Dataprocessing2

December 7, 2023

[ ]:
```python
import pandas as pd
import numpy as np
import os
from sklearn.preprocessing import MinMaxScaler
from random import shuffle
import random as rn
import pickle
from biosppy.signals.tools import band_power
```

[ ]:
```python

```

[ ]:
```python
rootFolder = "Data"
```

[ ]:
```python
def get_filepaths(mainfolder):
    training_filepaths = {}
    folders = os.listdir(mainfolder)
    for folder in folders:
        fpath = mainfolder + "/" + folder
#         fout = open(fpath + "/out.csv", "a")
#         p = 0
        if os.path.isdir(fpath) and "file" not in folder:
            filenames = os.listdir(fpath)
            for filename in filenames:
                fullpath = fpath + "/" + filename
#                 if "out" not in filename:
                training_filepaths[fullpath] = folder
#                     f = open(fullpath)
#                     for line in f:
#                         if (p >= 13614):
#                             f.close()
#                             fout.close()
#                             continue
#                         fout.write(line)
#                         p += 1
#                     f.close()  # not really needed
#         fout.close()
    return training_filepaths
```

[ ]:
```python
training_filepaths = get_filepaths(rootFolder)
training_filepaths_list = []
for i  in training_filepaths.keys():
    training_filepaths_list.append(i)

training_filepaths_list
```

[ ]:
```
['Data/Speaking/s 1cleaned.csv',
 'Data/Speaking/s 3cleaned.csv',
 'Data/Speaking/s 4cleaned.csv',
 'Data/Speaking/s 2cleaned.csv',
 'Data/Reading/r 5cleaned.csv',
 'Data/Reading/r 3cleaned.csv',
 'Data/Reading/r 4cleaned.csv',
 'Data/Reading/r 6cleaned.csv',
 'Data/Reading/r 1cleaned.csv',
 'Data/Reading/r 2cleaned.csv',
 'Data/Watching/w 1cleaned.csv',
 'Data/Watching/w 5cleaned.csv',
 'Data/Watching/w 2cleaned.csv',
 'Data/Watching/w 4cleaned.csv',
 'Data/Watching/w 3cleaned.csv',
 'Data/Watching/w 6cleaned.csv']
```

[ ]:
```python
def get_labels(mainfolder):
    """ Creates a dictionary of labels for each unique type of motion """
    labels = {}
    label = 0
    for folder in os.listdir(mainfolder):
        fpath = mainfolder + "/" + folder
        if os.path.isdir(fpath) and "MODEL" not in folder:
            labels[folder] = label
            label += 1
    return labels
```

[ ]:
```python
labels = get_labels(rootFolder)
```

[ ]:
```python
labels
```

[ ]:
```
{'Speaking': 0, 'Reading': 1, 'Watching': 2}
```

[ ]:
```python
def get_data(fp, labels, folders):
    #     if(os.path.isfile(fp + "filtered.file")):
    #         with open(fp + "filtered.file", "rb") as f:
    #             dump = pickle.load(f)
    #             return dump[0], dump[1], dump[2]
    file_dir = folders[fp]
```

```
    datasignals = pd.read_csv(filepath_or_buffer=fp, sep=',',
                              dtype='float', names=["EEG1", "EEG2", "Acc_X",␣
 ↪"Acc_Y", "Acc_Z"])
#     datasignals = (datasignals - datasignals.mean()) / datasignals.std(ddof=0)
#     datasignals = datasignals[['EEG1','EEG2']]
#     datasignals = norm_data(datasignals)
    one_hot = np.zeros(3)
    label = labels[file_dir]
    one_hot[label] = 1
#     with open(fp + "filtered.file", "wb") as f:
#         pickle.dump([datasignals, one_hot, label], f, pickle.HIGHEST_PROTOCOL)
    return datasignals, one_hot, label
```

[ ]:
```
datasignals, one_hot, label = get_data (training_filepaths_list[0],labels,␣
 ↪training_filepaths)
```

[ ]:
```
import pandas as pd
import glob

def get_data(fp):
    datasignals = pd.read_csv(filepath_or_buffer=fp, sep=',', dtype='float',
                              names=["EEG1", "EEG2", "Acc_X", "Acc_Y", "Acc_Z"])
    return datasignals

# Define the folder path where CSV files are located
folder_path = "your_folder_path_here"  # Update this with your folder path

# Get all CSV file paths within the folder
#file_paths = glob.glob(folder_path + "/*.csv")

# Initialize an empty list to store DataFrames
data_frames = []

# Load and concatenate all CSV files
for file_path in training_filepaths_list:
    df = get_data(file_path)  # Load CSV file as DataFrame
    data_frames.append(df)    # Append DataFrame to the list

# Concatenate all DataFrames along the rows (axis=0)
concatenated_data = pd.concat(data_frames, axis=0, ignore_index=True)

# Display the concatenated DataFrame
print(concatenated_data)  # Display the first few rows
```

```
               EEG1        EEG2       Acc_X       Acc_Y       Acc_Z
0        866.904663  842.229919 -531.250854  859.376343  238.281616
1        845.519897  822.490173 -519.532043  855.470093  246.094132
```

```
2       860.324707  835.650024 -519.532043  855.470093  246.094132
3       860.324707  835.650024 -542.969604  851.563843  238.281616
4       835.650024  837.294983 -535.157104  859.376343  238.281616
…             …           …           …           …           …
104475  847.164856  857.034729 -703.126099  750.001160  136.718964
104476  875.129517  837.294983 -703.126099  750.001160  136.718964
104477  852.099793  837.294983 -703.126099  750.001160  136.718964
104478  832.360046  870.194580 -707.032349  746.094910  132.812714
104479  843.874939  843.874939 -703.126099  746.094910  136.718964

[104480 rows x 5 columns]
```

[ ]:

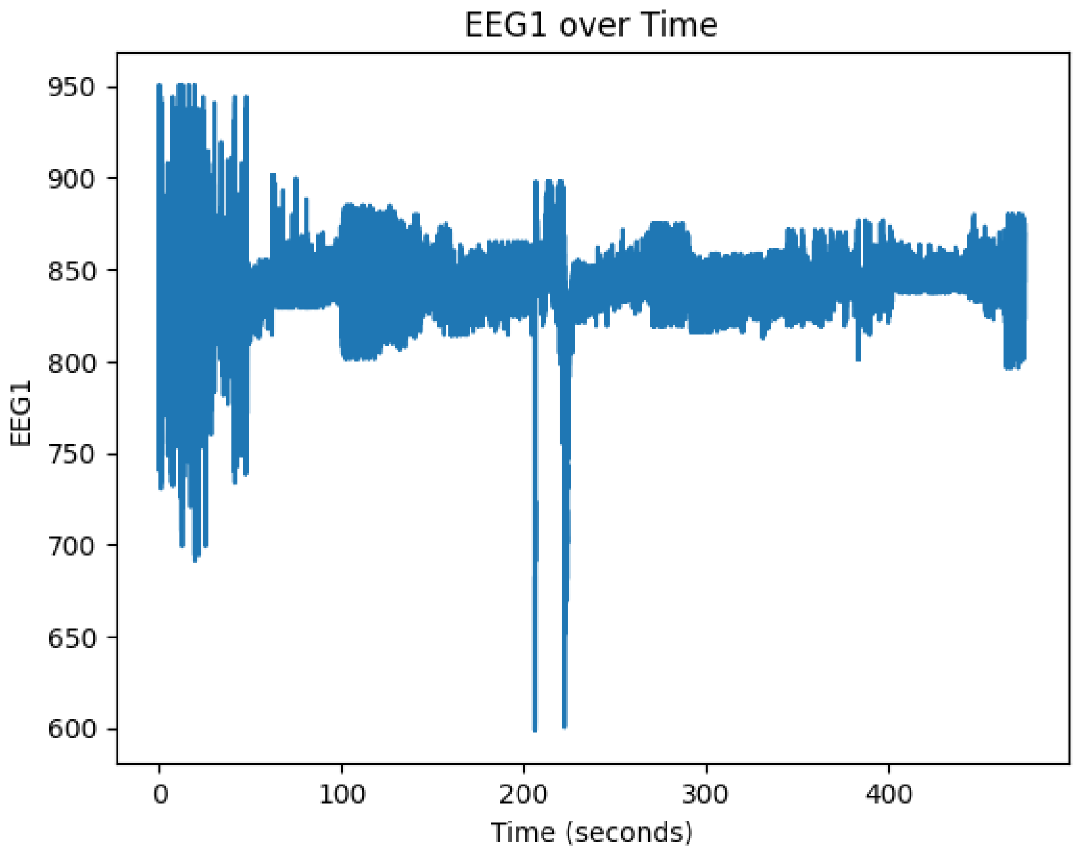
```python
import pandas as pd
import matplotlib.pyplot as plt

# Assuming you have concatenated your data into concatenated_data DataFrame
# Also assuming the sample rate is known and constant for all data points
sample_rate = 220  # Replace this with the actual sample rate

# Calculate time axis based on the sample rate
time = [i / sample_rate for i in range(len(concatenated_data))]

# Add the time axis as a column to the concatenated_data DataFrame
concatenated_data['Time'] = time

# Plotting 'EEG1' over time
plt.plot(concatenated_data['Time'], concatenated_data['EEG1'])
plt.xlabel('Time (seconds)')
plt.ylabel('EEG1')
plt.title('EEG1 over Time')
plt.show()
```



[ ]:

```
import pandas as pd
import plotly.express as px

# Assuming you have concatenated your data into concatenated_data DataFrame
# Also assuming the sample rate is known and constant for all data points
sample_rate = 220  # Replace this with the actual sample rate

# Calculate time axis based on the sample rate
time = [i / sample_rate for i in range(len(concatenated_data))]

# Add the time axis as a column to the concatenated_data DataFrame
concatenated_data['Time'] = time

# Plotting 'EEG1' over time using Plotly
fig = px.line(concatenated_data, x='Time', y='EEG1', title='EEG1 over Time')
fig.update_xaxes(title_text='Time (seconds)')
fig.update_yaxes(title_text='EEG1')
fig.show()
```

[ ]:
```python
import plotly.express as px

# Assuming you have concatenated your data into concatenated_data DataFrame
# Also assuming the sample rate is known and constant for all data points
sample_rate = 220  # Replace this with the actual sample rate

# Calculate time axis based on the sample rate
time_seconds = concatenated_data.index / sample_rate

# Add the time axis as a column to the concatenated_data DataFrame
concatenated_data['Time'] = time_seconds

# Define start and end times for the segment you want to visualize
start_time = 10  # Start time in seconds
end_time = 20    # End time in seconds

# Extract EEG1 segment within the specified time range
eeg1_segment = concatenated_data[(concatenated_data['Time'] >= start_time) &␣
 ↪(concatenated_data['Time'] <= end_time)]

# Plotting a segment of EEG1 over time using Plotly
fig = px.line(eeg1_segment, x='Time', y='EEG1', title='Segment of EEG1 over␣
 ↪Time')
fig.update_xaxes(title_text='Time (seconds)')
fig.update_yaxes(title_text='EEG1')
fig.show()
```

[ ]:
```python
# Assuming you have concatenated your data into concatenated_data DataFrame
# Assuming the sample rate is known and constant for all data points
sample_rate = 220  # Sample rate of 220 samples per second

# Define time ranges for different activities (in seconds)
reading_start_time = 100
reading_end_time = 200
speaking_start_time = 250
speaking_end_time = 350
watching_start_time = 400
watching_end_time = 500

# Create a list to store windowed EEG1 data and corresponding labels
windowed_data = []
labels = []

# Define sliding window parameters
window_size = 50  # Window size in seconds
stride = 50  # Stride length in seconds
```

```
# Iterate through EEG1 signal with sliding window
for i in range(0, len(concatenated_data) - int(window_size * sample_rate),␣
 ↪int(stride * sample_rate)):
    window_start = i / sample_rate
    window_end = (i + window_size * sample_rate) / sample_rate

    # Check and assign label based on the time ranges
    if reading_start_time <= window_start <= reading_end_time or␣
 ↪reading_start_time <= window_end <= reading_end_time:
        label = 'Reading'
    elif speaking_start_time <= window_start <= speaking_end_time or␣
 ↪speaking_start_time <= window_end <= speaking_end_time:
        label = 'Speaking'
    elif watching_start_time <= window_start <= watching_end_time or␣
 ↪watching_start_time <= window_end <= watching_end_time:
        label = 'Watching'


    # Extract EEG1 window and append to the list with its corresponding label
    eeg1_window = concatenated_data['EEG1'].iloc[i:i + int(window_size *␣
 ↪sample_rate)]
    windowed_data.append(eeg1_window)
    labels.append(label)

# Now 'windowed_data' contains EEG1 windows and 'labels' contain corresponding␣
 ↪activity labels
# You can further process 'windowed_data' for analysis or visualization
```

[ ]:
```
eeg1_window, windowed_data, labels
```

[ ]:
```
(88000    853.744812
 88001    850.454834
 88002    855.389771
 88003    848.809875
 88004    852.099793
             …
 98995    850.454834
 98996    848.809875
 98997    848.809875
 98998    845.519897
 98999    843.874939
 Name: EEG1, Length: 11000, dtype: float64,
 [0        866.904663
  1        845.519897
  2        860.324707
  3        860.324707
```

```
4        835.650024
            …
10995    838.940002
10996    832.360046
10997    830.715088
10998    837.294983
10999    835.650024
Name: EEG1, Length: 11000, dtype: float64,
11000    840.584961
11001    835.650024
11002    830.715088
11003    827.425110
11004    834.005005
            …
21995    840.584961
21996    840.584961
21997    840.584961
21998    850.454834
21999    850.454834
Name: EEG1, Length: 11000, dtype: float64,
22000    850.454834
22001    848.809875
22002    848.809875
22003    848.809875
22004    848.809875
            …
32995    852.099793
32996    845.519897
32997    858.679748
32998    837.294983
32999    840.584961
Name: EEG1, Length: 11000, dtype: float64,
33000    832.360046
33001    842.229919
33002    848.809875
33003    855.389771
33004    832.360046
            …
43995    832.360046
43996    848.809875
43997    842.229919
43998    848.809875
43999    848.809875
Name: EEG1, Length: 11000, dtype: float64,
44000    845.519897
44001    832.360046
44002    853.744812
```

```
44003    840.584961
44004    852.099793
            …
54995    843.874939
54996    845.519897
54997    842.229919
54998    840.584961
54999    843.874939
Name: EEG1, Length: 11000, dtype: float64,
55000    842.229919
55001    847.164856
55002    847.164856
55003    847.164856
55004    855.389771
            …
65995    840.584961
65996    847.164856
65997    847.164856
65998    847.164856
65999    835.650024
Name: EEG1, Length: 11000, dtype: float64,
66000    832.360046
66001    832.360046
66002    827.425110
66003    824.135132
66004    838.940002
            …
76995    852.099793
76996    843.874939
76997    850.454834
76998    842.229919
76999    840.584961
Name: EEG1, Length: 11000, dtype: float64,
77000    829.070068
77001    842.229919
77002    834.005005
77003    838.940002
77004    842.229919
            …
87995    842.229919
87996    852.099793
87997    835.650024
87998    835.650024
87999    838.940002
Name: EEG1, Length: 11000, dtype: float64,
88000    853.744812
88001    850.454834
```

```
 88002    855.389771
 88003    848.809875
 88004    852.099793
                …
 98995    850.454834
 98996    848.809875
 98997    848.809875
 98998    845.519897
 98999    843.874939
 Name: EEG1, Length: 11000, dtype: float64],
[0,
 'Reading',
 'Reading',
 'Reading',
 'Reading',
 'Speaking',
 'Speaking',
 'Speaking',
 'Watching'])
```

[ ]:
```
len(concatenated_data['EEG1'])
```

[ ]: 104480

[ ]:
```
import plotly.graph_objs as go
from plotly.subplots import make_subplots

# Assuming you have concatenated your data into concatenated_data DataFrame
# Extracting EEG1, EEG2, Acc_X, Acc_Y, Acc_Z data
eeg1_data = concatenated_data['EEG1']
eeg2_data = concatenated_data['EEG2']
acc_x_data = concatenated_data['Acc_X']
acc_y_data = concatenated_data['Acc_Y']
acc_z_data = concatenated_data['Acc_Z']

# Create subplots
fig = make_subplots(rows=2, cols=1, subplot_titles=("EEG1 and EEG2 Signals",␣
 ↪"Accelerometer Signals"))

# Add EEG1 and EEG2 traces to subplot 1
fig.add_trace(go.Scatter(x=concatenated_data.index, y=eeg1_data, mode='lines',␣
 ↪name='EEG1'), row=1, col=1)
fig.add_trace(go.Scatter(x=concatenated_data.index, y=eeg2_data, mode='lines',␣
 ↪name='EEG2'), row=1, col=1)
fig.update_xaxes(title_text="Time", row=1, col=1)
fig.update_yaxes(title_text="EEG Value", row=1, col=1)
```

```python
# Add Acc_X, Acc_Y, Acc_Z traces to subplot 2
fig.add_trace(go.Scatter(x=concatenated_data.index, y=acc_x_data, mode='lines',␣
↪name='Acc_X'), row=2, col=1)
fig.add_trace(go.Scatter(x=concatenated_data.index, y=acc_y_data, mode='lines',␣
↪name='Acc_Y'), row=2, col=1)
fig.add_trace(go.Scatter(x=concatenated_data.index, y=acc_z_data, mode='lines',␣
↪name='Acc_Z'), row=2, col=1)
fig.update_xaxes(title_text="Time", row=2, col=1)
fig.update_yaxes(title_text="Acceleration Value", row=2, col=1)

fig.update_layout(height=600, width=800, title_text="EEG and Accelerometer␣
↪Signals")
fig.show()
```

[ ]:
```python
import plotly.graph_objects as go
from plotly.subplots import make_subplots

# Assuming you have concatenated your data into concatenated_data DataFrame
# Extracting EEG1, EEG2, Acc_X, Acc_Y, Acc_Z data
eeg1_data = concatenated_data['EEG1']
eeg2_data = concatenated_data['EEG2']
acc_x_data = concatenated_data['Acc_X']
acc_y_data = concatenated_data['Acc_Y']
acc_z_data = concatenated_data['Acc_Z']

# Create subplots
fig = make_subplots(rows=5, cols=1, shared_xaxes=True, vertical_spacing=0.03)

# Add EEG1 trace to subplot 1
fig.add_trace(go.Scatter(x=concatenated_data.index, y=eeg1_data, mode='lines',␣
↪name='EEG1'), row=1, col=1)
fig.update_xaxes(title_text="Time", row=1, col=1)
fig.update_yaxes(title_text="EEG1 Value", row=1, col=1)

# Add EEG2 trace to subplot 2
fig.add_trace(go.Scatter(x=concatenated_data.index, y=eeg2_data, mode='lines',␣
↪name='EEG2'), row=2, col=1)
fig.update_xaxes(title_text="Time", row=2, col=1)
fig.update_yaxes(title_text="EEG2 Value", row=2, col=1)

# Add Acc_X trace to subplot 3
fig.add_trace(go.Scatter(x=concatenated_data.index, y=acc_x_data, mode='lines',␣
↪name='Acc_X'), row=3, col=1)
fig.update_xaxes(title_text="Time", row=3, col=1)
fig.update_yaxes(title_text="Acc_X Value", row=3, col=1)
```

```
# Add Acc_Y trace to subplot 4
fig.add_trace(go.Scatter(x=concatenated_data.index, y=acc_y_data, mode='lines',␣
 ↪name='Acc_Y'), row=4, col=1)
fig.update_xaxes(title_text="Time", row=4, col=1)
fig.update_yaxes(title_text="Acc_Y Value", row=4, col=1)

# Add Acc_Z trace to subplot 5
fig.add_trace(go.Scatter(x=concatenated_data.index, y=acc_z_data, mode='lines',␣
 ↪name='Acc_Z'), row=5, col=1)
fig.update_xaxes(title_text="Time", row=5, col=1)
fig.update_yaxes(title_text="Acc_Z Value", row=5, col=1)

fig.update_layout(height=800, showlegend=False, title_text="EEG and␣
 ↪Accelerometer Signals")
fig.show()
```