Internet of Things

Assignment 2

Name: Mohd. Rizwan Shaikh
Roll no. IMT2019513

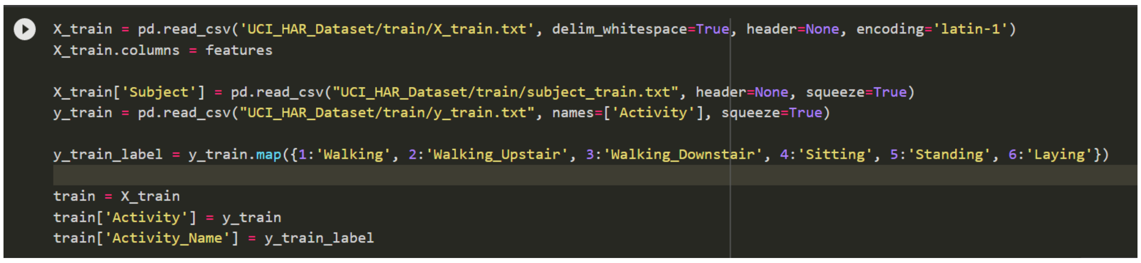
Part 1. Go through and understand the dataset described in [1]. It contains accelerometer and gyroscope sensor data collected using a smartphone.

The given dataset is built from the recordings of 30 volunteers performing daily tasks while carrying a waist-mounted smartphone with embedded inertial sensors. All the volunteers are aged between 19-48 years and perform one of WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING and LAYING. The dataset is recorded using Samsung Galaxy S2. The linear and angular acceleration and velocity has been captured at a rate of 50 Hz using the embedded accelerometer and gyroscope. The obtained dataset is split into training data and testing data in 7:3 ratio.

The dataset contains the following files:

• *'README.txt'*	
• *'features_info.txt'*	: Gives a brief overview of the dataset
• *'features.txt'*	: List of all the features
• *'activity_labels.txt'*	: Maps the class label with activity name
• *'train/X_train.txt'*	: Training features
• *'train/y_train.txt'*	: Training labels
• *'test/X_test.txt'*	: Testing features
• *'test/y_test.txt'*	: Testing labels

The code snippet used to load the training dataset into dataframe is shown below:

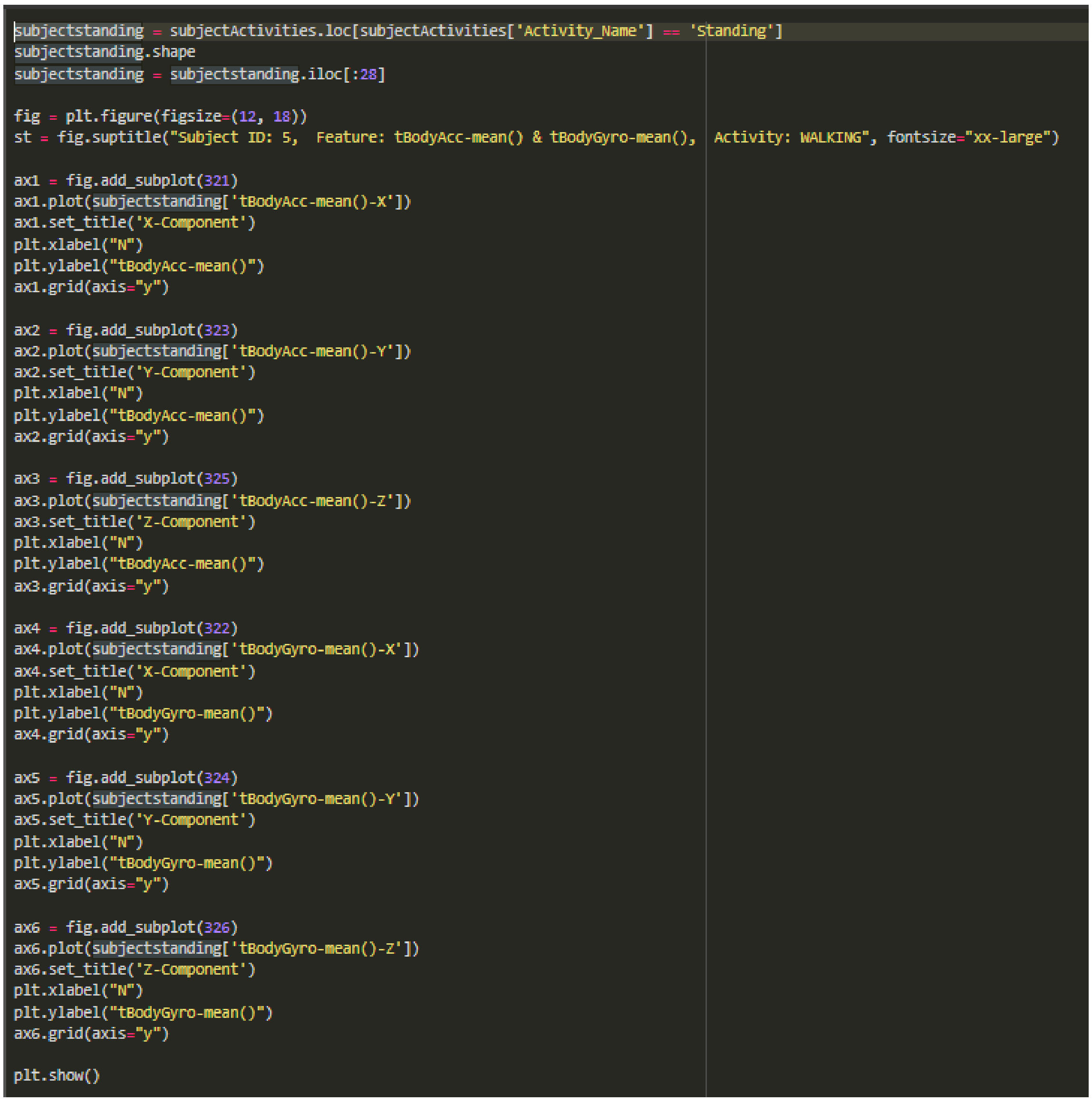

```python
X_train = pd.read_csv('UCI_HAR_Dataset/train/X_train.txt', delim_whitespace=True, header=None, encoding='latin-1')
X_train.columns = features

X_train['Subject'] = pd.read_csv("UCI_HAR_Dataset/train/subject_train.txt", header=None, squeeze=True)
y_train = pd.read_csv("UCI_HAR_Dataset/train/y_train.txt", names=['Activity'], squeeze=True)

y_train_label = y_train.map({1:'Walking', 2:'Walking_Upstair', 3:'Walking_Downstair', 4:'Sitting', 5:'Standing', 6:'Laying'})

train = X_train
train['Activity'] = y_train
train['Activity_Name'] = y_train_label
```

Test dataset can also be loaded into pandas dataframe using same logic.

Part 2. With small sized portions of the data, plot a graph for each human activity.

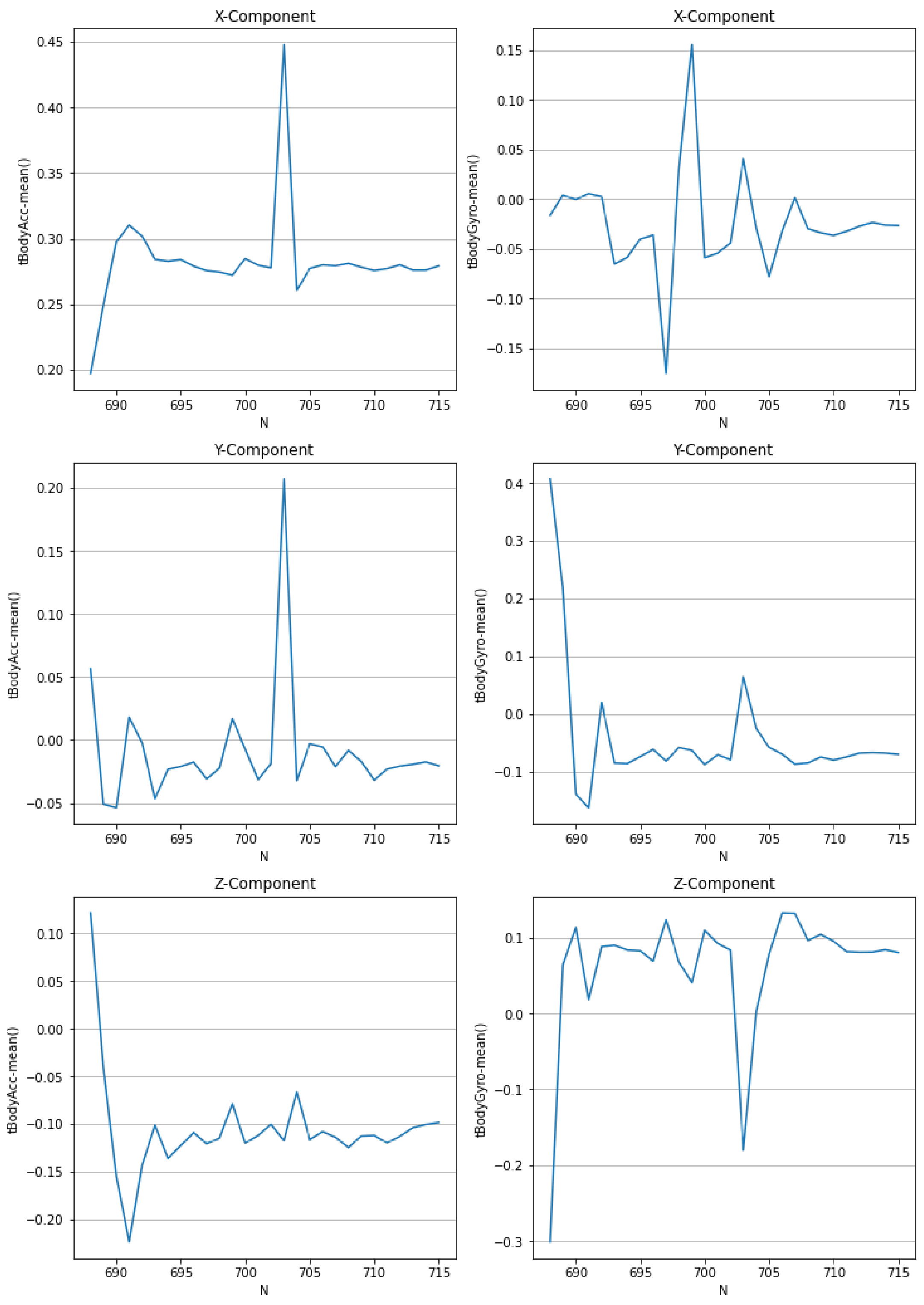
To plot the data, subject 5 is chosen. The features picked are *tBodyAcc-mean()* (accelerometer) and *tBodyGyro-mean()* (gyroscope). The data is plotted for X, Y and Z directions. Plots are obtained for all the six activities namely WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING and LAYING using a small subset of data from each activity.

The code snippet used for plotting the data for STANDING activity is shown below. By making relevant changes, the code snippet can be used to plot data for other activities as well.

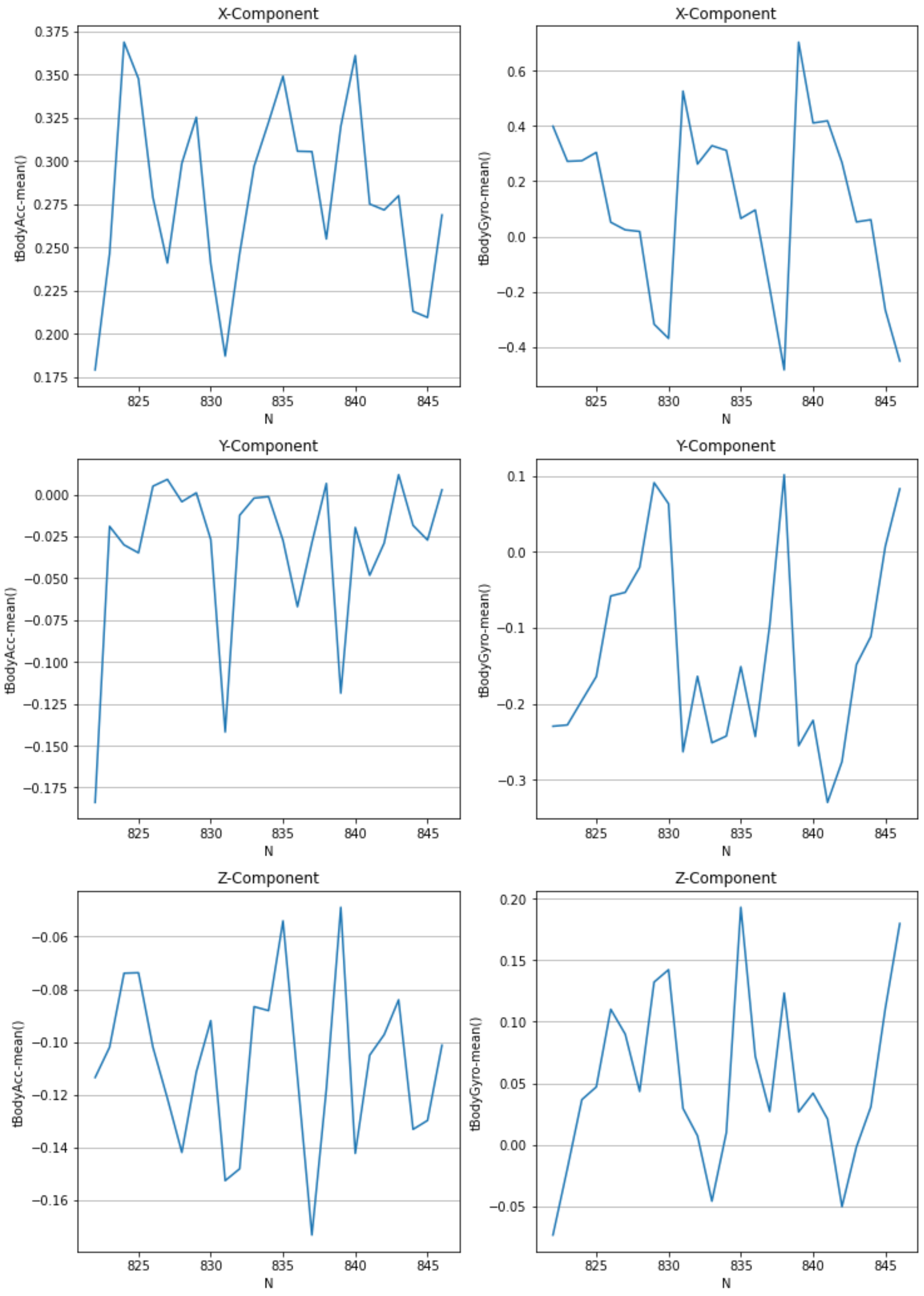


```
subjectstanding = subjectActivities.loc[subjectActivities['Activity_Name'] == 'Standing']
subjectstanding.shape
subjectstanding = subjectstanding.iloc[:28]

fig = plt.figure(figsize=(12, 18))
st = fig.suptitle("Subject ID: 5,  Feature: tBodyAcc-mean() & tBodyGyro-mean(),  Activity: WALKING", fontsize="xx-large")

ax1 = fig.add_subplot(321)
ax1.plot(subjectstanding['tBodyAcc-mean()-X'])
ax1.set_title('X-Component')
plt.xlabel("N")
plt.ylabel("tBodyAcc-mean()")
ax1.grid(axis="y")

ax2 = fig.add_subplot(323)
ax2.plot(subjectstanding['tBodyAcc-mean()-Y'])
ax2.set_title('Y-Component')
plt.xlabel("N")
plt.ylabel("tBodyAcc-mean()")
ax2.grid(axis="y")

ax3 = fig.add_subplot(325)
ax3.plot(subjectstanding['tBodyAcc-mean()-Z'])
ax3.set_title('Z-Component')
plt.xlabel("N")
plt.ylabel("tBodyAcc-mean()")
ax3.grid(axis="y")

ax4 = fig.add_subplot(322)
ax4.plot(subjectstanding['tBodyGyro-mean()-X'])
ax4.set_title('X-Component')
plt.xlabel("N")
plt.ylabel("tBodyGyro-mean()")
ax4.grid(axis="y")

ax5 = fig.add_subplot(324)
ax5.plot(subjectstanding['tBodyGyro-mean()-Y'])
ax5.set_title('Y-Component')
plt.xlabel("N")
plt.ylabel("tBodyGyro-mean()")
ax5.grid(axis="y")

ax6 = fig.add_subplot(326)
ax6.plot(subjectstanding['tBodyGyro-mean()-Z'])
ax6.set_title('Z-Component')
plt.xlabel("N")
plt.ylabel("tBodyGyro-mean()")
ax6.grid(axis="y")

plt.show()
```

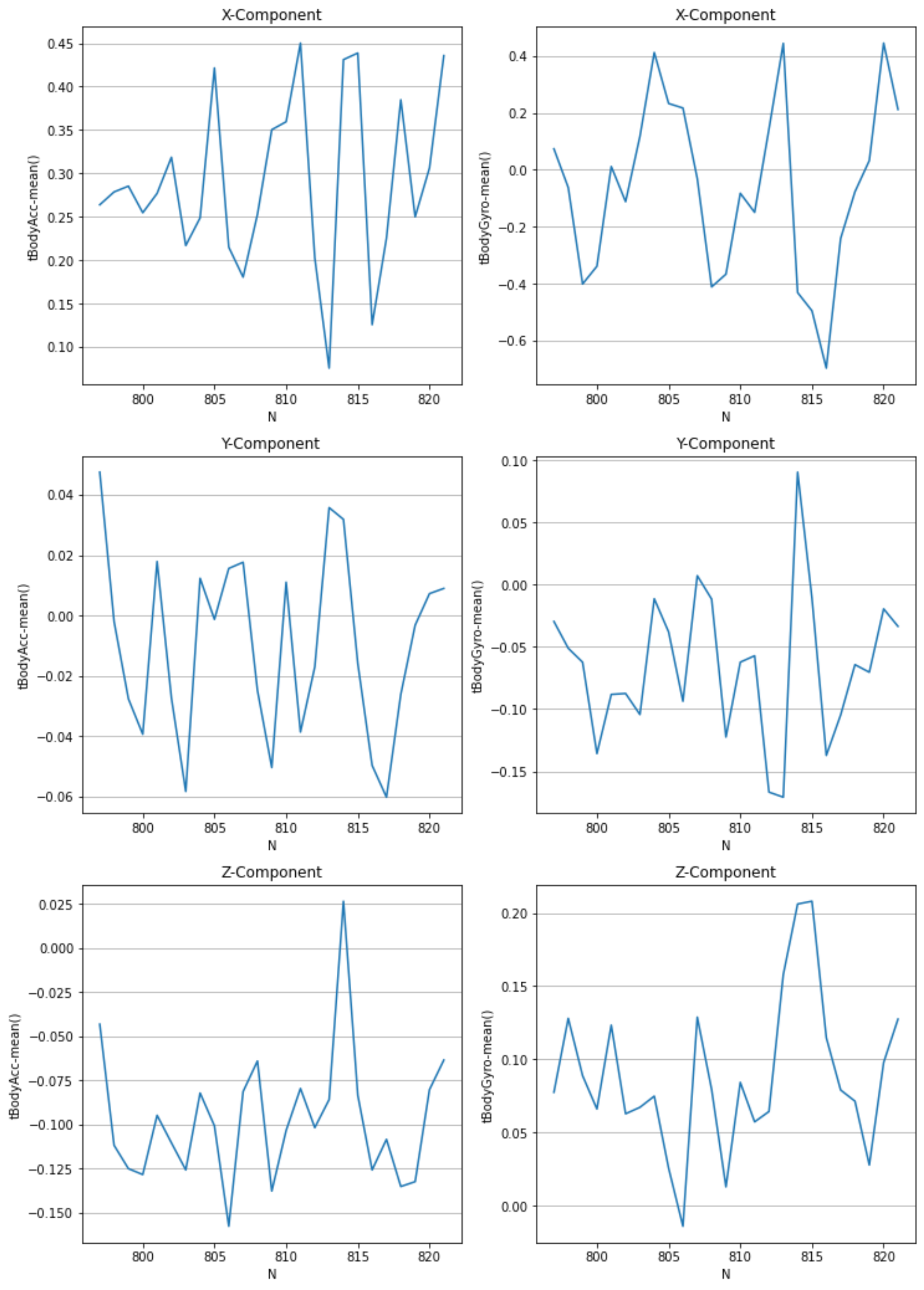
Subject ID: 5, Feature: tBodyAcc-mean() & tBodyGyro-mean(), Activity: WALKING



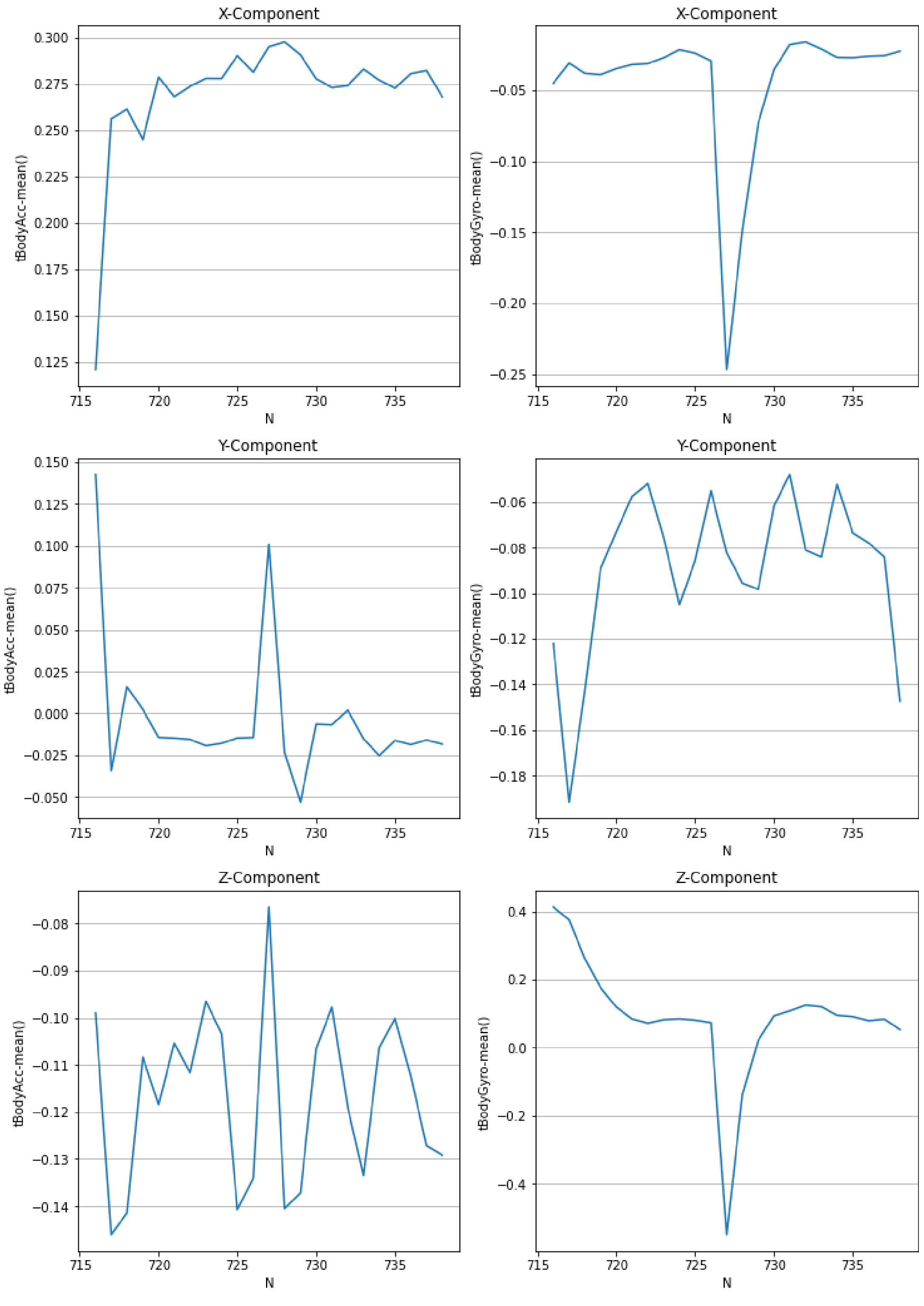
Subject ID: 5, Feature: tBodyAcc-mean() & tBodyGyro-mean(), Activity: WALKING_UPSTAIRS



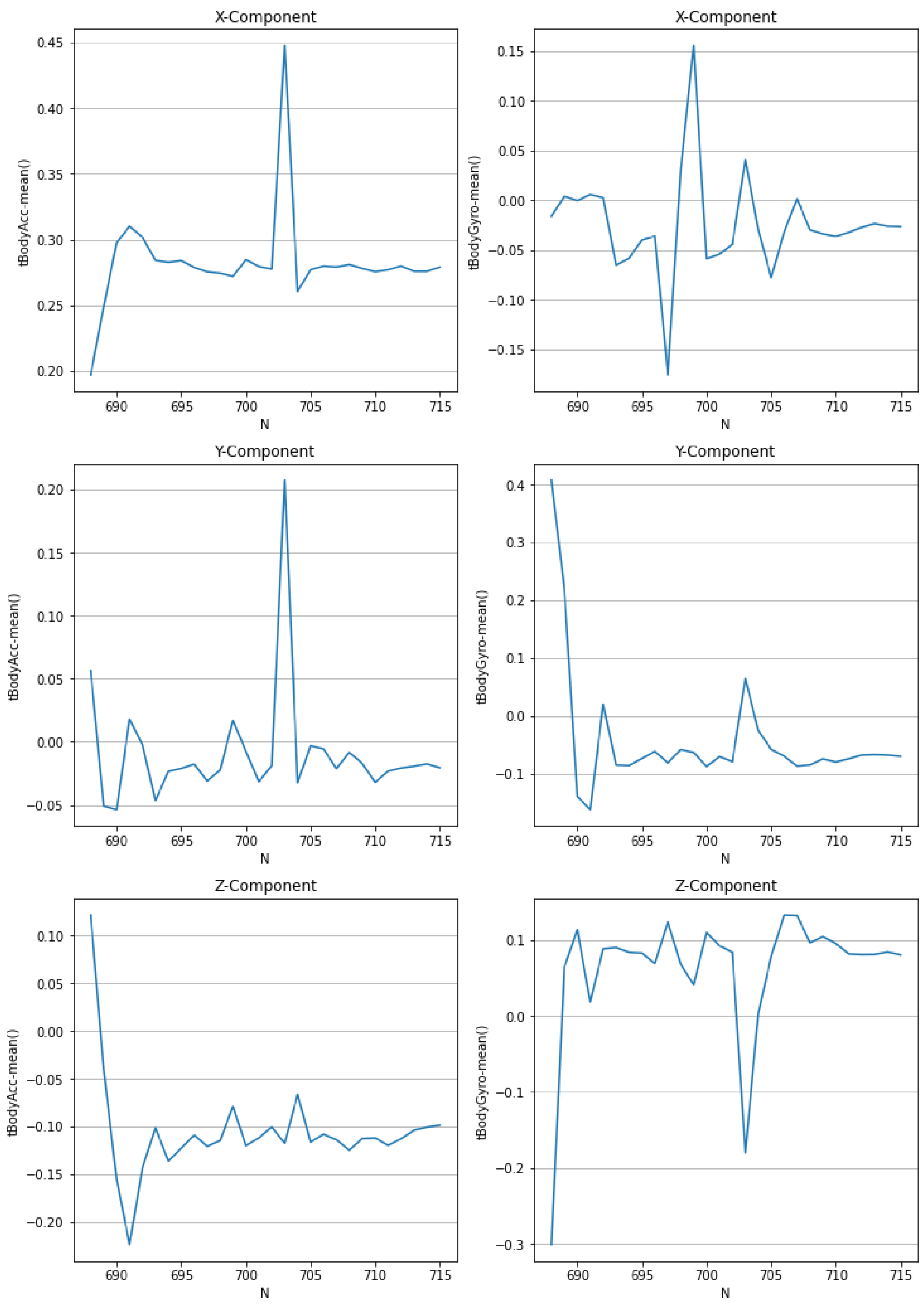
Subject ID: 5, Feature: tBodyAcc-mean() & tBodyGyro-mean(), Activity: WALKING_DOWNSTAIRS



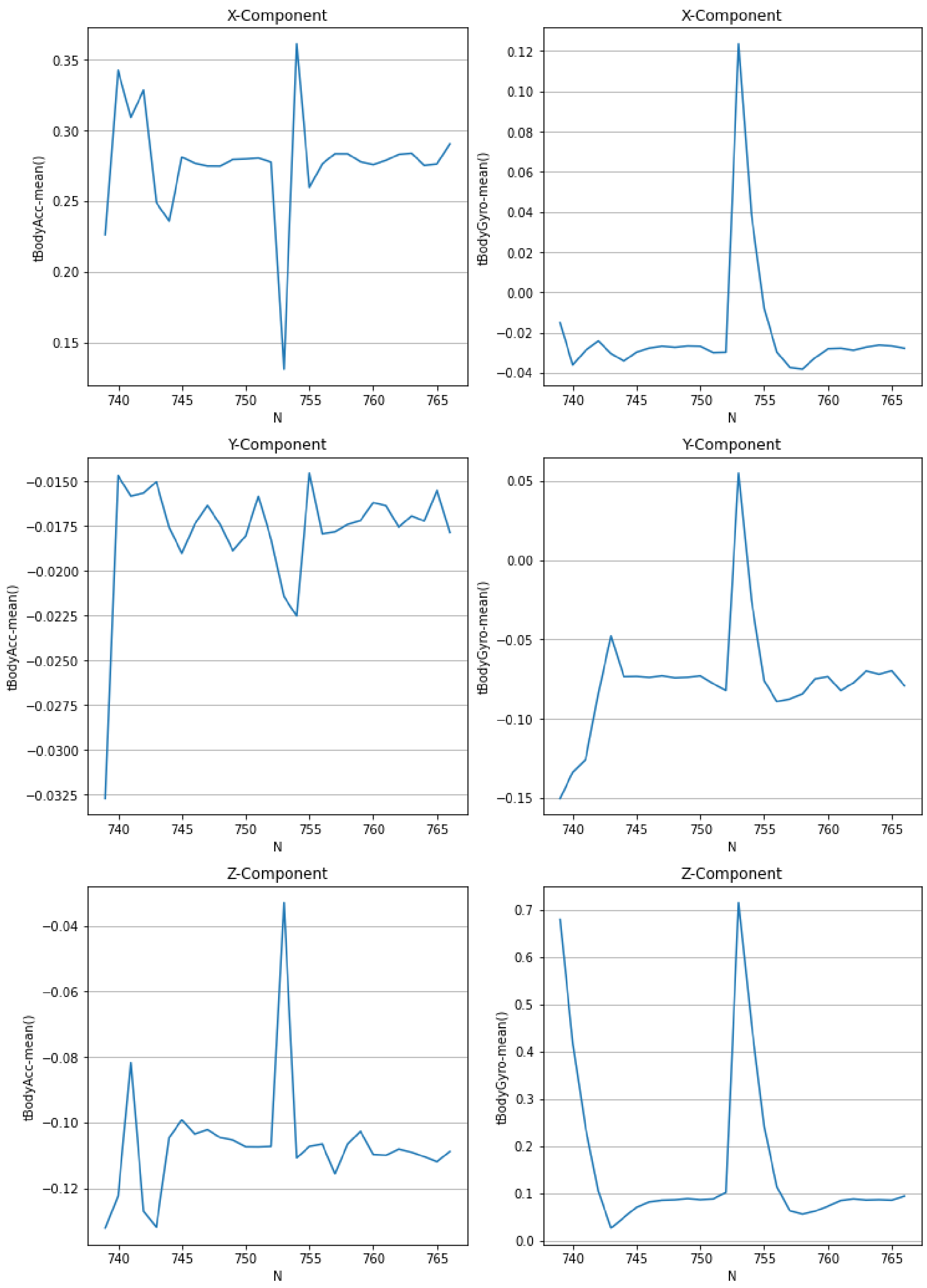
Subject ID: 5, Feature: tBodyAcc-mean() & tBodyGyro-mean(), Activity: SITTING



Subject ID: 5, Feature: tBodyAcc-mean() & tBodyGyro-mean(), Activity: STANDING



Subject ID: 5, Feature: tBodyAcc-mean() & tBodyGyro-mean(), Activity: LAYING

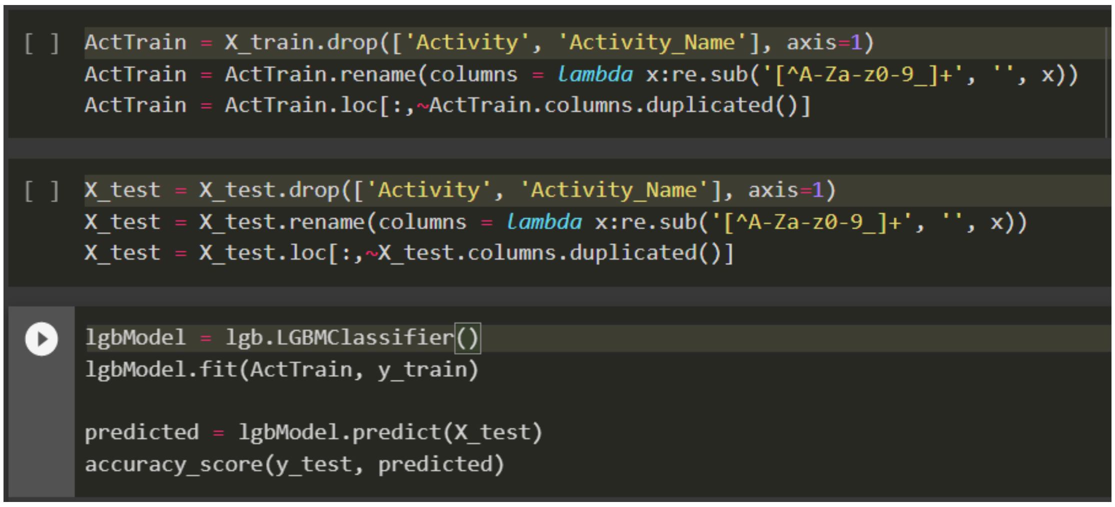


Part 3. Develop algorithms for activity detection and behaviour change detection [2]. If the dataset has individual files for each activity, then try to mix them up in portions to get a mixed bag of data, which can be used for the behaviour change detection. Use any programming tool of your choice to implement the same.

ACTIVITY DETECTION:

In order to predict activity, LightGBM model is implemented. It is a light gradient boosting framework that is built on tree-based models for classification tasks.

The code for activity detection is shown below:



```
[ ] ActTrain = X_train.drop(['Activity', 'Activity_Name'], axis=1)
    ActTrain = ActTrain.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
    ActTrain = ActTrain.loc[:,~ActTrain.columns.duplicated()]

[ ] X_test = X_test.drop(['Activity', 'Activity_Name'], axis=1)
    X_test = X_test.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
    X_test = X_test.loc[:,~X_test.columns.duplicated()]

    lgbModel = lgb.LGBMClassifier()
    lgbModel.fit(ActTrain, y_train)

    predicted = lgbModel.predict(X_test)
    accuracy_score(y_test, predicted)
```

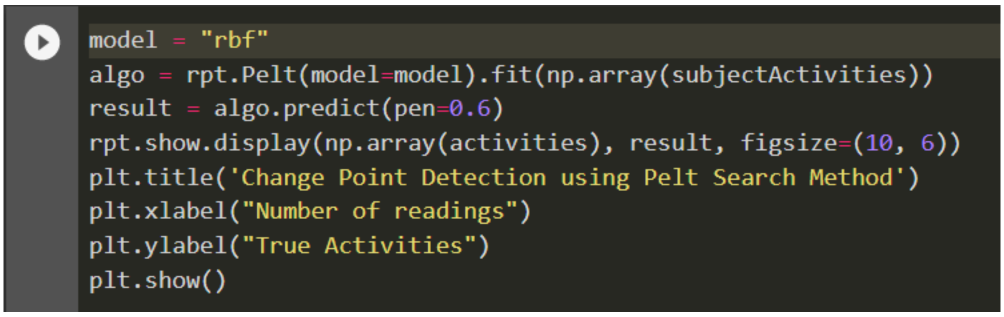
This model was able to predict the activity with an accuracy of 0.9317950458092976.

BEHAVIOUR CHANGE DETECTION:

In order to detect behaviour change, *Ruptures* library (https://github.com/deepcharles/ruptures) is used. It is a well-known library which performs change point detection on the input data.

In ruptures method, PELT method is used. The PELT (*Pruned Exact Linear Time*) method is an exact method, and generally produces quick and consistent results. It detects change points through the minimization of costs. The algorithm has a computational cost of O(n), where n is the number of data points.

The behaviour change is detected for the activities performed by subject 2. The code for detecting behaviour change is shown below:

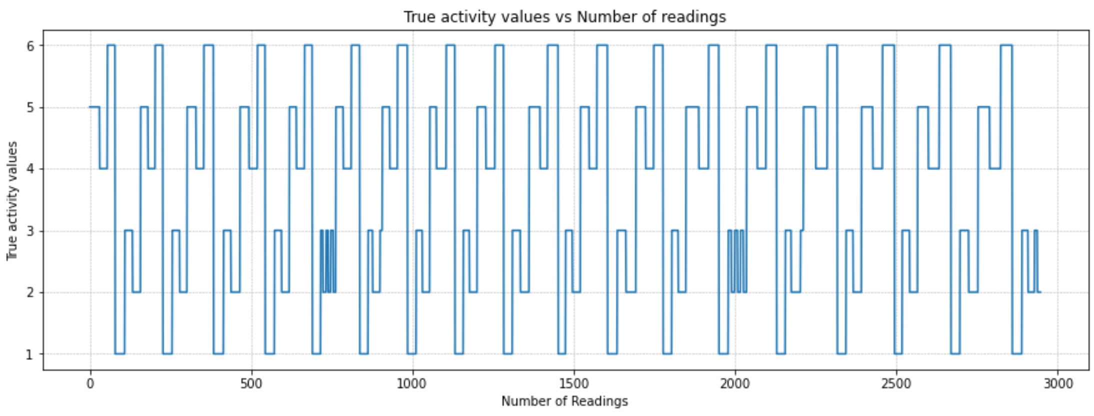


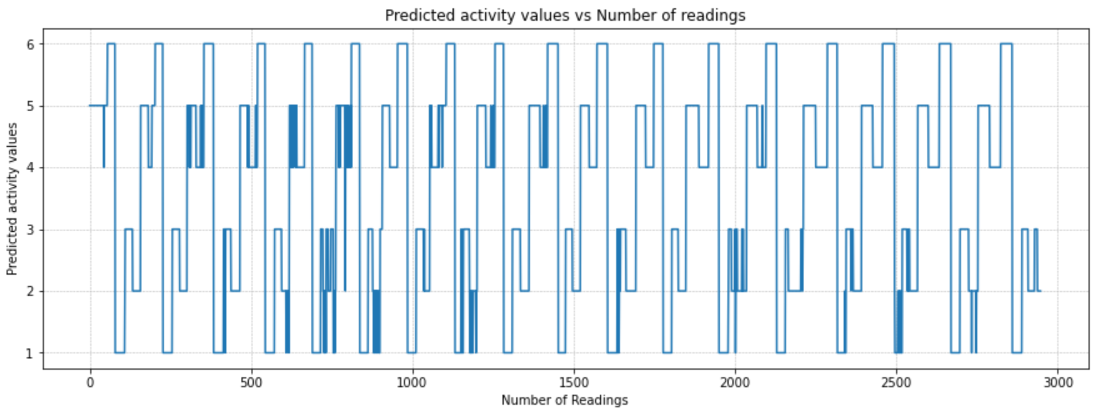
```
model = "rbf"
algo = rpt.Pelt(model=model).fit(np.array(subjectActivities))
result = algo.predict(pen=0.6)
rpt.show.display(np.array(activities), result, figsize=(10, 6))
plt.title('Change Point Detection using Pelt Search Method')
plt.xlabel("Number of readings")
plt.ylabel("True Activities")
plt.show()
```

Part 4. Show the activity detection and behaviour change graphically.

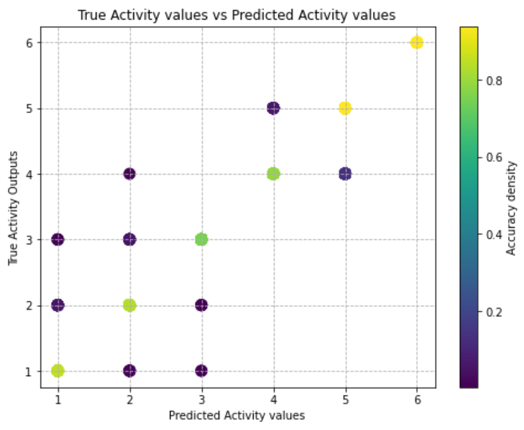
<u>ACTIVITY DETECTION:</u>

The plots for true activity values and predicted activity values are shown below:



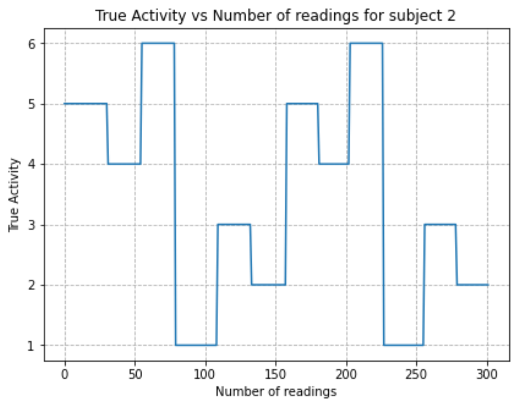


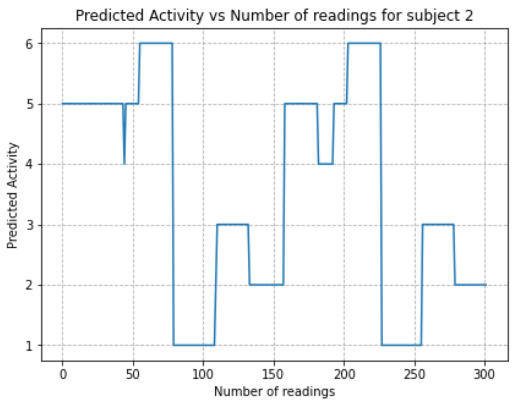
In the density scatter graph shown below (True activity outputs vs predicted activity values), we see that the model has been able to predict most of the activities accurately. This is observed by the green-yellow points along the line y = x. Some dark blue spots that are present are the misclassified points.



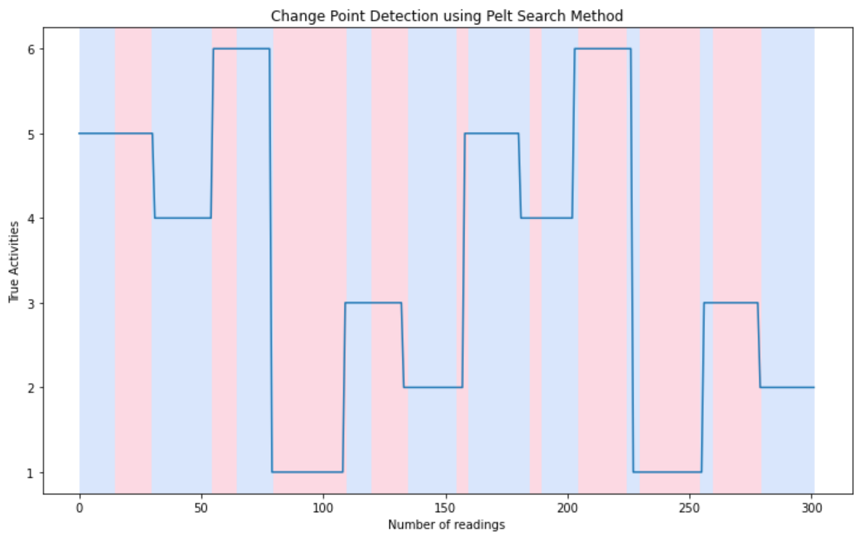
BEHAVIOUR CHANGE DETECTION:

The plots of true activities and predicted activities for subject 2 are shown below:





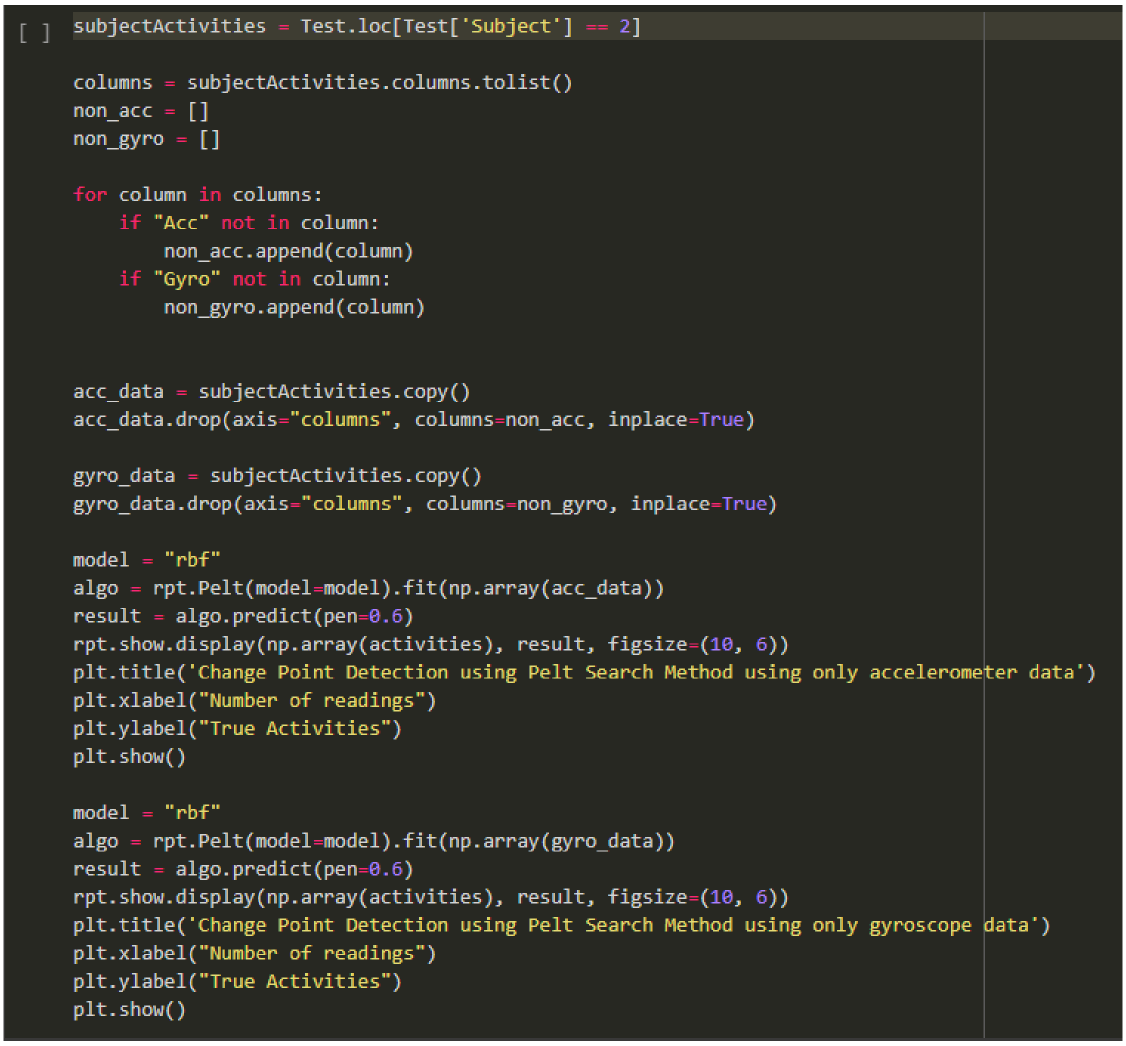
The plot for behaviour change detection is shown below. A change is colour is observed when the model detects an abrupt change and it is able to detect the change in performed activities to a certain level.



Part 5. Is accelerometer or gyroscope data alone sufficient to detect the behaviour change. Or will using both improve the detection. Substantiate.

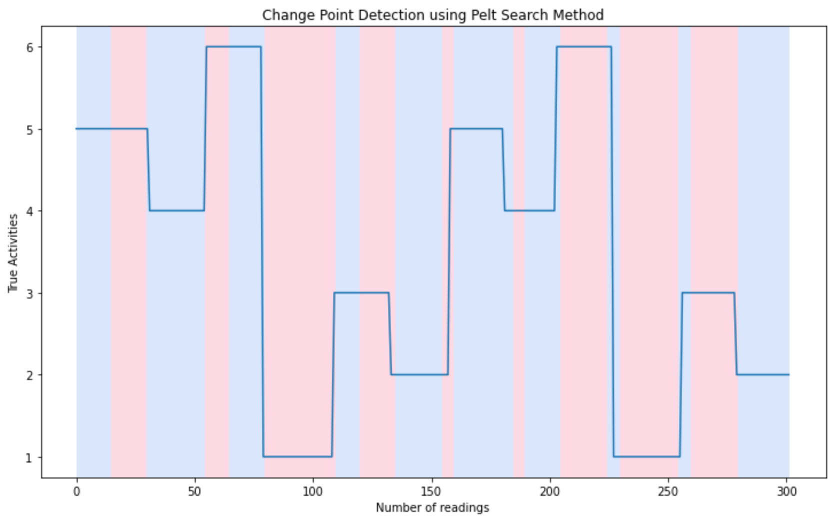
In order to do this question, two dataframes are obtained. The first one contains only the accelerometer data and the second one contains only the gyroscope data. Behaviour change is detected using the above two dataframes. After this, behaviour change is detected using the whole training dataset. This is done because the whole dataset contains both accelerometer and gyroscope data. The ruptures point detection algorithm is used on the datasets for detecting behaviour change.

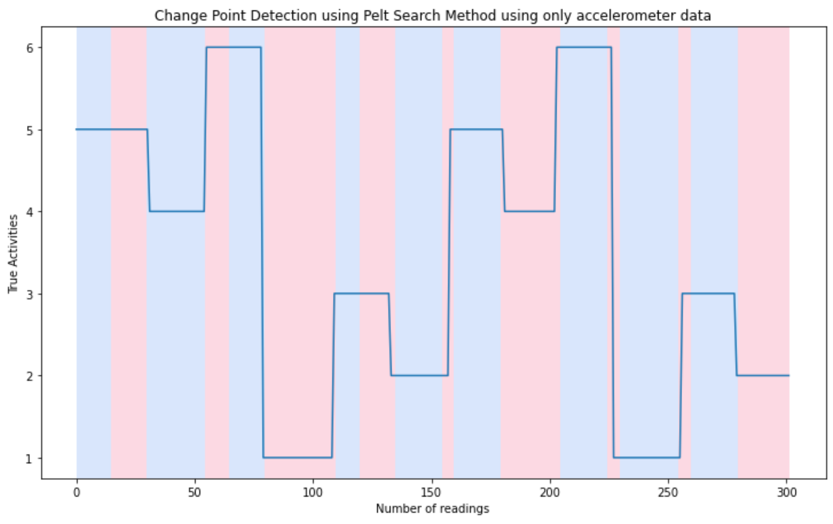
The dataset used for this task contains activities performed by subject 2. The code used is shown below:

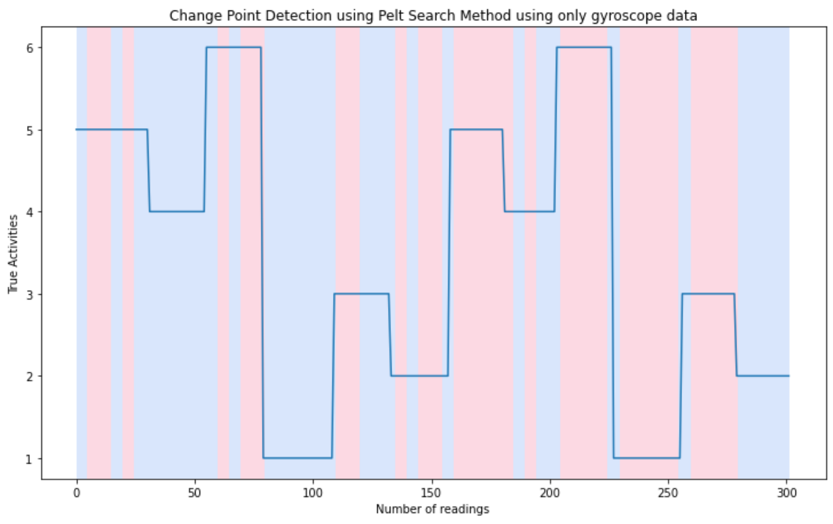


```
[ ] subjectActivities = Test.loc[Test['Subject'] == 2]

    columns = subjectActivities.columns.tolist()
    non_acc = []
    non_gyro = []

    for column in columns:
        if "Acc" not in column:
            non_acc.append(column)
        if "Gyro" not in column:
            non_gyro.append(column)


    acc_data = subjectActivities.copy()
    acc_data.drop(axis="columns", columns=non_acc, inplace=True)

    gyro_data = subjectActivities.copy()
    gyro_data.drop(axis="columns", columns=non_gyro, inplace=True)

    model = "rbf"
    algo = rpt.Pelt(model=model).fit(np.array(acc_data))
    result = algo.predict(pen=0.6)
    rpt.show.display(np.array(activities), result, figsize=(10, 6))
    plt.title('Change Point Detection using Pelt Search Method using only accelerometer data')
    plt.xlabel("Number of readings")
    plt.ylabel("True Activities")
    plt.show()

    model = "rbf"
    algo = rpt.Pelt(model=model).fit(np.array(gyro_data))
    result = algo.predict(pen=0.6)
    rpt.show.display(np.array(activities), result, figsize=(10, 6))
    plt.title('Change Point Detection using Pelt Search Method using only gyroscope data')
    plt.xlabel("Number of readings")
    plt.ylabel("True Activities")
    plt.show()
```

The three behaviour change detection plots are shown below:







On comparing the first figure (both accelerometer and gyroscope data) with the second figure (only accelerometer data), we observe that figures are almost identical. However, on close observation we notice that first figure has better change detection as compared to second figure.

On comparing first (both accelerometer and gyroscope data) and third figure (only gyroscope data), we notice that the change detection in third figure doesn’t have much accuracy as compared to first figure. This is concluded by the unnecessary colour toggling observed in the third figure.

Thus, from the figures, we conclude that behaviour change detection is much better with readings from both accelerometer and gyroscope data as compared to data considered from accelerometer and gyroscope individually.

References:

1. *A Brief Introduction to Change Point Detection using Python:*

A Brief Introduction to Change Point Detection using Python - Tech Rando