In [ ]:

```
!pip install biosppy
```

```
Collecting biosppy
  Downloading biosppy-0.7.3.tar.gz (85 kB)
     |████████████████████████████████| 85 kB 2.5 MB/s
Collecting bidict
  Downloading bidict-0.21.4-py3-none-any.whl (36 kB)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from biosp
py) (3.1.0)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from
biosppy) (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from bios
ppy) (1.19.5)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (fr
om biosppy) (1.0.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from bios
ppy) (1.4.1)
Collecting shortuuid
  Downloading shortuuid-1.0.8-py3-none-any.whl (9.5 kB)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from biospp
y) (1.15.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from bio
sppy) (1.1.0)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.7/dist-packages (f
rom biosppy) (4.1.2.30)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages
(from h5py->biosppy) (1.5.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-pack
ages (from matplotlib->biosppy) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (fr
om matplotlib->biosppy) (0.11.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib
/python3.7/dist-packages (from matplotlib->biosppy) (3.0.6)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-package
s (from matplotlib->biosppy) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pack
ages (from scikit-learn->biosppy) (3.0.0)
Building wheels for collected packages: biosppy
  Building wheel for biosppy (setup.py) ... done
  Created wheel for biosppy: filename=biosppy-0.7.3-py2.py3-none-any.whl size=95430 sha25
6=c03eeeede06183dccf5b3a1ee17ace4381c958441e47807b9b43d241017bd9bf
  Stored in directory: /root/.cache/pip/wheels/2f/4f/8f/28b2adc462d7e37245507324f4817ce1c
64ef2464f099f4f0b
Successfully built biosppy
Installing collected packages: shortuuid, bidict, biosppy
Successfully installed bidict-0.21.4 biosppy-0.7.3 shortuuid-1.0.8
```

In [ ]:

```
import pandas as pd
import numpy as np

from scipy import signal

from biosppy.signals import ecg
from biosppy.signals import eeg
from biosppy.signals import resp
from scipy.interpolate import interp1d
import pickle
import joblib
from sklearn.metrics import log_loss,f1_score
from timeit import default_timer as timer
import flask
from flask import Flask, jsonify, request
```

In [ ]:

```
# train_df = train_df.sample(5000)
# pickle.dump(train_df,open('train_df.pkl','wb'))
train_df = pickle.load(open('/content/drive/MyDrive/train_df.pkl','rb'))
```

In [ ]:

```
def interpolation_fn(timestamps,biosppy_ts, biosppy_values):
    """linear interpolation function to produce heart rate, resp rate all time steps"""
    interpolation = interp1d(biosppy_ts,biosppy_values, kind="linear", fill_value="extrap
olate")
    return interpolation(timestamps)
```

In [ ]:

```
def noise_free(data,w):

   ''' function takes raw  signal and removes some noise present init gives noise free sig
nal '''
  n=5
  b,a = signal.butter(n,w,fs=256)

  return signal.filtfilt(b,a,data)
```

In [ ]:

```
def biosppy(df):
    """THIS FUNCTION WILL DERIVE ALL FEATURE THAT IS GENEARTED USING BIOSPPY MODULE"""

    df['filt_ecg'] = noise_free(df.ecg,100)                       # filtering ecg signal
    df['filt_respiration'] = noise_free(df.r,0.7)                 # filtering r signal

    bio=ecg.ecg(df["ecg"],sampling_rate=256,show=False)
#heart rate from ecg
    df["heart_rate"]=interpolation_fn(df["time"],bio["heart_rate_ts"],bio["heart_rate"])


    bio=resp.resp(df["r"],sampling_rate=256,show=False)
#resp rate from r signal
    df["resp_rate"]=interpolation_fn(df["time"],bio["resp_rate_ts"],bio["resp_rate"])


    return df
```

In [ ]:

```
def potential_differences(df):
   """FUNCTION TO CALCULATE POTENTIAL DIFFERENCE BETWEEN ELECTRODES"""


  df['fp1_f7'] = df['eeg_fp1'] - df['eeg_f7']
  df['f7_t3'] = df['eeg_f7'] - df['eeg_t3']
  df['t3_t5'] = df['eeg_t3'] - df['eeg_t5']
  df['t5_o1'] = df['eeg_t5'] - df['eeg_o1']
  df['fp1_f3'] = df['eeg_fp1'] - df['eeg_f7']
  df['f3_c3'] = df['eeg_f3'] - df['eeg_c3']
  df['c3_p3'] = df['eeg_c3'] - df['eeg_p3']
  df['p3_o1'] = df['eeg_p3'] - df['eeg_o1']

  df['fz_cz'] = df['eeg_fz'] - df['eeg_cz']
  df['cz_pz'] = df['eeg_cz'] - df['eeg_pz']                          # train potential differ
ences
  df['pz_poz'] = df['eeg_pz'] - df['eeg_poz']

  df['fp2_f8'] = df['eeg_fp2'] - df['eeg_f8']
  df['f8_t4'] = df['eeg_f8'] - df['eeg_t4']
  df['t4_t6'] = df['eeg_t4'] - df['eeg_t6']
  df['t6_o2'] = df['eeg_t6'] - df['eeg_o2']
  df['fp2_f4'] = df['eeg_fp2'] - df['eeg_f4']
  df['f4_c4'] = df['eeg_f4'] - df['eeg_c4']
  df['c4_p4'] = df['eeg_c4'] - df['eeg_p4']
  df['p4_o2'] = df['eeg_p4'] - df['eeg_o2']
  return df
```

In [ ]:

```
features_n = ['fp1_f7', 'f7_t3', 't3_t5', 't5_o1', 'fp1_f3', 'f3_c3', 'c3_p3', 'p3_o1',
'fz_cz', 'cz_pz',
              'pz_poz', 'fp2_f8', 'f8_t4', 't4_t6', 't6_o2', 'fp2_f4', 'f4_c4', 'c4_p4
', 'p4_o2', 'resp_rate','heart_rate', "gsr",'filt_ecg','filt_respiration']
```

In [ ]:

```
# sample raw data

raw_data="""1,'DA',79.3125,0,-12.3193,-9.38664,-8.27289,4.182519999999999,-5.07408,-12.86
71,-1.7250900000000002,-11.9463,-9.22448,
 -2.7210099999999997,3.426,-9.89132,-0.274316,-6.72473,-2.2144,-0.5635399999999999,-1.517
68,-5.32143,5.04036,-6.22804,
 -4454.430176,735.140991,1076.25"""
```

In [ ]:

```
def prediction_func1(raw_data):
  ''' taking 1 datapoint as input with 27 features and returning the predicted output for
it '''

  start = timer()
  train=pickle.load(open('/content/drive/MyDrive/train_df.pkl','rb')) # sample 5000
  train=train.drop('event',axis=1)
  raw_data = list(raw_data.split(','))

  for i in range(len(raw_data)):

    if i==0 or i==3:
      raw_data[i] = int(raw_data[i])
    elif i==1 :
      raw_data[i] = raw_data[i]
    else:
      raw_data[i] = float(raw_data[i])


  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 = raw_data.reset_index()

  raw_data=biosppy(raw_data)
  raw_data=potential_differences(raw_data)
  model=pickle.load(open('/content/drive/MyDrive/model/lightgbm.pkl','rb'))
  prob = model.predict_proba(raw_data[features_n])

  end = timer()
  print('total time : ',end - start)

  return prob[0]


pred = prediction_func1(data)
pred
```

```
total time :  1.057310411000799
total time :  1.0575104111000799
```

Out[ ]:

```
array([0.76695001, 0.00535544, 0.19211803, 0.03557652])
```

In [ ]:

```
y_true=[1,0,0,0]
def metric_func2(pred,y):
    ''' returning the logloss for true and predicted values '''

    return log_loss(y,pred)
```

In [ ]:

```
metric_func2(pred,y_true)
```

Out[ ]:

```
0.13006689340819272
```