Internet Of Things Assignment- 2

Name: Kalahasti V V Deepesh

Roll Number: IMT2019508

Email ID: deepesh.vv@iiitb.ac.in

Question 1: Go through and understand the dataset. It contains accelerometer and gyroscope sensor data collected using a smartphone.

Answer:

The dataset that was given to us contains reading collected from accelerometer and gyroscope sensors from a Samsung Galaxy S2 smartphone of 30 people within the age of 19 – 48 years old.

The data was collected when the people were performing a certain set of pre – determined activities namely Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing and Laying. The dataset is in the form of test and train datasets making it useful to fit machine learning models and perform predictions.

Pre-processing of the dataset: I read the dataset using pandas and then dropped the missing value and duplicate values.

```
x_train = pd.read_csv('../input/local-dataset/UCI HAR Dataset/train/X_train.txt', delim_whitespace = True, header = None, names = features_list)

subject_train = pd.read_csv('../input/local-dataset/UCI HAR Dataset/train/subject_train.txt', header = None, squeeze = True)

x_train["Subject"] = subject_train.tolist()  # Creating a new column called Subject which contains all of the subjects.

activity = pd.read_csv('../input/local-dataset/UCI HAR Dataset/train/y_train.txt', names = ["Activity"], squeeze = True)
x_train["Activity"] = activity.tolist()  # Creating a new column called Activity which contains all of the activities of the subjects.
```

+ Code + Markdown

```
x_test = pd.read_csv('../input/local-dataset/UCI HAR Dataset/test/X_test.txt', delim_whitespace = True, header = None, names = features_list)

subject_test = pd.read_csv('../input/local-dataset/UCI HAR Dataset/test/subject_test.txt', header = None, squeeze = True)

x_test["Subject"] = subject_test.tolist()

activity = pd.read_csv('../input/local-dataset/UCI HAR Dataset/test/y_test.txt', names = ["Activity"], squeeze = True)
x_test["Activity"] = activity.tolist()
```

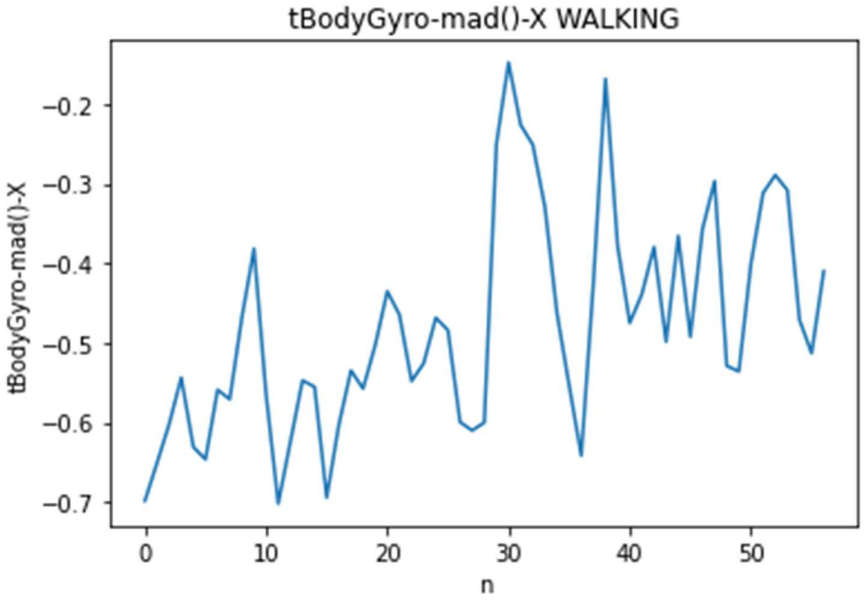
```
x_train = x_train.dropna()
x_test = x_test.dropna()
x_train = x_train.drop_duplicates()
x_test = x_test.drop_duplicates()
```

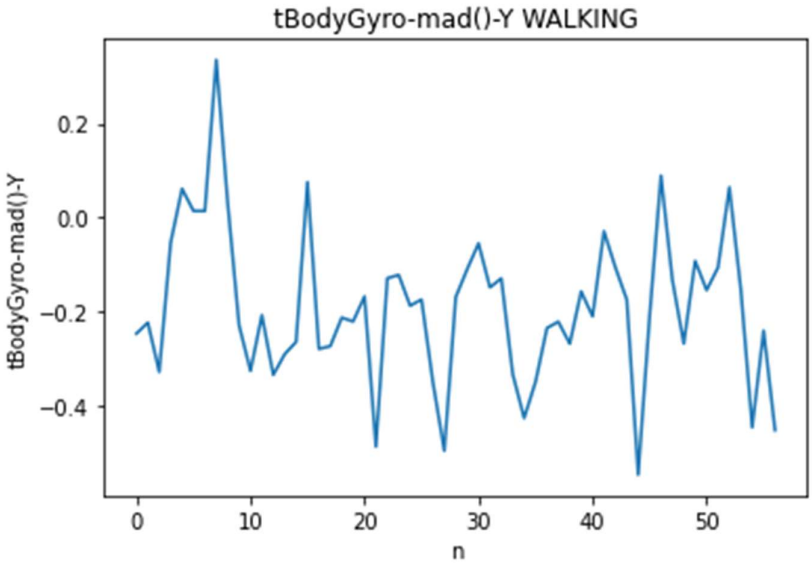
Question 2: With small sized portions of the data, plot a graph for each human activity.

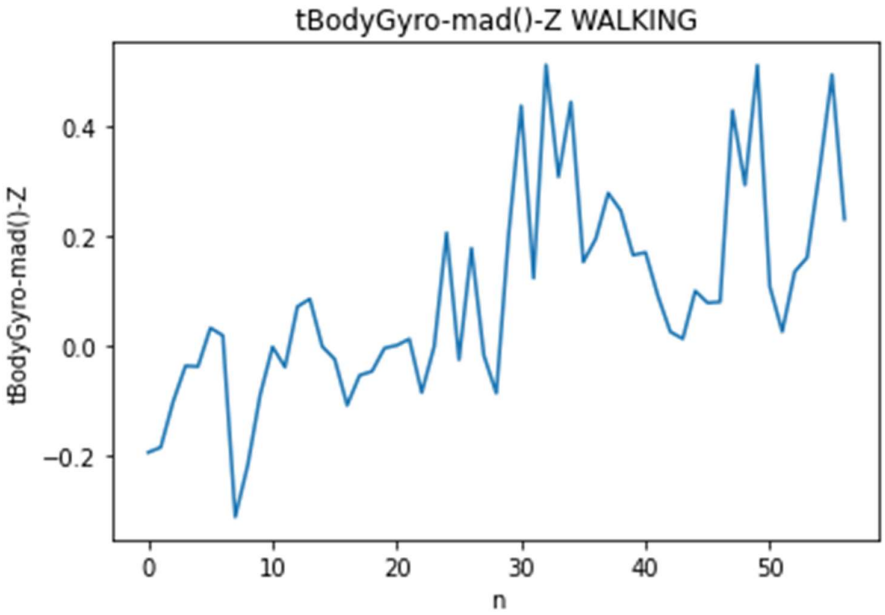
Answer:

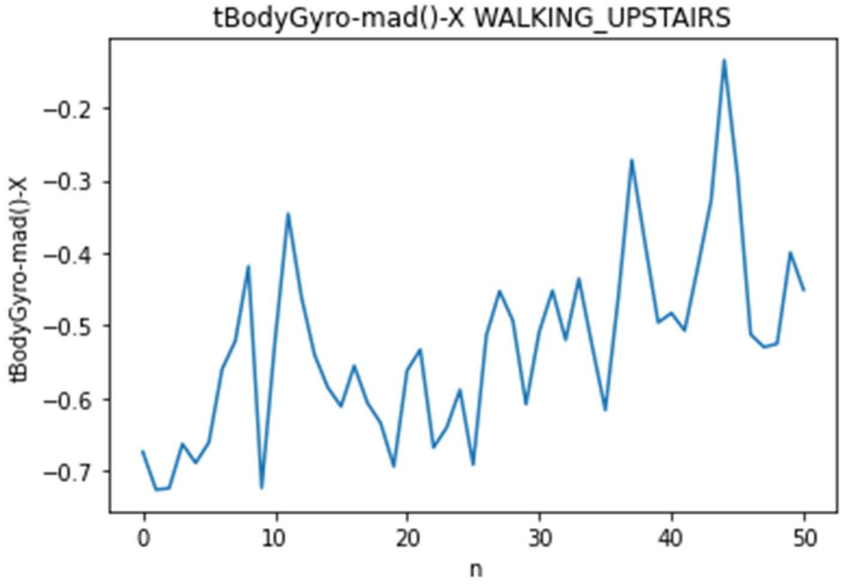
We choose the tBodyGyro-mad()- feature along all of the axes and we choose the subject 6 for plotting the graphs for each human activity.

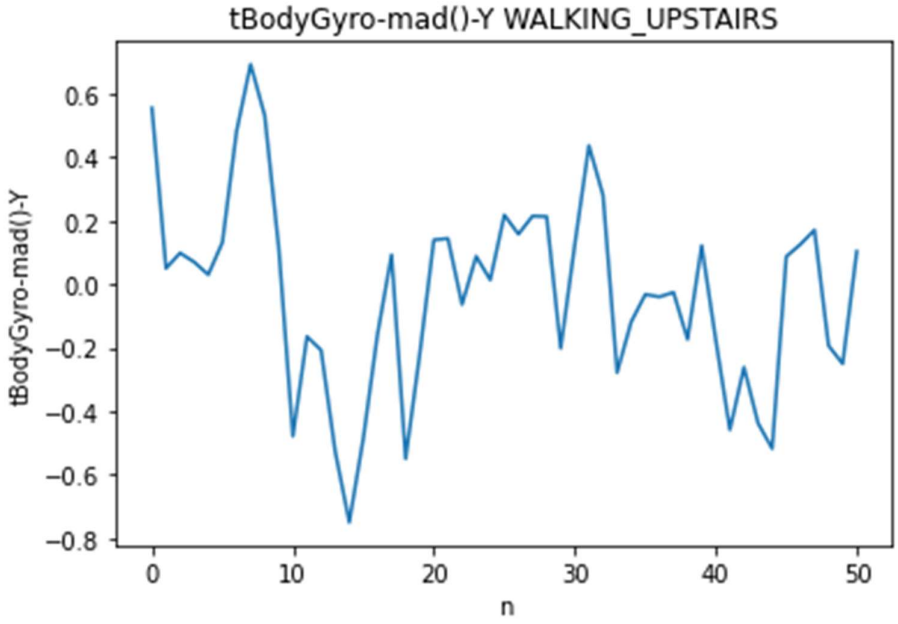
Plots Obtained:

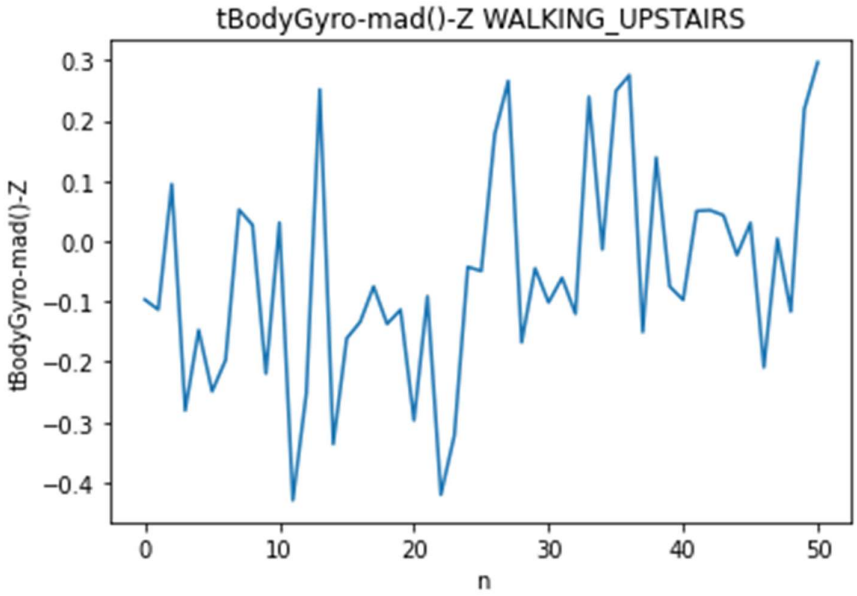


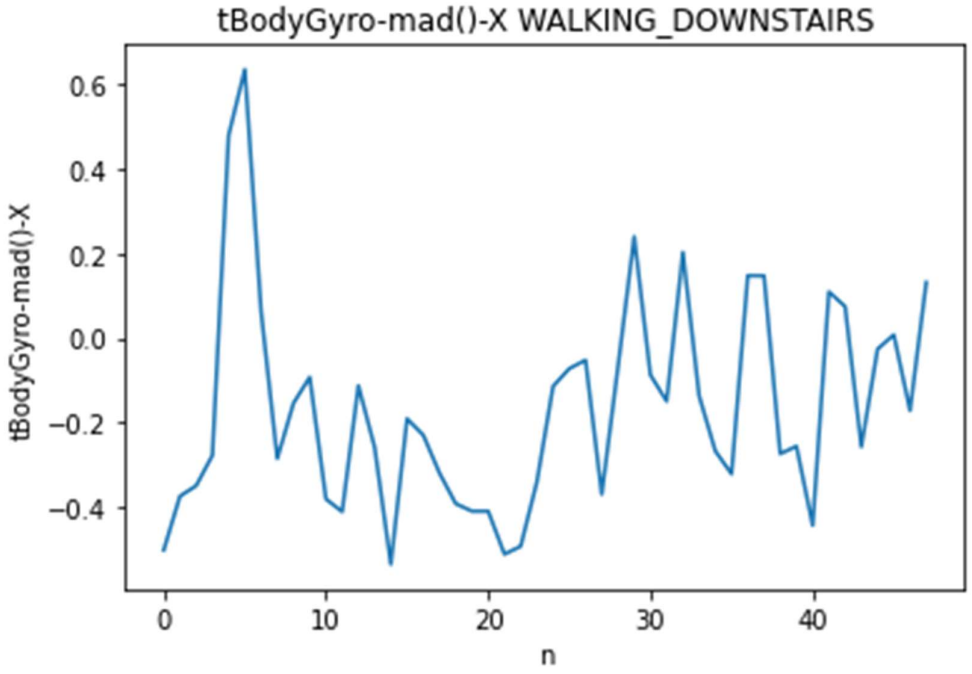


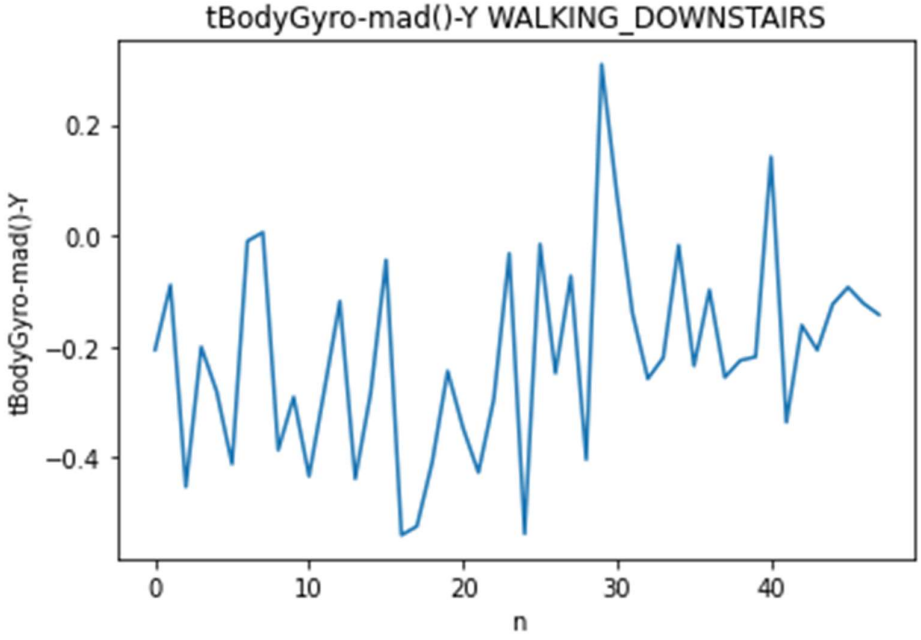

tBodyGyro-mad()-Z WALKING
tBodyGyro-mad()-Z
0.4
0.2
0.0
−0.2
0
10
20
30
40
50
n

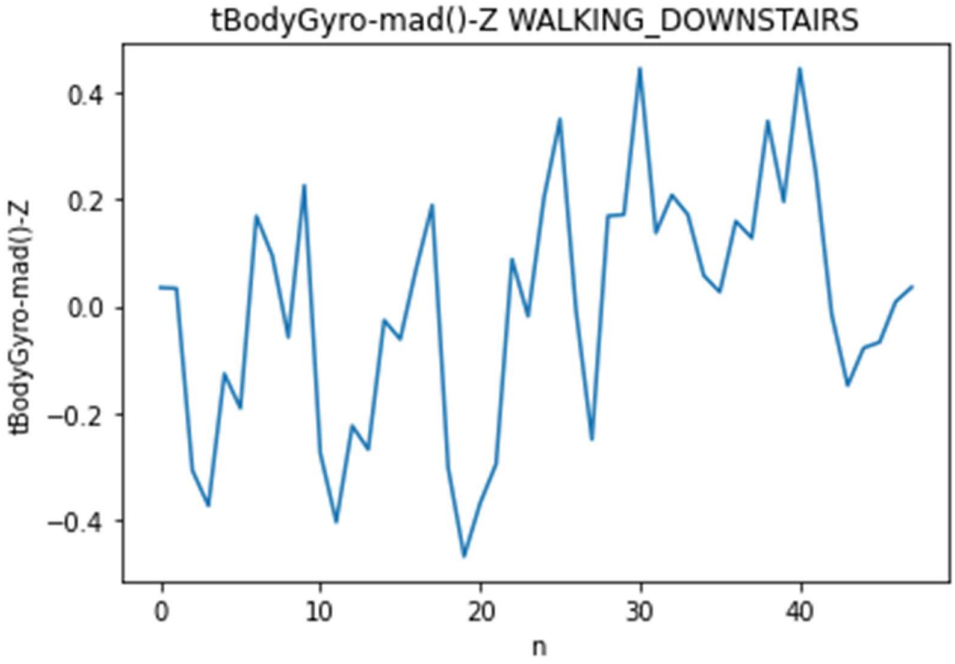

tBodyGyro-mad()-X WALKING_UPSTAIRS
tBodyGyro-mad()-X
−0.2
−0.3
−0.4
−0.5
−0.6
−0.7
0
10
20
30
40
50
n

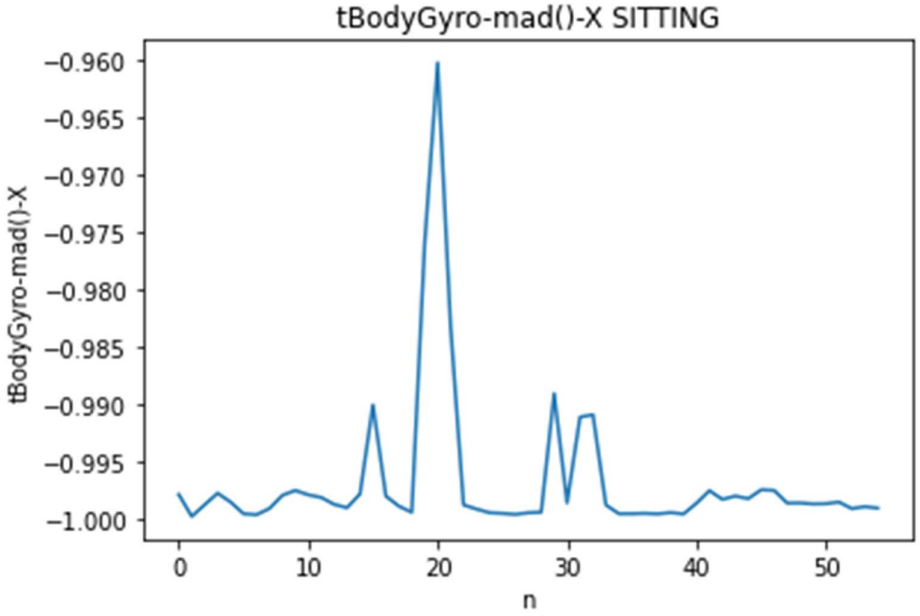

tBodyGyro-mad()-Y WALKING_UPSTAIRS
tBodyGyro-mad()-Y
n
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
10
20
30
40
50


tBodyGyro-mad()-Z WALKING_UPSTAIRS
tBodyGyro-mad()-Z
n
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
−0.4
0
10
20
30
40
50


tBodyGyro-mad()-X WALKING_DOWNSTAIRS
tBodyGyro-mad()-X
0.6
0.4
0.2
0.0
−0.2
−0.4
0
10
20
30
40
n

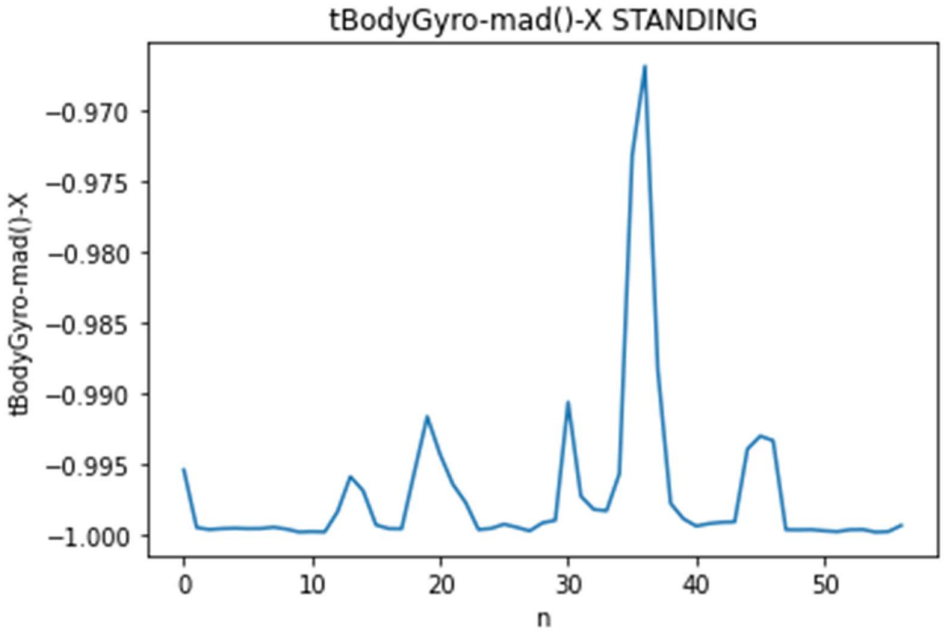

tBodyGyro-mad()-Y WALKING_DOWNSTAIRS
tBodyGyro-mad()-Y
0.2
0.0
−0.2
−0.4
0
10
20
30
40
n

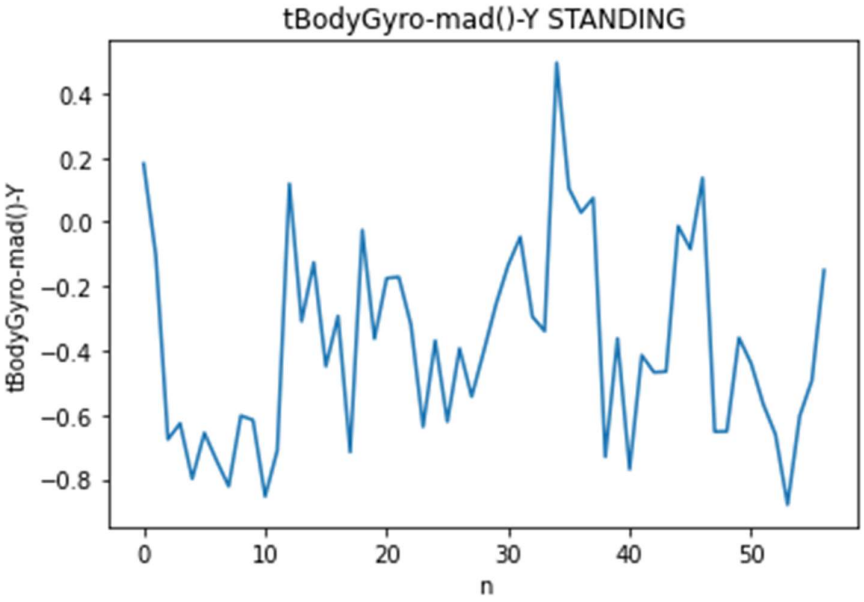

tBodyGyro-mad()-Z WALKING_DOWNSTAIRS
tBodyGyro-mad()-Z
0.4
0.2
0.0
−0.2
−0.4
0
10
20
30
40
n

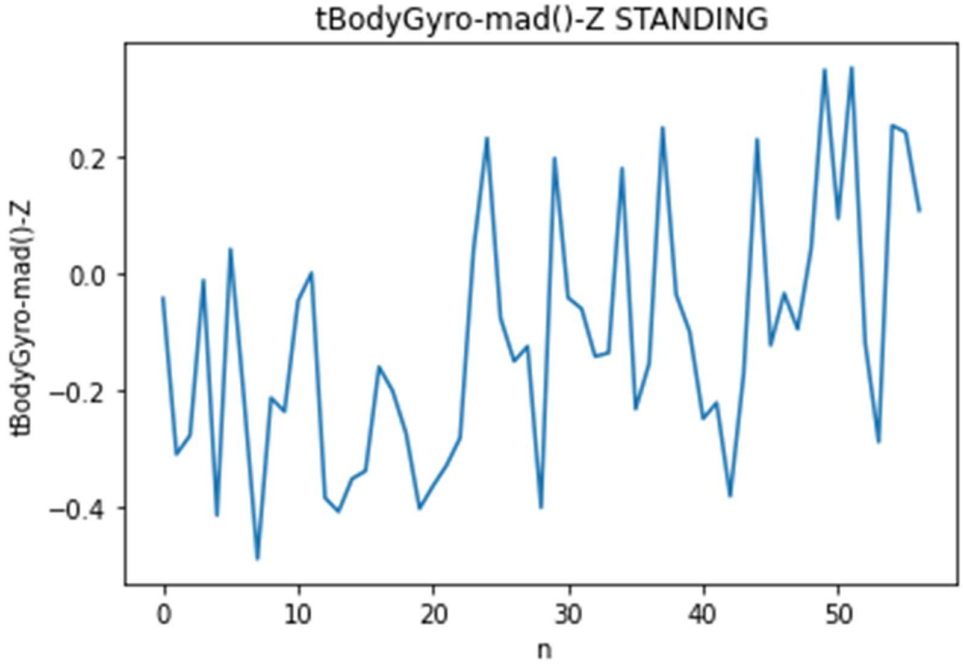

tBodyGyro-mad()-X SITTING
tBodyGyro-mad()-X
−0.960
−0.965
−0.970
−0.975
−0.980
−0.985
−0.990
−0.995
−1.000
0
10
20
30
40
50
n


tBodyGyro-mad()-Y SITTING
tBodyGyro-mad()-Y
n
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
10
20
30
40
50

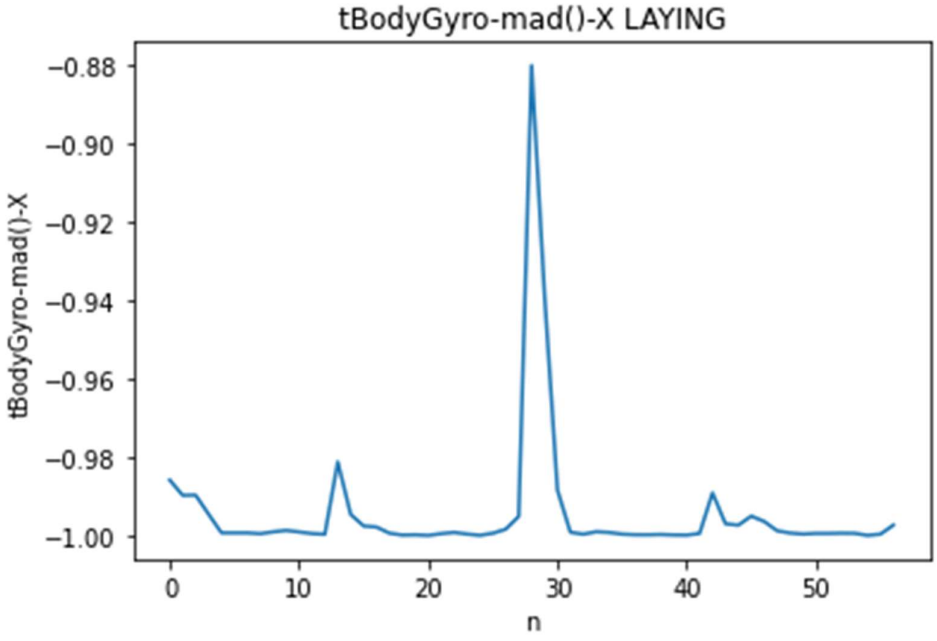

tBodyGyro-mad()-Z SITTING
tBodyGyro-mad()-Z
n
0.4
0.2
0.0
−0.2
−0.4
0
10
20
30
40
50

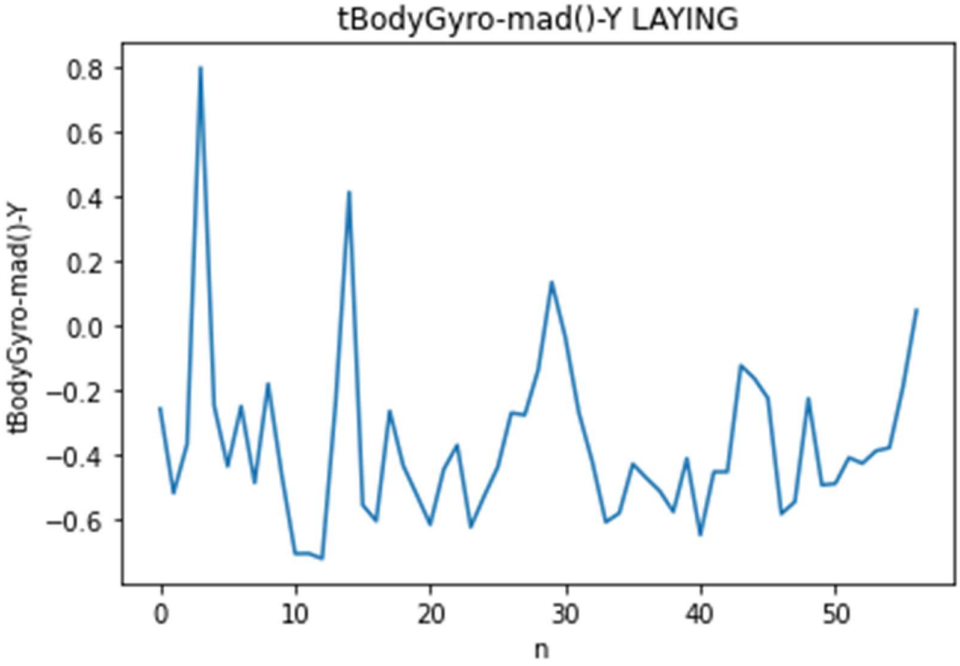

tBodyGyro-mad()-X STANDING
tBodyGyro-mad()-X
−0.970
−0.975
−0.980
−0.985
−0.990
−0.995
−1.000
0
10
20
30
40
50
n


tBodyGyro-mad()-Y STANDING
tBodyGyro-mad()-Y
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
10
20
30
40
50
n

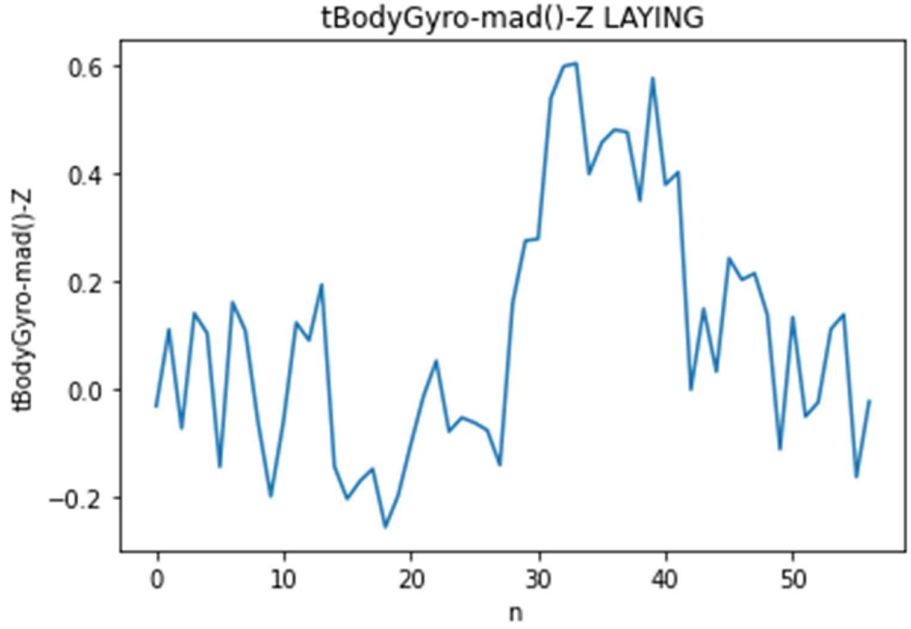

tBodyGyro-mad()-Z STANDING
tBodyGyro-mad()-Z
0.2
0.0
−0.2
−0.4
0
10
20
30
40
50
n

..


tBodyGyro-mad()-X LAYING
tBodyGyro-mad()-X
−0.88
−0.90
−0.92
−0.94
−0.96
−0.98
−1.00
0
10
20
30
40
50
n


tBodyGyro-mad()-Y LAYING
tBodyGyro-mad()-Y
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
0
10
20
30
40
50
n

''


tBodyGyro-mad()-Z LAYING
tBodyGyro-mad()-Z
0.6
0.4
0.2
0.0
−0.2
0
10
20
30
40
50
n

Question 3, 4:

Develop algorithms for activity detection and behaviour change detection. 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.

Show the activity change and behaviour change graphically.

Answer:

We can use a machine learning model for activity change detection and behaviour change detection. As our dataset has already been partitioned into train and test datasets, our work becomes a lot easier and we can use a pre-built model from packages like sklearn for prediction and pattern detection.

For Activity Change Detection, I have tried different classification models but have found that the lightGBM model gives the best results. Using lightGBM model, I have obtained an accuracy score of 93.5 percent. The lightGBM model is a gradient boosted machine learning model and gives accurate predictions pretty fast.

Code:

```
x_train = x_train.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
y_train = x_train["Activity"]
x_train.drop(columns = ["Activity"], inplace = True)
```

+ Code + Markdown

```
model = LGBMClassifier()
model.fit(x_train, y_train)
```

```
LGBMClassifier()
```

+ Code + Markdown

```
x_test = x_test.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
y_test = x_test["Activity"]
x_test.drop(columns = ["Activity"], inplace = True)
```

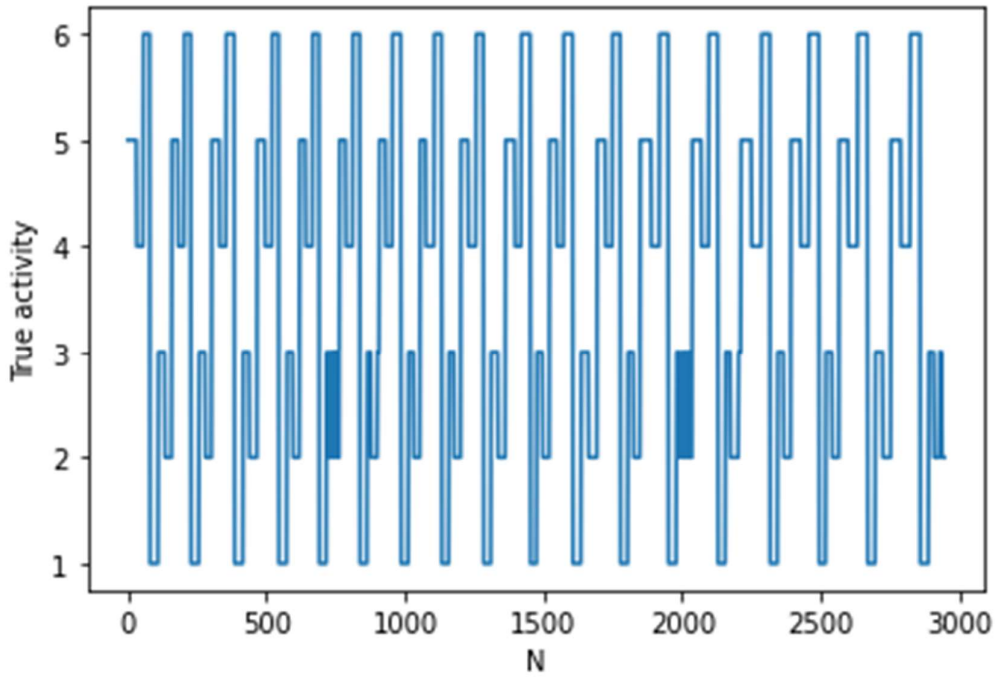
```
predictions = model.predict(x_test)
```

```
print("Accuracy Score: ", accuracy_score(predictions, y_test))
```

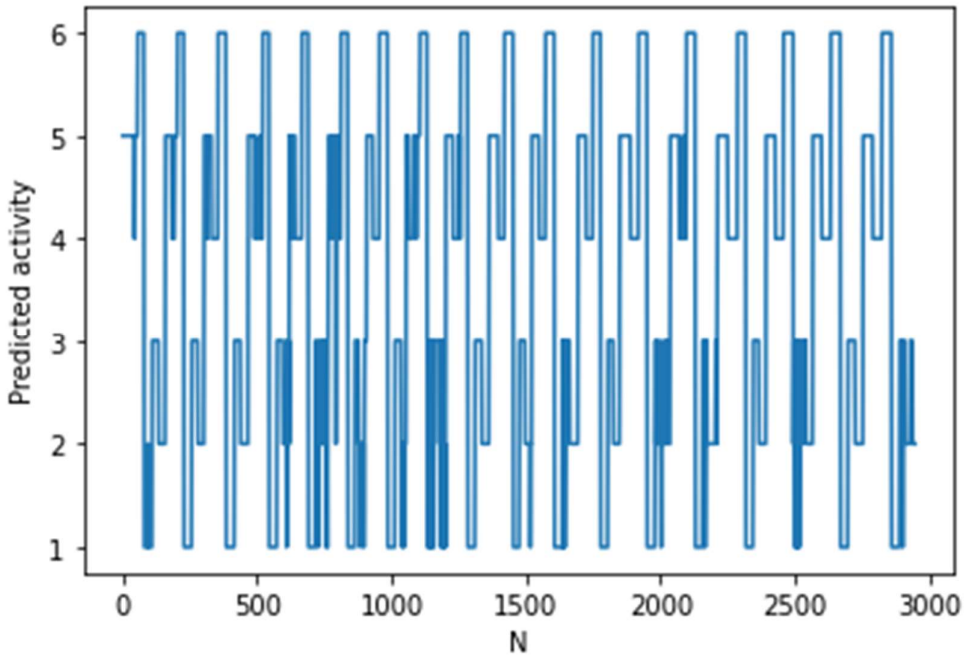
```
Accuracy Score:  0.9358669833729216
```

Plots Obtained:

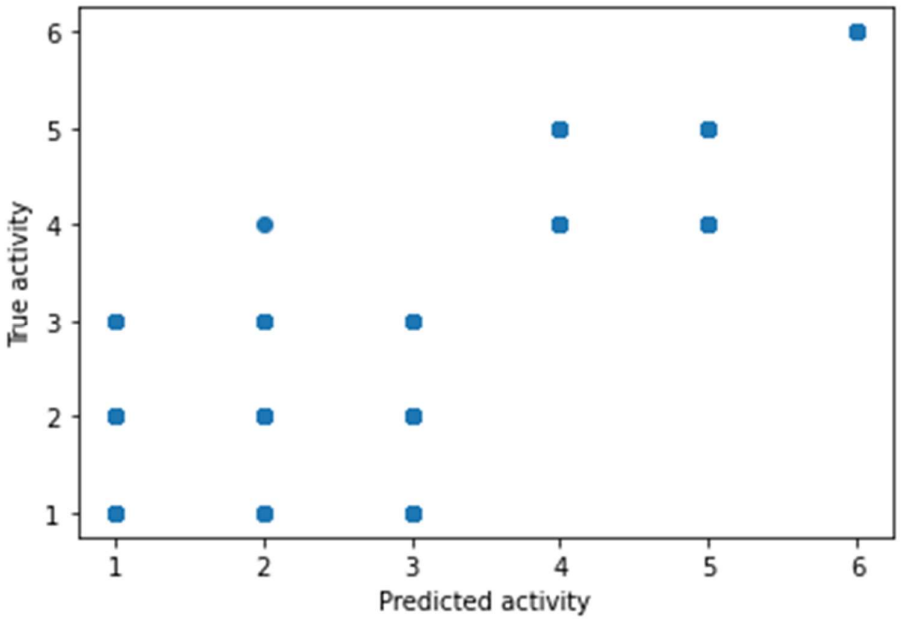
```
plt.plot(list(y_test))
plt.ylabel("True activity")
plt.xlabel("N")
plt.show()
```



```
plt.plot(list(predictions))
plt.ylabel("Predicted activity")
plt.xlabel("N")
plt.show()
```



```
plt.scatter(list(predictions),list(y_test))
plt.ylabel("True activity")
plt.xlabel("Predicted activity")
plt.show()
```



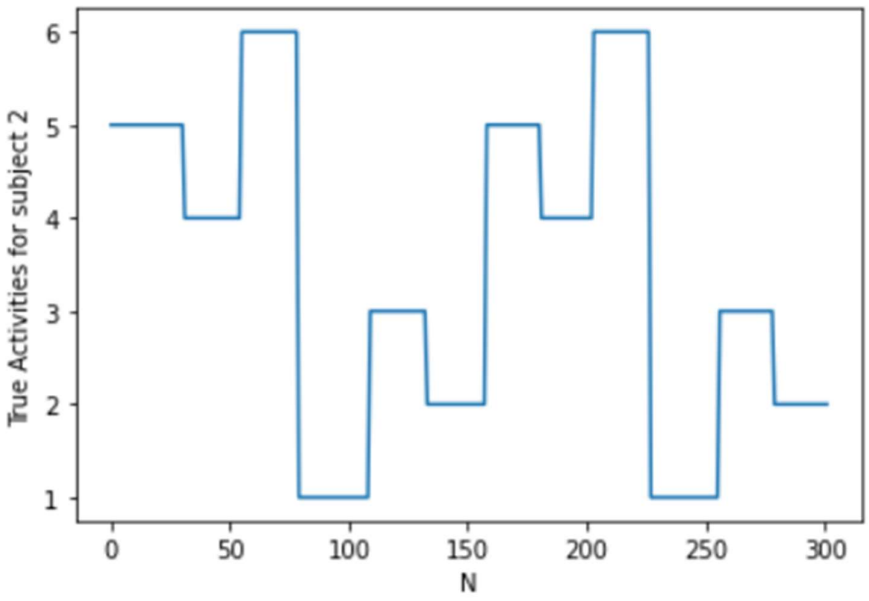
From the above plot we can see that our model was able to predict the value with a high amount of accuracy.

For Behaviour change detection, we can use a python library called Ruptures (https://github.com/deepcharles/ruptures) which can detect change points in the data and also predict the change points.

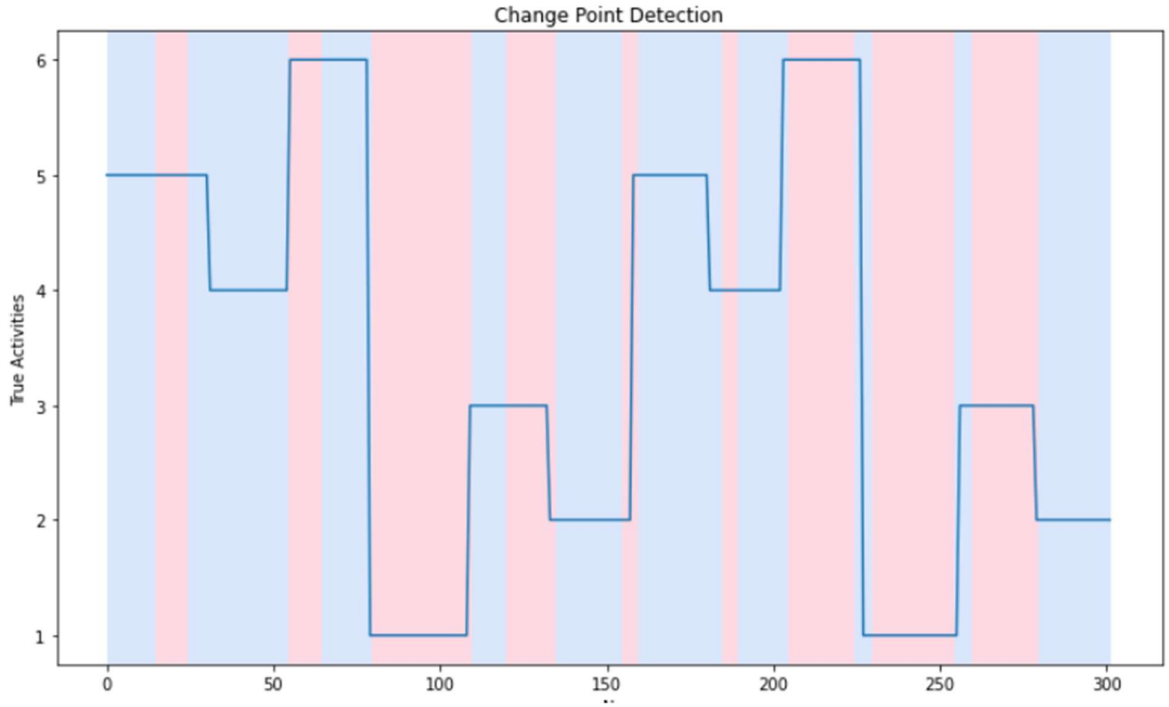
We used the ruptures package with PELT (Pruned Exact Linear Time) method, the change point detection is done for subject 2. The PELt method is exact and gives more accurate results when compared to other algorithms.

Plots obtained:

```
plt.plot(list(y_subject_test))
plt.ylabel('True Activities for subject 2')
plt.xlabel("N")
plt.show()
```



```
rpt.display(y_subject_test, result, figsize=(10, 6))
plt.title('Change Point Detection')
plt.xlabel("N")
plt.ylabel("True Activities")
plt.show()
```



From the above graph, we can observe that the blue part is where the predicted values follow the true values and read parts are where the predicted values deviate from the true values.

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

Answer:

We first divide the dataset into those containing only gyroscopic data and those containing only accelerometer data. Once this is done, we use the ruptures package to do change detection and then compare with other plots. We have taken the data of subject 2 for this question.

Code:

[77]:
```
subject = 2
x_test = x_test[x_test["Subject"] == subject]
x_test = x_test.rename(columns = lambda x:re.sub('[^A-Za-z0-9_]+', '', x))
y_test = x_test["Activity"]
x_test.drop(columns = ["Activity"], inplace = True)

non_accelerometer = []
non_gyroscopic = []

subject_columns = subject_info.columns.tolist()

for column in subject_columns:
    if("Acc" not in column):
        non_accelerometer.append(column)
    if("Gyro" not in column):
        non_gyroscopic.append(column)
```

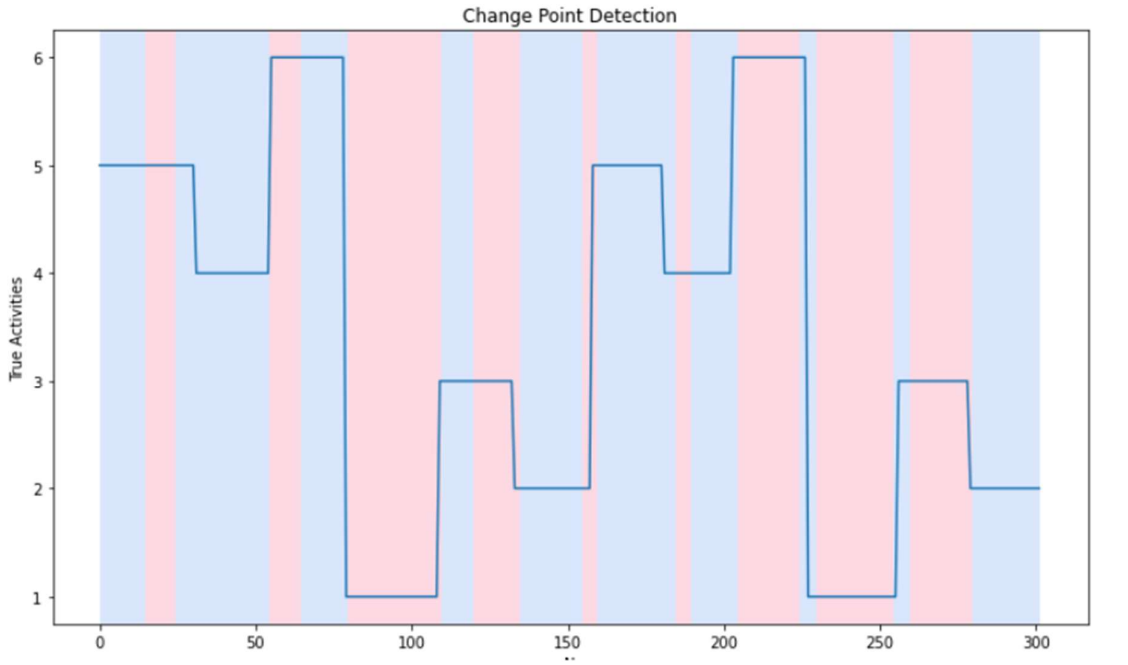
+ Code + Markdown

[78]:
```
accelerometer_data = subject_info.copy()
accelerometer_data = accelerometer_data.drop(columns = non_accelerometer)

gyroscopic_data = subject_info.copy()
gyroscopic_data = gyroscopic_data.drop(columns = non_gyroscopic)
```

Plots:

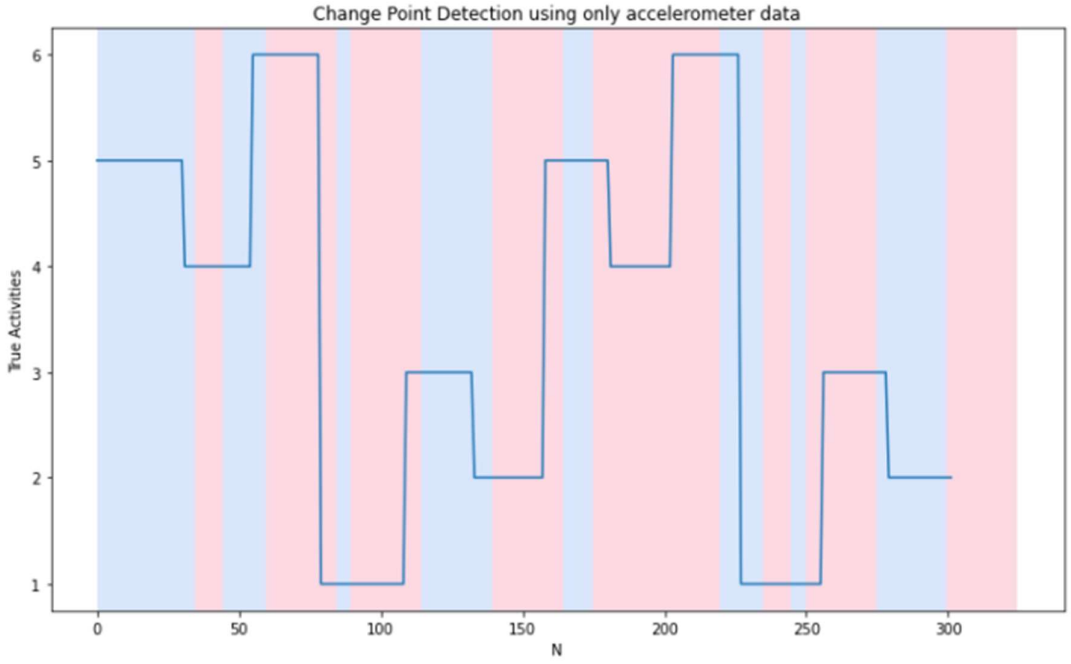
```
rpt.display(y_subject_test, result, figsize=(10, 6))
plt.title('Change Point Detection')
plt.xlabel("N")
plt.ylabel("True Activities")
plt.show()
```



]:

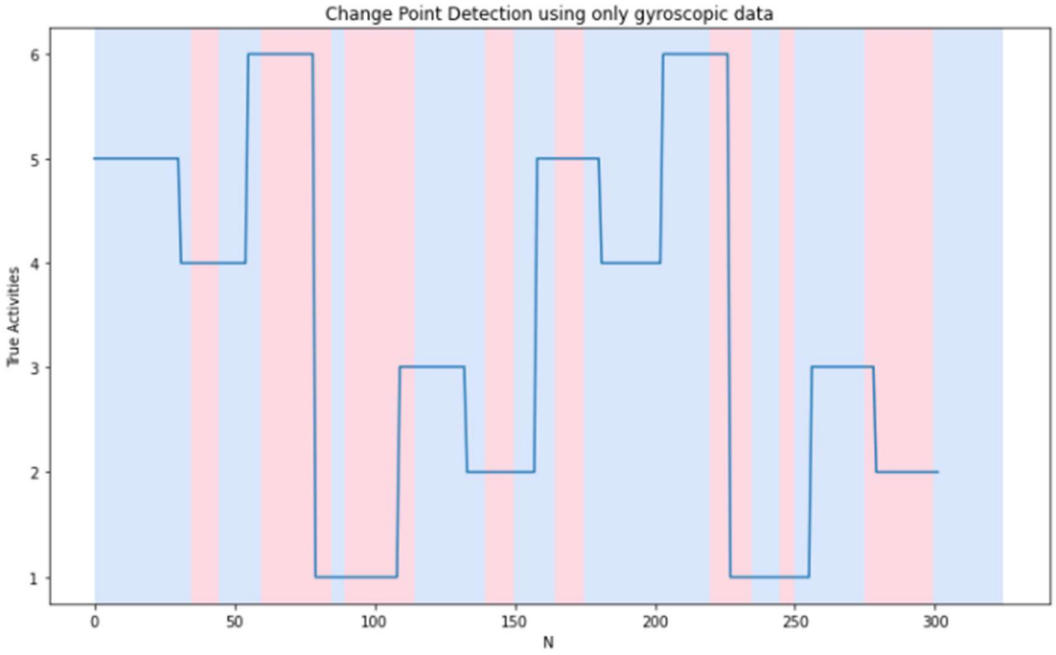
```
algorithm = rpt.Pelt(model="rbf").fit(accelerometer_data)
result = algorithm.predict(pen=0.6)

rpt.display(y_test, result, figsize=(10, 6))
plt.title('Change Point Detection using only accelerometer data')
plt.xlabel("N")
plt.ylabel("True Activities")
plt.show()
```



```
algorithm = rpt.Pelt(model="rbf").fit(gyroscopic_data)
result = algorithm.predict(pen=0.6)

rpt.display(y_test, result, figsize=(10, 6))
plt.title('Change Point Detection using only gyroscopic data')
plt.xlabel("N")
plt.ylabel("True Activities")
plt.show()
```



From the plots, we can observe that the read areas i.e. areas where the predicted values do not match up with the true values are more pronounced in the plots with only gyroscopic data and accelerometer data than the plot with both gyroscopic and accelerometer data.

Thus, we can conclude that we can get better results from change detection when using both accelerometer data and gyroscopic data than when using only one sensor data.

References:

https://centre-borelli.github.io/ruptures-docs/

https://article.sciencepublishinggroup.com/html/10.11648.j.ajtas.20150406.30.html