In [158]:

```
import warnings
warnings.filterwarnings("ignore")
import pickle
from xgboost.sklearn import XGBClassifier
import numpy as np
import pandas as pd
from scipy.interpolate import interp1d
from biosppy.signals import ecg
from biosppy.signals import resp
from biosppy.signals import eda
from biosppy.signals import eeg
from timeit import default_timer as timer
from sklearn.metrics import log_loss
```

In [159]:

```
def interpolated_value(t, feature_ts, feature_data):
    ''' to predict values that fall within two existing data points using interpolation '''
    new = interp1d(feature_ts, feature_data, kind='cubic', fill_value="extrapolate")
    return new(t)
```

In [160]:

```
def pilot_features(data, loca, eeg_features):
    ''' to add new features using biosppy.signals functions'''

    temp_df = data.loc[loca][['time', 'ecg', 'r', 'gsr'] + eeg_features].values
    temp_df = temp_df[temp_df[:,0].argsort()]

    #if any value is zero for sensors, then replace it with nan. Else create new features like 'he
art_rate', 'resp_rate', 'gsr_amp'

    if np.allclose(temp_df[:,1], 0, rtol=1e-10):
        data.loc[loca, 'ecg'] = np.nan
        print('missing egc, value should not be zero')
    else:
        try:
            heart_sig = ecg.ecg(signal=temp_df[:,1], sampling_rate=256., show=False)
            heart_rate = heart_sig['heart_rate']
            heart_rate_ts = heart_sig['heart_rate_ts']
            data.loc[loca, 'heart_rate'] = interpolated_value(temp_df[:,0], heart_rate_ts, heart_ra
te)
        except ValueError:
            pass


    if np.allclose(temp_df[:,2], 0, rtol=1e-10):
        data.loc[loca, 'r'] = np.nan
        print('missing r, value should not be zero')
    else:
        try:
            resp_sig = resp.resp(signal=temp_df[:,2], sampling_rate=256., show=False)
            resp_rate = resp_sig['resp_rate']
            resp_rate_ts = resp_sig['resp_rate_ts']
            data.loc[loca, 'resp_rate'] = interpolated_value(temp_df[:,0], resp_rate_ts, resp_rate)
        except ValueError:
            pass


    if np.allclose(temp_df[:,3], 0, rtol=1e-10):
        data.loc[loca, 'gsr'] = np.nan
        print('missing gsr, value should not be zero')
    else:
        try:
            gsr_sig = eda.eda(signal=temp_df[:,3], sampling_rate=256., show=False)
            gsr_amp = gsr_sig['amplitudes']
            gsr_amp_ts = temp_df[gsr_sig['onsets'], 0]
            data.loc[loca, 'gsr_amp'] = interpolated_value(temp_df[:,0], gsr_amp_ts, gsr_amp)
        except IndexError:
            pass
```

```
        except ValueError:
            pass


    # creating 5 more features with 'get_power_features' function which returns 6 values, using in
terpolation on top of that.
    try:
        eeg_feat_sig = eeg.get_power_features(signal=temp_df[:,4:], sampling_rate=256.)
        eeg_ts = eeg_feat_sig['ts']
        eeg_theta = eeg_feat_sig['theta']
        eeg_alpha_low = eeg_feat_sig['alpha_low']
        eeg_alpha_high = eeg_feat_sig['alpha_high']
        eeg_beta = eeg_feat_sig['beta']
        eeg_gamma = eeg_feat_sig['gamma']
        for i, e in enumerate(eeg_features):
            data.loc[loca, e + '_theta'] = interpolated_value(temp_df[:,0], eeg_ts, eeg_theta[:,i])
            data.loc[loca, e + '_alpha_low'] = interpolated_value(temp_df[:,0], eeg_ts,
eeg_alpha_low[:,i])
            data.loc[loca, e + '_alpha_high'] = interpolated_value(temp_df[:,0], eeg_ts,
eeg_alpha_high[:,i])
            data.loc[loca, e + '_beta'] = interpolated_value(temp_df[:,0], eeg_ts, eeg_beta[:,i])
            data.loc[loca, e + '_gamma'] = interpolated_value(temp_df[:,0], eeg_ts, eeg_gamma[:,i])
    except ValueError:
        pass
```

In [164]:

```
def func1(raw_data):
    ''' taking 1 datapoint as input with 27 features and returning the predicted output for it '''

    start = timer()
    train=pd.read_csv('train.csv').sample(4000)
    train=train.drop('event',axis=1)

    if raw_data[1] == 'LOFT':
        raw_data[1]=4
    elif raw_data[1] == 'CA':
        raw_data[1]=0
    elif raw_data[1] == 'DA':
        raw_data[1]=1
    elif raw_data[1] == 'SS':
        raw_data[1]=3

    raw_data=np.array(raw_data,dtype=float)
    raw_data=raw_data.reshape(1,27)
    raw_data=pd.DataFrame(raw_data,columns=train.columns.tolist())
    raw_data=raw_data.append(train)
    raw_data.reset_index(inplace = True)
    raw_data=raw_data.drop('index',axis=1)
    raw_data['heart_rate'] = np.nan
    raw_data['resp_rate'] = np.nan
    raw_data['gsr_amp'] = np.nan
    eeg_features = ["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"]
    for e in eeg_features:
        raw_data[e + '_theta'] = np.nan
        raw_data[e + '_alpha_low'] = np.nan
        raw_data[e + '_alpha_high'] = np.nan
        raw_data[e + '_beta'] = np.nan
        raw_data[e + '_gamma'] = np.nan
    pilot_features(raw_data, raw_data.index.values,eeg_features)

    pilot=[]
    for i in range(len(raw_data)):
        pilot.append(raw_data['crew'][i]*10+raw_data['seat'][i])
    raw_data['pilot']=pilot

    raw_data=raw_data.drop(['crew','experiment','time','seat'],axis=1)
    features=pd.read_csv('PermImp_df.csv')
    featr=[]
    for i in range(len(features)):
        if (features['weight'][i]<=0):
            featr.append(features.iloc[i]['feature'])
```

```
    raw_data=raw_data.drop(featr,axis=1)

    best_model = pickle.load(open("xgb_model.pickle.dat", "rb"))
    pred=best_model.predict_proba(raw_data)
    end = timer()
    print('total time : ',end - start)
    return pred[0]
```

In [169]:

```
def func2(pred,y):
    ''' returning the log loss for true and predicted values '''

    return log_loss(y,pred)
```

In [165]:

```
# sample raw data

data=[1,'DA',79.3125,0,-12.3193,-9.38664,-8.27289,4.182519999999999,-5.07408,-12.8671,-1.7250900000
000002,-11.9463,-9.22448,
 -2.7210099999999997,3.426,-9.89132,-0.274316,-6.72473,-2.2144,-0.5635399999999999,-1.51768,-5.3214
3,5.04036,-6.22804,
 -4454.430176,735.140991,1076.25]

len(data)
```

Out[165]:

```
27
```

In [174]:

```
# function 1 calling

pred=func1(data)
print('predicted values are:',pred)

# function 2 calling

y_true=[1,0,0,0]    # specify true values of y for each class
loss=func2(pred,y_true)
print('log_loss for our data point is:',loss)
```

```
total time :  18.589751288000116
predicted values are: [9.9833626e-01 1.1539460e-03 1.0484047e-04 4.0494994e-04]
log_loss for our data point is: 0.0008324070859089261
```

In [ ]:

In [ ]: