In this notebook I am trying to pick top 20 features after using 12 features in one of my previous notebook since results were not very bad in the previous notebook I thought that I should try picking out 20 top features with pilot feature making 21 in total and see what are the results using those

In [ ]:

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (X11; Linux 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/apng,*/*;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.googleapis.com/kaggle-competitions-data/kaggle-v2/11835/224935/compressed/train.csv.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1596891716&Signature=MCKpY0KXos%2BWUBwZUS7riqVdW4b7GiykWFS%2BQz3bnMxj8o4KMtv1ieGQZgx2Mtmo1waGXOZxsLdjhdIvRUjVUAGhv6WceEUwRNkfcgdL0c%2BorfGNFEFHekdIEIM6sUtc7dBMyq29aWK%2F0h1V0QN9Uw62kRNHaxO7A8xgXdL9AfBCJCSMwJ2QuqMGduy0wnrOE4WXHiEE99BJhR51kw5bUUi7JAzSSO9g3NapGYJsh%2BXXmLcjONGz%2BrKYZ4vwl49jHpEGKaiO%2B3yfDqW5kLNhCK8GKglxB10Z2ZU091Ltx9SIVhkfGjekgZyOIBpN2x%2F6iS9M0ZDCSOgPbAIFflPGhg%3D%3D&response-content-disposition=attachment%3B+filename%3Dtrain.csv.zip" -c -O 'train.csv.zip'
```

```
--2020-08-06 04:11:26--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-v2/11835/224935/compressed/train.csv.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1596891716&Signature=MCKpY0KXos%2BWUBwZUS7riqVdW4b7GiykWFS%2BQz3bnMxj8o4KMtv1ieGQZgx2Mtmo1waGXOZxsLdjhdIvRUjVUAGhv6WceEUwRNkfcgdL0c%2BorfGNFEFHekdIEIM6sUtc7dBMyq29aWK%2F0h1V0QN9Uw62kRNHaxO7A8xgXdL9AfBCJCSMwJ2QuqMGduy0wnrOE4WXHiEE99BJhR51kw5bUUi7JAzSSO9g3NapGYJsh%2BXXmLcjONGz%2BrKYZ4vwl49jHpEGKaiO%2B3yfDqW5kLNhCK8GKglxB10Z2ZU091Ltx9SIVhkfGjekgZyOIBpN2x%2F6iS9M0ZDCSOgPbAIFflPGhg%3D%3D&response-content-disposition=attachment%3B+filename%3Dtrain.csv.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 173.194.69.128, 173.194.79.128, 108.177.119.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|173.194.69.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 456337398 (435M) [application/zip]
Saving to: ‘train.csv.zip’

train.csv.zip       100%[===================>] 435.20M  70.6MB/s    in 7.2s

2020-08-06 04:11:34 (60.8 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; Linux 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/apng,*/*;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.googleapis.com/kaggle-competitions-data/kaggle-v2/11835/224935/compressed/test.csv.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1596891757&Signature=bjWisHQtx8jjPi2pj6S4wSnRT%2FhKcPZW4wjJWGbh8gYHBeSqI1jKA0B4DYZuf7VfkaiRAzUcUcPIBJjOrzPVU3pvPHjrwzYU9VgWOcWfQF4vm7zHnjxpnfcwCWO8BenLgGLwK9%2B9BmNCHKBV5R6DE%2BwZvfaraCVHKM5mUwne9MXhR7VoaFVAHAh4T%2B3W7ibgqgzaU2ycBxSA8eE3nWZBCuPcXts9XyLAE8ZKqvQhCATBjE3hsz8eKDCwTUtbBU9oxc8e5WE%2Fn0ZkLw6pd9OYFXZSDlNj5Rje%2BVTSJIiH9Vuln7AihRSObRvFRoIfd2V0fAgQz3LncqEukE1bGAkXuA%3D%3D&response-content-disposition=attachment%3B+filename%3Dtest.csv.zip" -c -O 'test.csv.zip'
```

```
--2020-08-06 04:11:53--  https://storage.googleapis.com/kaggle-competitions-data/kaggle-v2/11835/224935/compressed/test.csv.zip?GoogleAccessId=web-data@kaggle-161607.iam.gserviceaccount.com&Expires=1596891757&Signature=bjWisHQtx8jjPi2pj6S4wSnRT%2FhKcPZW4wjJWGbh8gYHBeSqI1jKA0B4DYZuf7VfkaiRAzUcUcPIBJjOrzPVU3pvPHjrwzYU9VgWOcWfQF4vm7zHnjxpnfcwCWO8BenLgGLwK9%2B9BmNCHKBV5R6DE%2BwZvfaraCVHKM5mUwne9MXhR7VoaFVAHAh4T%2B3W7ibgqgzaU2ycBxSA8eE3nWZBCuPcXt
```

```
s9XyLAE8ZKqvQhCATBjE3hsz8eKDCwTUtbBU9oxc8e5WE%2Fn0ZkLw6pd9OYFXZSDlNj5Rje%2BVTSJIiH9Vuln7A
ihRSObRvFRoIfd2V0fAgQz3LncqEukE1bGAkXuA%3D%3D&response-content-disposition=attachment%3B+
filename%3Dtest.csv.zip
Resolving storage.googleapis.com (storage.googleapis.com)... 108.177.126.128, 108.177.127
.128, 2a00:1450:4013:c01::80, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|108.177.126.128|:443... con
nected.
HTTP request sent, awaiting response... 200 OK
Length: 1791131386 (1.7G) [application/zip]
Saving to: 'test.csv.zip'

test.csv.zip        100%[===================>]   1.67G  18.3MB/s    in 37s

2020-08-06 04:12:31 (46.8 MB/s) - 'test.csv.zip' saved [1791131386/1791131386]
```

In [ ]:

```
! unzip test.csv
```

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

In [ ]:

```
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 [ ]:

```
# This is to be used for memory optimization because the data is very large.
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
def featureModify(isTrain, numRows):
    if isTrain:
        df = dd.read_csv('train.csv',nrows=numRows)
        df = df.compute()
        # df['pilot'] = 100*df['crew']+df['seat']
        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['pilot'] = 100*df['crew']+df['seat']
        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)
```

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
													-

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	
2	1	CA	0.019531	1	10.671875	30.421875	15.351562	24.718750	16.140625	32.15625	25.437500	0.098634	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['crew']+train['seat']
```

In [ ]:

```
train = train[['crew', 'eeg_fp1', 'eeg_f7', 'eeg_f8',
       'eeg_t6', 'eeg_t3', 'eeg_fp2', 'eeg_o1','eeg_c3',
       'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_f4', 'eeg_c4', 'eeg_poz',
       'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr','pilot']]
```

In [ ]:

```
# from sklearn.preprocessing import MinMaxScaler
# ms = MinMaxScaler()
# train['pilot'] = ms.fit_transform(np.array(train['pilot']).reshape(-1,1))
```

In [ ]:

```
train, train_test, y, y_test = train_test_split(train, y, test_size=0.2, 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: 0.128614 valid_1's multi_error: 0.129043
[200] training's multi_error: 0.105809 valid_1's multi_error: 0.106007
[300] training's multi_error: 0.100033 valid_1's multi_error: 0.100435
[400] training's multi_error: 0.0960992 valid_1's multi_error: 0.0965582
[500] training's multi_error: 0.0933051 valid_1's multi_error: 0.0938659
[600] training's multi_error: 0.0898893 valid_1's multi_error: 0.0904328
[700] training's multi_error: 0.0863255 valid_1's multi_error: 0.0868498
```

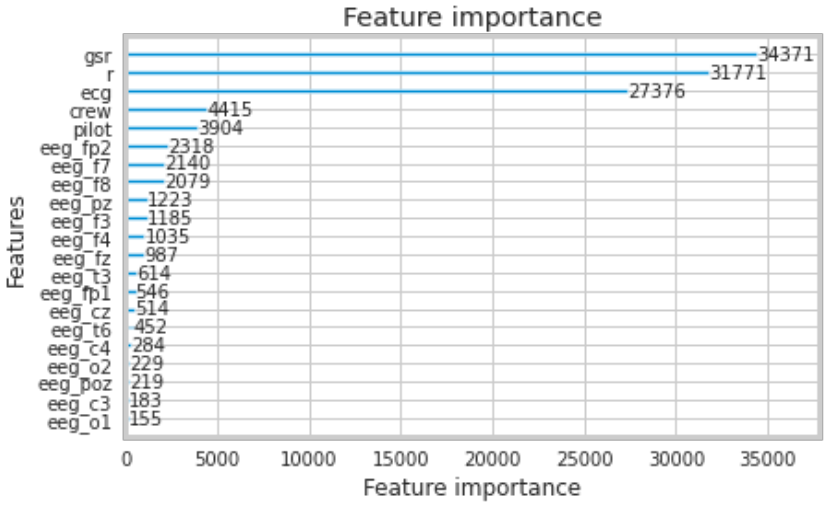
```
[800] training's multi_error: 0.0836259 valid_1's multi_error: 0.0842098
[900] training's multi_error: 0.0813986 valid_1's multi_error: 0.0819674
[1000] training's multi_error: 0.0787329 valid_1's multi_error: 0.0793222
Did not meet early stopping. Best iteration is:
[1000] training's multi_error: 0.0787329 valid_1's multi_error: 0.0793222
```

In [ ]:

```
lgb.plot_importance(model)
```

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f958feaa438>
```



In [ ]:

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

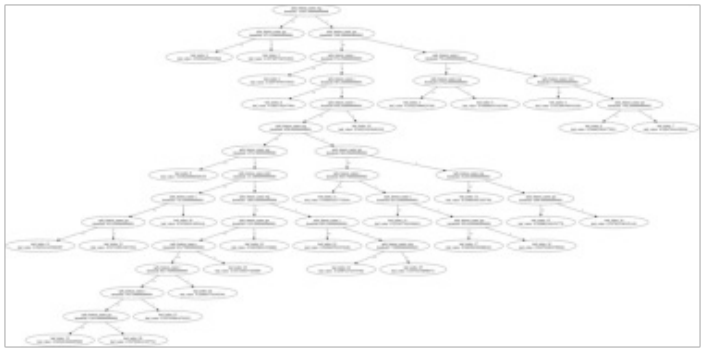
Out[ ]:

In [ ]:

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

Out[ ]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9573457588>
```



In [ ]:

```
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 [ ]:

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

In [ ]:

```
test = test[['crew', 'eeg_fp1', 'eeg_f7', 'eeg_f8',
       'eeg_t6', 'eeg_t3', 'eeg_fp2', 'eeg_o1','eeg_c3',
       'eeg_pz', 'eeg_f3', 'eeg_fz', 'eeg_ f4', 'eeg_c4', 'eeg_poz',
       'eeg_cz', 'eeg_o2', 'ecg', 'r', 'gsr','pilot']]
```

In [ ]:

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

In [ ]:

```
y_pred
```

Out[ ]:

```
array([[0.95652104, 0.00418882, 0.01537454, 0.0239156 ],
       [0.96050908, 0.02607144, 0.00779653, 0.00562294],
       [0.93161329, 0.00342408, 0.01731141, 0.04765123],
       ...,
       [0.42482424, 0.02867457, 0.50875497, 0.03774623],
       [0.93230396, 0.0043076 , 0.00275002, 0.06063842],
       [0.42629844, 0.02877407, 0.51052042, 0.03440707]])
```

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("Model_building4.csv", index=False)
```

```
                id         A         B         C         D
0                0  0.956521  0.004189  0.015375  0.023916
1                1  0.960509  0.026071  0.007797  0.005623
2                2  0.931613  0.003424  0.017311  0.047651
3                3  0.960632  0.026075  0.007798  0.005496
4                4  0.955304  0.003959  0.018107  0.022630
...            ...       ...       ...       ...       ...
17965138  17965138  0.461766  0.026809  0.475994  0.035431
17965139  17965139  0.930125  0.005662  0.003405  0.060808
17965140  17965140  0.424824  0.028675  0.508755  0.037746
17965141  17965141  0.932304  0.004308  0.002750  0.060638
17965142  17965142  0.426298  0.028774  0.510520  0.034407

[17965143 rows x 5 columns]
```

Conclusion-

I have submitted this file and I got the result on kaggle which is public score of 0.597 and private score of 0.784

Model_building4.zip 6 days ago by AtharvaMusale add submission details	0.78498	0.59792	☐

In [ ]: