.csv.zip?GoogleAcces
sId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1596623324&Signature=Jh3PPYb9p
RvKZEhcxQC04c0wCuhcialw85rMmtsEvJLWvxNX97iA%2BLVAlbstk19TV4HPqMq99YUL%2BFlxzLeapjc5lLtjL2
OjmPZfm9B9prFKkfxvpn88txS%2FedMxPJEkMhHdUVpDNVaLf0Yks3jmaCN3lKcIflmteHphPDnOwLkakEQsynXuC
%2FpAB9%2F6kIw5XAKUUvIfJgMnSHxbFNf3iqjNZKPr3wBL%2F4T4EUM0Tv9W2wcG69Vb5iMuZMND8z3fXaQRcNgc
tSYaE1%2FcOtZJU20nX%2F%2B9VO8AVoGi4j4mjU%2FCUckXak0PmdiFMr9mQosYdEMUg9LodKerKNS4KyzsgQ%3D
%3D&response-content-disposition=attachment%3B+filename%3Dtest.csv.zip" -c -O 'test.csv.z
ip'
```

```
--2020-08-04 12:30:41--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-v
2/11835/224935/compressed/test.csv.zip?GoogleAccessId=web-data@kaggle-161607.iam.gservice
account.com&Expires=1596623324&Signature=Jh3PPYb9pRvKZEhcxQC04c0wCuhcialw85rMmtsEvJLWvxNX
```

```
97iA%2BLVAlbstk19TV4HPqMq99YUL%2BFlxzLeapjc5lLtjL2OjmPZfm9B9prFKkfxvpn88txS%2FedMxPJEkMhH
dUVpDNVaLf0Yks3jmaCN3lKcIflmteHphPDnOwLkakEQsynXuC%2FpAB9%2F6kIw5XAKUUvIfJgMnSHxbFNf3iqjN
ZKPr3wBL%2F4T4EUM0Tv9W2wcG69Vb5iMuZMND8z3fXaQRcNgctSYaE1%2FcOtZJU20nX%2F%2B9VO8AVoGi4j4mj
U%2FCUckXak0PmdiFMr9mQosYdEMUg9LodKerKNS4KyzsgQ%3D%3D&response-content-disposition=attach
ment%3B+filename%3Dtest.csv.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.142.128, 74.125.195.1
28, 173.194.202.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.142.128|:443... conn
ected.
HTTP request sent, awaiting response... 200 OK
Length: 1791131386 (1.7G) [application/zip]
Saving to: ‘test.csv.zip’

test.csv.zip        100%[===================>]   1.67G  70.6MB/s    in 23s

2020-08-04 12:31:04 (73.2 MB/s) - ‘test.csv.zip’ saved [1791131386/1791131386]
```

In [ ]:

```
! unzip test.csv
```

```
Archive:  test.csv.zip
  inflating: test.csv
```

In [ ]:

```python
import warnings
import itertools
import numpy as np
import pandas as pd
import seaborn as sns
import lightgbm as lgb
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook as tqdm
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, log_loss
import dask.dataframe as dd
import dask
import gc

from yellowbrick.text import TSNEVisualizer

%matplotlib inline
plt.style.use("fivethirtyeight")

# import os
# print(os.listdir("../input"))

warnings.filterwarnings(action='ignore')
sns.set_style('whitegrid')
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: p
andas.util.testing is deprecated. Use the functions in the public API at pandas.testing i
nstead.
  import pandas.util.testing as tm
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: T
he sklearn.metrics.classification module is  deprecated in version 0.22 and will be remov
ed in version 0.24. The corresponding classes / functions should instead be imported from
sklearn.metrics. Anything that cannot be imported from sklearn.metrics is now part of the
private API.
  warnings.warn(message, FutureWarning)
```

In [ ]:

```python
# This is to be used for memory optimization because the data is very large.
# For the working of iinfo function refer- https://numpy.org/doc/stable/reference/generat
ed/numpy.iinfo.html
# So in this we typically take the max and min value of each feature and convert it into
respective size to reduce memory usage
```

```
def reduce_mem_usage(df):
    """ iterate through all the columns of a dataframe and modify the data type
        to reduce memory usage.
    """
    start_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max
:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).m
ax:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
        else:
            df[col] = df[col].astype('category')

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))

    return df
# FeatureModify function encodes the categorical feature into numeric form.
def featureModify(isTrain, numRows):
    if isTrain:
        df = dd.read_csv('train.csv',nrows=numRows)
        df = df.compute()
        df = reduce_mem_usage(df)
        df['event'] = df['event'].map({
            'A':0,
            'B':1,
            'C':2,
            'D':3
        })
    else:
        df = dd.read_csv('test.csv',nrows=numRows)
        df = df.compute()
        df = reduce_mem_usage(df)

    return df
train = featureModify(True, None)
y = train['event']
# train = train.drop('event',axis=1)
print(train.shape)
print(train.columns)
```

```
Memory usage of dataframe is 1076.93 MB
Memory usage after optimization is: 278.52 MB
Decreased by 74.1%
(4867421, 28)
Index(['crew', 'experiment', 'time', 'seat', 'eeg_fp1', 'eeg_f7', 'eeg_f8',
       'eeg_t4', 'eeg_t6', 'eeg_t5', 'eeg_t3', 'eeg_fp2', 'eeg_o1', 'eeg_p3',
       'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_p4', 'eeg_poz',
       'eeg_c3', 'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr', 'event'],
      dtype='object')
```

In [ ]:

```
train.head()
```

Out[ ]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	
0	1	CA	0.011719	1	-5.285156	26.781250	-9.523438	-12.796875	16.718750	33.75000	23.718750	-6.695312	2
1	1	CA	0.015625	1	-2.427734	28.437500	-9.320312	-3.757812	15.968750	30.43750	21.015625	-6.476562	2
2	1	CA	0.019531	1	10.671875	30.421875	15.351562	24.718750	16.140625	32.15625	25.437500	-0.088684	2
3	1	CA	0.023438	1	11.453125	25.609375	2.433594	12.414062	20.531250	31.50000	19.140625	-0.256592	3
4	1	CA	0.027344	1	7.285156	25.937500	0.113586	5.746094	19.828125	28.75000	20.578125	-1.953125	3

In [ ]:

```
train['pilot'] = 100*train['seat']+train['crew']
```

In [ ]:

```
train = train[['gsr','r','ecg','crew','eeg_fp2','pilot','eeg_f7','eeg_f8','eeg_fp1','eeg_
pz','eeg_f4','eeg_f3']]
```

In [ ]:

```
# splitiing the data into train and test
# gc is used to collect the garbage
train, train_test, y, y_test = train_test_split(train, y, test_size=0.25, shuffle=True)
train = lgb.Dataset(train, label=y,categorical_feature=[1])
del y
gc.collect()


train_test = lgb.Dataset(train_test, label=y_test,categorical_feature=[1])
del y_test
gc.collect()
```

Out[ ]:

```
0
```

In [ ]:

```
params = {
        "objective" : "multiclass",
        "metric" : "multi_error",
        'num_class':4,
        "num_leaves" : 30,
        "learning_rate" : 0.01,
        "bagging_fraction" : 0.9,
        "bagging_seed" : 0,
        "num_threads" : 4,
        'min_data_in_leaf':100,
        'min_split_gain':0.00019
}

model = lgb.train(  params,
                    train_set = train,
                    num_boost_round=1000,
                    early_stopping_rounds=200,
                    verbose_eval=100,
                    valid_sets=[train,train_test]
                  )
```

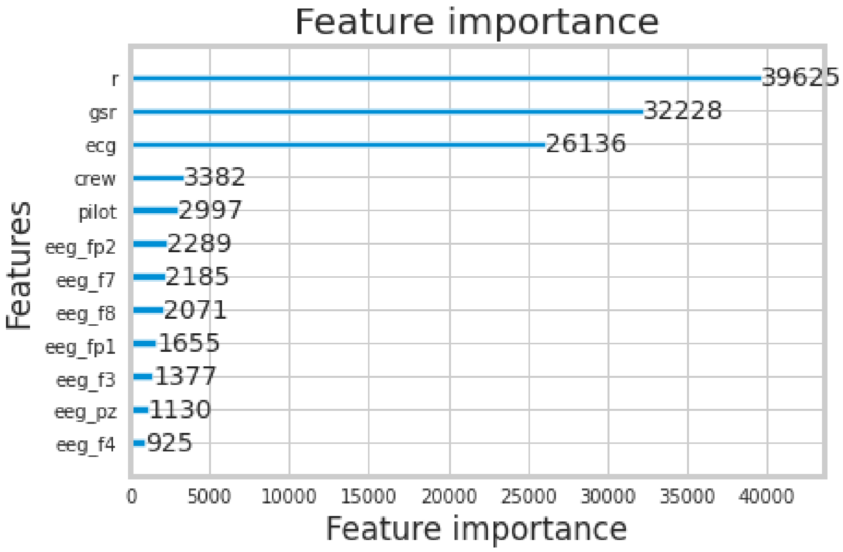
```
Training until validation scores don't improve for 200 rounds.
[100] training's multi_error: 0.139957 valid_1's multi_error: 0.140022
[200] training's multi_error: 0.121732 valid_1's multi_error: 0.121863
[300] training's multi_error: 0.111193 valid_1's multi_error: 0.111311
[400] training's multi_error: 0.0988633 valid_1's multi_error: 0.0991539
[500] training's multi_error: 0.0939383 valid_1's multi_error: 0.0942264
[600] training's multi_error: 0.0894708 valid_1's multi_error: 0.0897312
[700] training's multi_error: 0.0849679 valid_1's multi_error: 0.0852689
[800] training's multi_error: 0.0816068 valid_1's multi_error: 0.0818371
[900] training's multi_error: 0.0789856 valid_1's multi_error: 0.079269
[1000] training's multi_error: 0.0767895 valid_1's multi_error: 0.0770905
Did not meet early stopping. Best iteration is:
[1000] training's multi_error: 0.0767895 valid_1's multi_error: 0.0770905
```

In [ ]:

```
# Checking the feature importance.
lgb.plot_importance(model)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4701813710>
```



In [ ]:

```
lgb.create_tree_digraph(model)
```

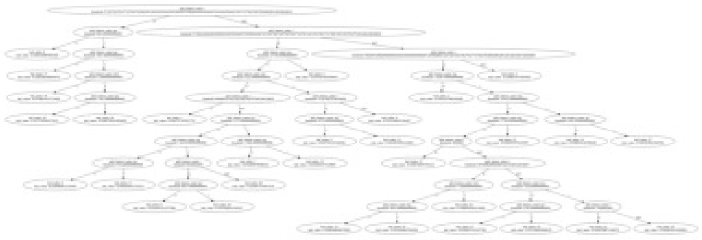
Out[ ]:

In [ ]:

```
lgb.plot_tree(model)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f47018242b0>
```



In [ ]:

```
# Preparing the testing data for prediction
test = featureModify(False, None)
print("Done test read")
```

```
Memory usage of dataframe is 3974.83 MB
Memory usage after optimization is: 1079.37 MB
Decreased by 72.8%
Done test read
```

In [ ]:

```
test.head()
```

Out[ ]:

	id	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg
0	0	1	LOFT	0.000000	0	17.906250	6.128906	0.994629	-28.203125	-47.687500	-187.125000	-33.187500	-4.22
1	1	1	LOFT	0.000000	1	45.875000	94.750000	23.296875	1.391602	2.060547	-5.144531	6.394531	33.40
2	2	1	LOFT	0.003906	0	33.125000	28.359375	-7.238281	-7.691406	-25.828125	-107.250000	12.843750	1.21
3	3	1	LOFT	0.003906	1	43.281250	95.875000	18.703125	-1.432617	-4.234375	-8.023438	7.425781	27.34
4	4	1	LOFT	0.007812	0	7.929688	3.460938	-10.859375	-26.359375	-25.890625	37.000000	-50.343750	11.67

In [ ]:

```
# storing the ids column
df_sub = pd.DataFrame()
df_sub['id'] = test['id']
test = test.drop('id',axis=1)
```

In [ ]:

```
test['pilot']= 100*test['crew']+test['seat']
```

In [ ]:

```
#Selecting the top
test = test[['gsr','r','ecg','crew','eeg_fp2','pilot','eeg_f7','eeg_f8','eeg_fp1','eeg_pz
','eeg_f4','eeg_f3']]
```

In [ ]:

```
test.head()
```

Out[ ]:

	gsr	r	ecg	crew	eeg_fp2	pilot	eeg_f7	eeg_f8	eeg_fp1	eeg_pz	eeg_f4	eeg_f3
0	595.00	643.0	-7324.0	1	-4.222656	100	6.128906	0.994629	17.906250	33.812500	-7.042969	21.750000
1	136.25	826.5	-3336.0	1	33.406250	101	94.750000	23.296875	45.875000	29.640625	19.890625	16.218750
2	595.00	643.0	-7324.0	1	1.214844	100	28.359375	-7.238281	33.125000	37.593750	-7.640625	29.078125
3	136.25	826.5	-3336.0	1	27.343750	101	95.875000	18.703125	43.281250	27.734375	13.828125	7.218750
4	595.00	643.0	-7324.0	1	-11.679688	100	3.460938	-10.859375	7.929688	34.062500	2.044922	22.906250

In [ ]:

```
y_pred = model.predict(test, num_iteration=model.best_iteration)
```

In [ ]:

```
y_pred
```

Out[ ]:

```
array([[9.96882908e-01, 1.19782911e-03, 1.01123989e-03, 9.08023300e-04],
       [9.22321676e-01, 1.97908943e-02, 4.61369559e-02, 1.17504742e-02],
       [9.96626437e-01, 1.19752094e-03, 9.98507664e-04, 1.17753426e-03],
       ...,
       [5.51198553e-01, 2.29694339e-01, 1.89156421e-01, 2.99506875e-02],
       [9.84226054e-01, 4.53610778e-04, 1.59946949e-03, 1.37208653e-02],
       [5.50143608e-01, 2.29254725e-01, 1.88794393e-01, 3.18072743e-02]])
```

In [ ]:

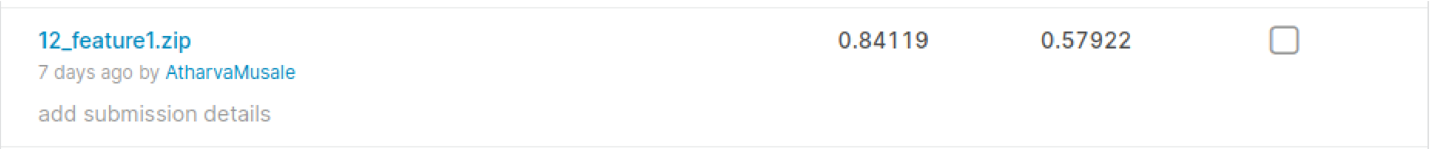
```
df_sub = pd.DataFrame(np.concatenate((np.arange(len(test))[:, np.newaxis], y_pred), axis
=1), columns=['id', 'A', 'B', 'C', 'D'])
df_sub['id'] = df_sub['id'].astype(int)
print(df_sub)
df_sub.to_csv("12_feature.csv", index=False)
```

```
                id         A         B         C         D
0                0  0.996883  0.001198  0.001011  0.000908
1                1  0.922322  0.019791  0.046137  0.011750
2                2  0.996626  0.001198  0.000999  0.001178
3                3  0.923315  0.019588  0.045663  0.011434
4                4  0.996949  0.001173  0.000995  0.000883
...            ...       ...       ...       ...       ...
17965138  17965138  0.580927  0.214643  0.176043  0.028387
17965139  17965139  0.984228  0.000454  0.001599  0.013719
17965140  17965140  0.551199  0.229694  0.189156  0.029951
17965141  17965141  0.984226  0.000454  0.001599  0.013721
17965142  17965142  0.550144  0.229255  0.188794  0.031807

[17965143 rows x 5 columns]
```

Conclusion-

The scores after submission in kaggle which I got are as follows: I got public score of 0.579 and private score of 0.841



12_feature1.zip
7 days ago by AtharvaMusale
add submission details

0.84119 0.57922 □