import matplotlib.pyplot as plt

import base64

from io import BytesIO

import mne

import matplotlib

matplotlib.use("Agg")

def get\_graph():

buffer=BytesIO()

plt.savefig(buffer,format='png')

buffer.seek(0)

image\_png=buffer.getvalue()

graph=base64.b64encode(image\_png)

graph=graph.decode('utf-8')

buffer.close()

return graph

def read(file):

#data=mne.io.read\_raw\_edf('C:/Users/DIVYA/DESKTOP/MAJOR PROJECT/finalproject/'+file.name,preload=True)

import mne

data=mne.io.read\_raw\_edf(file.name,preload=True)

return data

def get\_plot(data):

#data=mne.io.read\_raw\_edf('C:\\Users\\DIVYA\\Desktop\\MAJOR PROJECT\\FINALDATASET\\chb03\_01.edf',preload=True)

#data.plot(duration=50)

#or

data.plot(n\_channels=23, scalings={"eeg":75e-6},title='Auto-scaled Data from arrays',

show=True,color=dict(eeg='darkblue'), duration=15.0,start=5)

#plt.switch\_backend('AGG')

#plt.title('sales of items')

#plt.plot(x,y)

# plt.xticks(rotation=45)

# plt.xlabel('item')

# plt.ylabel('price')

# plt.tight\_layout()

graph=get\_graph()

return graph

def get\_beta\_plot(data):

low\_freq, high\_freq = 12.0, 30.0

data = data.filter(low\_freq, high\_freq, n\_jobs=4)

data.plot(n\_channels=23, scalings={"eeg":75e-6},title='Auto-scaled Data from arrays',

show=True,color=dict(eeg='darkblue'), duration=15.0,start=10)

graph=get\_graph()

return graph

def get\_alpha\_plot(data):

low\_freq, high\_freq = 8.0, 12.0

data = data.filter(low\_freq, high\_freq, n\_jobs=4)

data.plot(n\_channels=23, scalings={"eeg":75e-6},title='Auto-scaled Data from arrays',

show=True,color=dict(eeg='darkblue'), duration=15.0,start=10)

graph=get\_graph()

return graph

def get\_theta\_plot(data):

low\_freq, high\_freq = 4.0, 8.0

data = data.filter(low\_freq, high\_freq, n\_jobs=4)

data.plot(n\_channels=23,scalings={"eeg":75e-6}, title='Auto-scaled Data from arrays',

show=True,color=dict(eeg='darkblue'), duration=15.0,start=10)

graph=get\_graph()

return graph

def get\_delta\_plot(data):

low\_freq, high\_freq = 0.5, 4.0

data = data.filter(low\_freq, high\_freq, n\_jobs=4)

data.plot(n\_channels=23, scalings={"eeg":75e-6},title='Auto-scaled Data from arrays',

show=True,color=dict(eeg='darkblue'), duration=15.0,start=10)

graph=get\_graph()

return graph

def get\_acc():

#plt.figure(figsize = (8,7))

models = ['Random Forest','XTrees','Xgboost','CNN']

total = [96,96.6,97,95]

# Passing the parameters to the bar function

g=plt.bar(models, total, width= 0.4, align='center',color='cyan', edgecolor = 'red')

plt.xticks(fontsize=16)

plt.title("Bar plot representing the accuracy of all the models",pad=20,fontname="Times New Roman",fontsize=22)

plt.xlabel('ALGORITHM',fontname="Times New Roman")

plt.ylabel('ACCURACY',fontname="Times New Roman")

for p in g:

width = p.get\_width()

height = p.get\_height()

x, y = p.get\_xy()

plt.annotate(f'{height/100:0.00%}', (x + width/2, y + height\*1.02), ha='center')

graph=get\_graph()

return graph