In [1]:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [2]:

```
import os
os.environ['KAGGLE_CONFIG_DIR'] = "/content"
%cd /content
```

```
/content
```

In [3]:

```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-packages (1.5.8)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/dist-packages (from
kaggle) (2.8.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/dist-packages (from
kaggle) (4.0.1)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-packages (from kaggle)
(1.15.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from kaggle)
(2020.6.20)
Requirement already satisfied: slugify in /usr/local/lib/python3.6/dist-packages (from kaggle)
(0.0.1)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages
(from kaggle) (1.24.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from kaggle)
(4.41.1)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from kaggle)
(2.23.0)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.6/dist-packages (from
python-slugify->kaggle) (1.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from
requests->kaggle) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from
requests->kaggle) (2.10)
```

In [2]:

```
from google.colab import files
files.upload()
```

Choose File No file selected

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
```

Out[2]:

```
{'kaggle.json': b'{"username":"sankalpchawla","key":"1d2fc96a3f385db0c3807663a6ffb2c5"}'}
```

In [3]:

```
!pip install -q kaggle
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!ls ~/.kaggle
!chmod 600 /root/.kaggle/kaggle.json
```

```
kaggle.json
```

kaggle.json

In [4]:

```
!unzip -q "/content/gdrive/My Drive/train.csv.zip" -d "/content/"
```

In [5]:

```
!unzip -q "/content/gdrive/My Drive/test.csv.zip" -d "/content/"
```

In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
from scipy import stats
import imblearn
import warnings
import math
import pickle
from sklearn.externals import joblib
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import log_loss
from prettytable import PrettyTable
import lightgbm as lgb
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
warnings.filterwarnings('ignore')
from sklearn.preprocessing import MinMaxScaler
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
/usr/local/lib/python3.6/dist-packages/sklearn/externals/six.py:31: FutureWarning: The module is d
eprecated in version 0.21 and will be removed in version 0.23 since we've dropped support for
Python 2.7. Please rely on the official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/utils/deprecation.py:144: FutureWarning: The
sklearn.neighbors.base module is  deprecated in version 0.22 and will be removed in version 0.24.
The corresponding classes / functions should instead be imported from sklearn.neighbors. Anything
that cannot be imported from sklearn.neighbors is now part of the private API.
  warnings.warn(message, FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/externals/joblib/__init__.py:15: FutureWarning: skl
earn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this
functionality directly from joblib, which can be installed with: pip install joblib. If this
warning is raised when loading pickled models, you may need to re-serialize those models with scik
it-learn 0.21+.
  warnings.warn(msg, category=FutureWarning)
```

In [50]:

```
df_train = pd.read_csv('/content/train.csv')
df_test = pd.read_csv('/content/test.csv')
```

Approach 1

In [86]:

```
df_train['pilot'] = 100 * df_train['seat'] + df_train['crew']
df_test['pilot'] = 100 * df_test['seat'] + df_test['crew']
```

In [87]:

```
# df_train['experiment'] = df_train['experiment'].map({'CA': 0, 'DA': 1, 'SS': 2, 'LOFT': 3})
# df_train["experiment"] = df_train["experiment"].astype('int8')
# df_train['event'] = df_train['event'].map({'A': 3, 'B': 2, 'C': 0, 'D': 1})
# df_train["event"] = df_train["event"].astype('int8')
df_train['experiment'] = df_train['experiment'].map({'CA': 2, 'DA': 3, 'SS': 1, 'LOFT': 0})
df_train["experiment"] = df_train["experiment"].astype('int8')
df_train['event'] = df_train['event'].map({'A': 0, 'B': 1, 'C': 2, 'D': 3})
df_train["event"] = df_train["event"].astype('int8')
y = df_train['event']

df_test['experiment'] = df_test['experiment'].map({'CA': 2, 'DA': 3, 'SS': 1, 'LOFT': 0})
df_test["experiment"] = df_test["experiment"].astype('int8')
```

In [88]:

```
df_test_id = df_test['id']
df_test = df_test.drop('id',axis=1)
df_train = df_train.drop('event',axis=1)
```

In [89]:

```
# train, train_test, y, y_test = train_test_split(train, y, test_size=0.25, shuffle=False)
X_train, X_test, y_train, y_test = train_test_split(df_train, y,test_size=0.25, shuffle=False)
```

In [15]:

```
train, test = train_test_split(df_train, test_size=0.2, shuffle=False,random_state=42)
```

In [72]:

```
# X_train = train.loc[:, df_train.columns != 'event']
# y_train = train.event

# X_test = test.loc[:, df_train.columns != 'event']
# y_test = test.event

print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(3650565, 27) (3650565,)
(1216856, 27) (1216856,)
```

In [94]:

```
lgbtrain = lgb.Dataset(X_train, y_train,categorical_feature=[1])
lgbtest = lgb.Dataset(X_test, y_test,categorical_feature=[1])
# lgbtrain = lgb.Dataset(X_train, y_train)
# lgbtest = lgb.Dataset(X_test, y_test)
```

In [95]:

```
params = {'bagging_fraction': 0.9,
 'bagging_seed': 0,
 'boosting_type': 'gbdt',
 'learning_rate': 0.01,
 'metric': 'multi_error',
 'num_class': 4,
 'num_leaves': 10,
 'num_threads' : 4,
 'colsample_bytree' : 0.5,
 'objective': 'multiclass',
 'min_data_in_leaf':100,
 'min_split_gain':0.0002
}

model_lgb = lgb.train(params, lgbtrain, 2000, valid_sets=[lgbtrain,lgbtest], early_stopping_rounds=
200, verbose_eval=100)
```

```
Training until validation scores don't improve for 200 rounds.
[100] training's multi_error: 0.0736081 valid_1's multi_error: 0.0838012
[200] training's multi_error: 0.0669611 valid_1's multi_error: 0.0837774
[300] training's multi_error: 0.0525864 valid_1's multi_error: 0.0830123
[400] training's multi_error: 0.0501473 valid_1's multi_error: 0.0822768
[500] training's multi_error: 0.0478479 valid_1's multi_error: 0.0816736
[600] training's multi_error: 0.0459033 valid_1's multi_error: 0.0789839
[700] training's multi_error: 0.043579 valid_1's multi_error: 0.0756696
[800] training's multi_error: 0.0416711 valid_1's multi_error: 0.0718294
[900] training's multi_error: 0.0403869 valid_1's multi_error: 0.0700822
[1000] training's multi_error: 0.039126 valid_1's multi_error: 0.0688701
[1100] training's multi_error: 0.0381267 valid_1's multi_error: 0.0675593
[1200] training's multi_error: 0.0371981 valid_1's multi_error: 0.066929
[1300] training's multi_error: 0.0361015 valid_1's multi_error: 0.0659684
[1400] training's multi_error: 0.0349584 valid_1's multi_error: 0.0624585
[1500] training's multi_error: 0.0337745 valid_1's multi_error: 0.0614806
[1600] training's multi_error: 0.0324662 valid_1's multi_error: 0.0613088
[1700] training's multi_error: 0.0312187 valid_1's multi_error: 0.0613565
Early stopping, best iteration is:
[1545] training's multi_error: 0.0331839 valid_1's multi_error: 0.0606966
```

In [96]:

```
model_lgb.best_iteration
```

Out[96]:

```
1545
```

In [97]:

```
predicted_lgb = model_lgb.predict(X_test, num_iteration= model_lgb.best_iteration)
print('Log loss',round(log_loss(y_test.to_numpy(),predicted_lgb),8))
```

```
Log loss 0.13611073
```

In [98]:

```
joblib.dump(model_lgb, 'lightgbm.pkl')
```

Out[98]:

```
['lightgbm.pkl']
```

In [99]:

```
predicted_lgb = model_lgb.predict(df_test, num_iteration= model_lgb.best_iteration)
```

In [100]:

```
submission = pd.DataFrame(np.concatenate((np.arange(len(df_test))[:, np.newaxis], predicted_lgb),
axis=1), columns=['id', 'A', 'B', 'C', 'D'])
submission['id'] = df_test_id.astype(int)
submission.to_csv("Submission.csv", index=False)
```

In [101]:

```
!kaggle competitions submit -c reducing-commercial-aviation-fatalities -f Submission.csv -m "lgb w
ithout FE and some parameter tweaking"
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (server 1.5.6 /
client 1.5.4)
100% 1.58G/1.58G [00:20<00:00, 83.7MB/s]
Successfully submitted to Reducing Commercial Aviation Fatalities
```

Scores:

- Private = 1.25462

- Public = 0.58422

Approach 2

In [8]:

```
df_train['experiment'] = df_train['experiment'].map({'CA': 2, 'DA': 3, 'SS': 1, 'LOFT': 0})
df_train["experiment"] = df_train["experiment"].astype('int8')
df_train['event'] = df_train['event'].map({'A': 0, 'B': 1, 'C': 2, 'D': 3})
df_train["event"] = df_train["event"].astype('int8')
y = df_train['event']

df_test['experiment'] = df_test['experiment'].map({'CA': 2, 'DA': 3, 'SS': 1, 'LOFT': 0})
df_test["experiment"] = df_test["experiment"].astype('int8')
```

In [9]:

```
df_test_id = df_test['id']
df_test = df_test.drop('id',axis=1)
df_train = df_train.drop('event',axis=1)
```

In [10]:

```
df_train['fp1_f7'] = df_train['eeg_fp1'] - df_train['eeg_f7']
df_train['f7_t3'] = df_train['eeg_f7'] - df_train['eeg_t3']
df_train['t3_t5'] = df_train['eeg_t3'] - df_train['eeg_t5']
df_train['t5_o1'] = df_train['eeg_t5'] - df_train['eeg_o1']
df_train['fp1_f3'] = df_train['eeg_fp1'] - df_train['eeg_f7']
df_train['f3_c3'] = df_train['eeg_f3'] - df_train['eeg_c3']
df_train['c3_p3'] = df_train['eeg_c3'] - df_train['eeg_p3']
df_train['p3_o1'] = df_train['eeg_p3'] - df_train['eeg_o1']
df_train['fz_cz'] = df_train['eeg_fz'] - df_train['eeg_cz']
df_train['cz_pz'] = df_train['eeg_cz'] - df_train['eeg_pz']
df_train['pz_poz'] = df_train['eeg_pz'] - df_train['eeg_poz']
df_train['fp2_f8'] = df_train['eeg_fp2'] - df_train['eeg_f8']
df_train['f8_t4'] = df_train['eeg_f8'] - df_train['eeg_t4']
df_train['t4_t6'] = df_train['eeg_t4'] - df_train['eeg_t6']
df_train['t6_o2'] = df_train['eeg_t6'] - df_train['eeg_o2']
df_train['fp2_f4'] = df_train['eeg_fp2'] - df_train['eeg_f4']
df_train['f4_c4'] = df_train['eeg_f4'] - df_train['eeg_c4']
df_train['c4_p4'] = df_train['eeg_c4'] - df_train['eeg_p4']
df_train['p4_o2'] = df_train['eeg_p4'] - df_train['eeg_o2']
```

In [11]:

```
df_test['fp1_f7'] = df_test['eeg_fp1'] - df_test['eeg_f7']
df_test['f7_t3'] = df_test['eeg_f7'] - df_test['eeg_t3']
df_test['t3_t5'] = df_test['eeg_t3'] - df_test['eeg_t5']
df_test['t5_o1'] = df_test['eeg_t5'] - df_test['eeg_o1']
df_test['fp1_f3'] = df_test['eeg_fp1'] - df_test['eeg_f7']
df_test['f3_c3'] = df_test['eeg_f3'] - df_test['eeg_c3']
df_test['c3_p3'] = df_test['eeg_c3'] - df_test['eeg_p3']
df_test['p3_o1'] = df_test['eeg_p3'] - df_test['eeg_o1']
df_test['fz_cz'] = df_test['eeg_fz'] - df_test['eeg_cz']
df_test['cz_pz'] = df_test['eeg_cz'] - df_test['eeg_pz']
df_test['pz_poz'] = df_test['eeg_pz'] - df_test['eeg_poz']
df_test['fp2_f8'] = df_test['eeg_fp2'] - df_test['eeg_f8']
df_test['f8_t4'] = df_test['eeg_f8'] - df_test['eeg_t4']
df_test['t4_t6'] = df_test['eeg_t4'] - df_test['eeg_t6']
df_test['t6_o2'] = df_test['eeg_t6'] - df_test['eeg_o2']
df_test['fp2_f4'] = df_test['eeg_fp2'] - df_test['eeg_f4']
df_test['f4_c4'] = df_test['eeg_f4'] - df_test['eeg_c4']
df_test['c4_p4'] = df_test['eeg_c4'] - df_test['eeg_p4']
df_test['p4_o2'] = df_test['eeg_p4'] - df_test['eeg_o2']
```

In [10]:

```
X_train, X_test, y_train, y_test = train_test_split(df_train, y,test_size=0.25, shuffle=False)
```

In [41]:

```
lgbtrain = lgb.Dataset(X_train, y_train,categorical_feature=[1])
lgbtest = lgb.Dataset(X_test, y_test,categorical_feature=[1])
```

In [12]:

```
params = {'bagging_fraction': 0.9,
 'bagging_seed': 0,
 'boosting_type': 'gbdt',
 'learning_rate': 0.01,
 'metric': 'multi_error',
 'num_class': 4,
 'num_leaves': 10,
 'num_threads' : 4,
 'colsample_bytree' : 0.5,
 'objective': 'multiclass',
 'min_data_in_leaf':100,
 'min_split_gain':0.0002
}

model_lgb = lgb.train(params, lgbtrain, 2000, valid_sets=[lgbtrain,lgbtest], early_stopping_rounds=
200, verbose_eval=100)
```

```
Training until validation scores don't improve for 200 rounds.
[100] training's multi_error: 0.0738036 valid_1's multi_error: 0.0838012
[200] training's multi_error: 0.0640621 valid_1's multi_error: 0.0836155
[300] training's multi_error: 0.0520991 valid_1's multi_error: 0.0828792
[400] training's multi_error: 0.0495247 valid_1's multi_error: 0.0821272
[500] training's multi_error: 0.0472713 valid_1's multi_error: 0.0817558
[600] training's multi_error: 0.0453313 valid_1's multi_error: 0.0780824
[700] training's multi_error: 0.0432382 valid_1's multi_error: 0.070812
[800] training's multi_error: 0.0411832 valid_1's multi_error: 0.0626122
[900] training's multi_error: 0.0398845 valid_1's multi_error: 0.0602955
[1000] training's multi_error: 0.0388145 valid_1's multi_error: 0.0586602
[1100] training's multi_error: 0.0377662 valid_1's multi_error: 0.057974
[1200] training's multi_error: 0.0367776 valid_1's multi_error: 0.0575639
[1300] training's multi_error: 0.0355893 valid_1's multi_error: 0.0571834
[1400] training's multi_error: 0.034307 valid_1's multi_error: 0.0566558
[1500] training's multi_error: 0.0332414 valid_1's multi_error: 0.0561011
[1600] training's multi_error: 0.0321539 valid_1's multi_error: 0.0561266
[1700] training's multi_error: 0.0310171 valid_1's multi_error: 0.0562392
Early stopping, best iteration is:
[1574] training's multi_error: 0.0324843 valid_1's multi_error: 0.0557897
```

In [13]:

```
model_lgb.best_iteration
```

Out[13]:

```
1574
```

In [14]:

```
predicted_lgb = model_lgb.predict(X_test, num_iteration= model_lgb.best_iteration)
print('Log loss',round(log_loss(y_test.to_numpy(),predicted_lgb),8))
```

```
Log loss 0.12914405
```

In [15]:

```
predicted_lgb = model_lgb.predict(df_test, num_iteration= model_lgb.best_iteration)
submission = pd.DataFrame(np.concatenate((np.arange(len(df_test))[:, np.newaxis], predicted_lgb),
axis=1), columns=['id', 'A', 'B', 'C', 'D'])
submission['id'] = df_test_id.astype(int)
submission.to_csv("Submission.csv", index=False)
```

In [19]:

```
!kaggle competitions submit -c reducing-commercial-aviation-fatalities -f Submission.csv -m "Appro
ach 2"
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (server 1.5.6 / client 1.5.4)
100% 1.57G/1.57G [00:28<00:00, 60.1MB/s]
Successfully submitted to Reducing Commercial Aviation Fatalities
```

Score:

- Private = 1.25894
- Public = 0.58775

Approach 3

In [164]:

```
df_train['experiment'] = df_train['experiment'].map({'CA': 0, 'DA': 1, 'SS': 3, 'LOFT': 4})
df_train["experiment"] = df_train["experiment"].astype('int8')
df_train['event'] = df_train['event'].map({'A': 0, 'B': 1, 'C': 2, 'D': 3})
df_train["event"] = df_train["event"].astype('int8')
y = df_train['event']

df_test['experiment'] = df_test['experiment'].map({'CA': 0, 'DA': 1, 'SS': 3, 'LOFT': 4})
df_test["experiment"] = df_test["experiment"].astype('int8')
```

In [46]:

```
df_train.columns
```

Out[46]:

```
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 [165]:

```
feat = ['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']
```

In [166]:

```
sc = MinMaxScaler()
for i in feat:
  df_train[i] = sc.fit_transform(np.array(df_train[i]).reshape(-1,1))
df_train.head()
```

Out[166]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_c
0	1	0	0.011719	1	0.406790	0.442990	0.431285	0.462933	0.583200	0.532208	0.466463	0.396589	0.6119
1	1	0	0.015625	1	0.407647	0.443446	0.431339	0.465714	0.582847	0.530859	0.465496	0.396652	0.6113
2	1	0	0.019531	1	0.411577	0.443994	0.437849	0.474482	0.582929	0.531555	0.467078	0.398478	0.6116
3	1	0	0.023438	1	0.411811	0.442668	0.434441	0.470692	0.584998	0.531289	0.464828	0.398430	0.6122
4	1	0	0.027344	1	0.410560	0.442760	0.433829	0.468640	0.584669	0.530168	0.465339	0.397945	0.6124

In [167]:

```
sc = MinMaxScaler()
for i in feat:
  df_test[i] = sc.fit_transform(np.array(df_test[i]).reshape(-1,1))
```

In [168]:

```
X_train, X_test, y_train, y_test = train_test_split(df_train, y,test_size=0.25, shuffle=False)
```

In [182]:

```
lgbtrain = lgb.Dataset(X_train, y_train,categorical_feature=[1])
lgbtest = lgb.Dataset(X_test, y_test,categorical_feature=[1])
```

In [116]:

```
params = {
        'objective' : 'multiclass',
        'metric' : 'multi_error',
        'boosting' :'gbdt',
        'num_class':4,
        'num_leaves' : 30,
        'learning_rate' : 0.04,
        'bagging_fraction' : 0.9,
        'bagging_seed' : 0,
        'num_threads' : 4,
        'colsample_bytree' : 0.5,
        'min_data_in_leaf':100,
        'min_split_gain':0.00015
          }

model_lgb1 = lgb.train(params, lgbtrain, 2000, valid_sets=[lgbtrain,lgbtest], early_stopping_rounds
=200, verbose_eval=100)
```

```
Training until validation scores don't improve for 200 rounds.
[100] training's multi_error: 0 valid_1's multi_error: 5.01292e-05
[200] training's multi_error: 0 valid_1's multi_error: 0
Early stopping, best iteration is:
[78] training's multi_error: 0 valid_1's multi_error: 0.000110942
```

In [119]:

```
predicted_lgb = model_lgb1.predict(X_test, num_iteration= model_lgb1.best_iteration)
print('Log loss',round(log_loss(y_test.to_numpy(),predicted_lgb),8))
```

```
Log loss 0.0833791
```

In [173]:

```
params = {
        'objective' : 'multiclass',
        'metric' : 'multi_error',
        'boosting' :'gbdt',
        'num_class':4,
        'num_leaves' : 30,
        'learning_rate' : 0.05,
        'bagging_fraction' : 0.9,
        'bagging_seed' : 0,
        'num_threads' : 4,
        'colsample_bytree' : 0.5,
        'min_data_in_leaf':100,
        'min_split_gain':0.00015
          }

model_lgb = lgb.train(params, lgbtrain, 2000, valid_sets=[lgbtrain,lgbtest], early_stopping_rounds=
200, verbose_eval=100)

#0.59629
#0.38000
```

```
Training until validation scores don't improve for 200 rounds.
[100] training's multi_error: 0 valid_1's multi_error: 1.9723e-05
[200] training's multi_error: 0 valid_1's multi_error: 0
Early stopping, best iteration is:
[69] training's multi_error: 0 valid_1's multi_error: 8.46444e-05
```

In [174]:

```
predicted_lgb = model_lgb.predict(X_test, num_iteration= model_lgb.best_iteration)
print('Log loss',round(log_loss(y_test.to_numpy(),predicted_lgb),8))
```

```
Log loss 0.06922234
```

In [175]:

```
predicted_lgb = model_lgb.predict(df_test, num_iteration= model_lgb.best_iteration)
submission = pd.DataFrame(np.concatenate((np.arange(len(df_test))[:, np.newaxis], predicted_lgb),
axis=1), columns=['id', 'A', 'B', 'C', 'D'])
submission['id'] = df_test_id.astype(int)
submission.to_csv("Submission.csv", index=False)
```

In [183]:

```
params = {
        'objective' : 'multiclass',
        'metric' : 'multi_error',
        'boosting' :'gbdt',
        'num_class':4,
        'num_leaves' : 30,
        'learning_rate' : 0.06,
        'bagging_fraction' : 0.9,
        'bagging_seed' : 0,
        'num_threads' : 4,
        'colsample_bytree' : 0.4,
        'min_data_in_leaf':100,
        'min_split_gain':0.00015
          }

#0.61659
#0.33278

model_lgb2 = lgb.train(params, lgbtrain, 2000, valid_sets=[lgbtrain,lgbtest], early_stopping_rounds
=200, verbose_eval=100)
```

```
Training until validation scores don't improve for 200 rounds.
[100] training's multi_error: 0 valid_1's multi_error: 0.0002112
[200] training's multi_error: 0 valid_1's multi_error: 4.93074e-06
Early stopping, best iteration is:
[73] training's multi_error: 0 valid_1's multi_error: 0.00107737
```

In [184]:

```
predicted_lgb = model_lgb2.predict(X_test, num_iteration= model_lgb2.best_iteration)
print('Log loss',round(log_loss(y_test.to_numpy(),predicted_lgb),8))
```

```
Log loss 0.06403297
```

In [185]:

```
predicted_lgb = model_lgb2.predict(df_test, num_iteration= model_lgb2.best_iteration)
submission = pd.DataFrame(np.concatenate((np.arange(len(df_test))[:, np.newaxis], predicted_lgb),
axis=1), columns=['id', 'A', 'B', 'C', 'D'])
submission['id'] = df_test_id.astype(int)
submission.to_csv("Submission.csv", index=False)
```

In [186]:

```
!kaggle competitions submit -c reducing-commercial-aviation-fatalities -f Submission.csv -m "Appro
ach 3 parameter tuning test"
```

```
Warning: Looks like you're using an outdated API Version, please consider updating (server 1.5.6 /
client 1.5.4)
100% 1.49G/1.49G [00:14<00:00, 107MB/s]
Successfully submitted to Reducing Commercial Aviation Fatalities
```

Scores:

- Private = 0.60805
- Public = 0.34161

Approach 4

In [155]:

```
df_train['experiment'] = df_train['experiment'].map({'CA': 2, 'DA': 3, 'SS': 1, 'LOFT': 0})
df_train["experiment"] = df_train["experiment"].astype('int8')
df_train['event'] = df_train['event'].map({'A': 0, 'B': 1, 'C': 2, 'D': 3})
df_train["event"] = df_train["event"].astype('int8')
y = df_train['event']

df_test['experiment'] = df_test['experiment'].map({'CA': 2, 'DA': 3, 'SS': 1, 'LOFT': 0})
df_test["experiment"] = df_test["experiment"].astype('int8')
```

In [156]:

```
df_test_id = df_test['id']
df_test = df_test.drop('id',axis=1)
df_train = df_train.drop('event',axis=1)
```

In [157]:

```
feat = ['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']
```

In [158]:

```
sc = MinMaxScaler()
for i in feat:
  df_train[i] = sc.fit_transform(np.array(df_train[i]).reshape(-1,1))
df_train.head()
```

Out[158]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_c
0	1	2	0.011719	1	0.406790	0.442990	0.431285	0.462933	0.583200	0.532208	0.466463	0.396589	0.6119
1	1	2	0.015625	1	0.407647	0.443446	0.431339	0.465714	0.582847	0.530859	0.465496	0.396652	0.6113
2	1	2	0.019531	1	0.411577	0.443994	0.437849	0.474482	0.582929	0.531555	0.467078	0.398478	0.6116
3	1	2	0.023438	1	0.411811	0.442668	0.434441	0.470692	0.584998	0.531289	0.464828	0.398430	0.6122
4	1	2	0.027344	1	0.410560	0.442760	0.433829	0.468640	0.584669	0.530168	0.465339	0.397945	0.6124

In [159]:

```
sc = MinMaxScaler()
for i in feat:
  df_test[i] = sc.fit_transform(np.array(df_test[i]).reshape(-1,1))
```

In [160]:

```
X_train, X_test, y_train, y_test = train_test_split(df_train, y,test_size=0.3, shuffle=False)
```

In [161]:

```
lgbtrain = lgb.Dataset(X_train, y_train,categorical_feature=[1])
lgbtest = lgb.Dataset(X_test, y_test,categorical_feature=[1])
```

In [ ]:

```
params = {
        "objective" : "multiclass",
        "metric" : "multi_error",
        "boosting" :'gbdt',
        'num_class':4,
        "num_leaves" : 30,
        "learning_rate" : 0.01,
        "bagging_fraction" : 0.9,
        "bagging_seed" : 0,
        "num_threads" : 4,
        "colsample_bytree" : 0.4,
        'min_data_in_leaf':100,
        'min_split_gain':0.00015
          }

model_lgb = lgb.train(params, lgbtrain, 2000, valid_sets=[lgbtrain,lgbtest], early_stopping_rounds=
200, verbose_eval=100)
```

In [137]:

```
predicted_lgb = model_lgb.predict(X_test, num_iteration= model_lgb.best_iteration)
print('Log loss',round(log_loss(y_test.to_numpy(),predicted_lgb),8))
```

```
Log loss 0.3020943
```

In [ ]:

```
predicted_lgb = model_lgb.predict(df_test, num_iteration= model_lgb.best_iteration)
submission = pd.DataFrame(np.concatenate((np.arange(len(df_test))[:, np.newaxis], predicted_lgb),
axis=1), columns=['id', 'A', 'B', 'C', 'D'])
submission['id'] = df_test_id.astype(int)
submission.to_csv("Submission.csv", index=False)
```

In [ ]:

```
!kaggle competitions submit -c reducing-commercial-aviation-fatalities -f Submission.csv -m "Appro
ach 4 parameter tuning"
```

Approach 5

In [51]:

```
df_train['experiment'] = df_train['experiment'].map({'CA': 0, 'DA': 1, 'SS': 3, 'LOFT': 4})
df_train["experiment"] = df_train["experiment"].astype('int8')
df_train['event'] = df_train['event'].map({'A': 0, 'B': 1, 'C': 2, 'D': 3})
df_train["event"] = df_train["event"].astype('int8')
y = df_train['event']

df_test['experiment'] = df_test['experiment'].map({'CA': 0, 'DA': 1, 'SS': 3, 'LOFT': 4})
df_test["experiment"] = df_test["experiment"].astype('int8')
```

In [52]:

```
feat = ['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']
```

In [53]:

```
sc = MinMaxScaler()
for i in feat:
  df_train[i] = sc.fit_transform(np.array(df_train[i]).reshape(-1,1))
df_train.head()
```

Out[53]:

	crew	experiment	time	seat	eeg_fp1	eeg_f7	eeg_f8	eeg_t4	eeg_t6	eeg_t5	eeg_t3	eeg_fp2	eeg_o
0	1	0	0.011719	1	0.406790	0.442990	0.431285	0.462933	0.583200	0.532208	0.466463	0.396589	0.6119
1	1	0	0.015625	1	0.407647	0.443446	0.431339	0.465714	0.582847	0.530859	0.465496	0.396652	0.6113
2	1	0	0.019531	1	0.411577	0.443994	0.437849	0.474482	0.582929	0.531555	0.467078	0.398478	0.6116
3	1	0	0.023438	1	0.411811	0.442668	0.434441	0.470692	0.584998	0.531289	0.464828	0.398430	0.6122
4	1	0	0.027344	1	0.410560	0.442760	0.433829	0.468640	0.584669	0.530168	0.465339	0.397945	0.6124

In [54]:

```
sc = MinMaxScaler()
for i in feat:
  df_test[i] = sc.fit_transform(np.array(df_test[i]).reshape(-1,1))
```

In [55]:

```
tr_col = df_train.columns
```

In [56]:

```
df_train,y = SMOTE().fit_resample(df_train,y.ravel())
df_train = pd.DataFrame(df_train)
```

In [57]:

```
df_train.columns = tr_col
```

In [58]:

```
X_train, X_test, y_train, y_test = train_test_split(df_train, y,test_size=0.25, shuffle=False)
```

In [59]:

```
lgbtrain = lgb.Dataset(X_train, y_train,categorical_feature=[1])
lgbtest = lgb.Dataset(X_test, y_test,categorical_feature=[1])
```

In [60]:

```
params = {
        "objective" : "multiclass",
        "metric" : "multi_error",
        "boosting" :'gbdt',
        'num_class':4,
        "num_leaves" : 30,
        "learning_rate" : 0.06,
        "bagging_fraction" : 0.9,
        "bagging_seed" : 0,
        "num_threads" : 4,
        "colsample_bytree" : 0.4,
        'min_data_in_leaf':100,
        'min_split_gain':0.00015
          }


model_lgb2 = lgb.train(params, lgbtrain, 2000, valid_sets=[lgbtrain,lgbtest], early_stopping_rounds
=200, verbose_eval=100)
```

```
Training until validation scores don't improve for 200 rounds.
[100] training's multi_error: 0 valid_1's multi_error: 0
[200] training's multi_error: 0 valid_1's multi_error: 0
Early stopping, best iteration is:
[85] training's multi_error: 0 valid_1's multi_error: 0
```

In [ ]:

```
predicted_lgb = model_lgb2.predict(df_test, num_iteration= model_lgb2.best_iteration)
submission = pd.DataFrame(np.concatenate((np.arange(len(df_test))[:, np.newaxis], predicted_lgb),
axis=1), columns=['id', 'A', 'B', 'C', 'D'])
submission['id'] = df_test_id.astype(int)
submission.to_csv("Submission.csv", index=False)
```

In [ ]:

```
!kaggle competitions submit -c reducing-commercial-aviation-fatalities -f Submission.csv -m "Appro
ach 5 parameter tuning test with smote"
```