How to recognise normal human activities using smartphone sensors

Paulo Sousa *Departamento de Informática*
IPV-ESTGV
Viseu, Portugal
estgv14743@alunos.estgv.ipv.pt

Hugo Loureiro *Departamento de Informática*
IPV-ESTGV
Viseu, Portugal
estgv14660@alunos.estgv.ipv.pt

January 31, 2021

Abstract

Through the years, the smartphone technology is spread both in the academic and commercial world.

At this moment, almost everyone has a smartphone in their pocket, that is used not only to call other people, but also to share their location, take pictures, record videos and monitor some of our daily activities. For example, we can compute our position using the sensors integrated in our smartphone that include accelerometers and gyroscopes.

In this paper, we will present the analysis of data extracted from the accelerometer and gyroscope sensors of a smartphone while performing some of our daily activities. Each activity will be performed for a certain amount of time by two or three performers.

These activities will include basic ambulation-related activities like walking, running and climbing stairs.

The data extracted will be associated not only to the activity being performed, but also to the performer of the activity. Which means that the data can be used to activity recognition mode

1 Introduction

In this paper, we will present the analysis of data extracted from the accelerometer and gyroscope sensors of a smartphone while performing some of our daily activities. Each activity will be performed for a certain amount of time by two or three performers.

Over this paper, we will talk a little bit and explain what consists the accelerometer and gyroscope sensors, and the importance of this two to the world as we see it today.

The purpose of this work is to collect data from these two sensors. We will show at least 3 activities that we do or happens all the time in our day to day, we will analyze this extracted data and apply artificial intelligence to model the data to our liking.

At the end we will present the data properly analyzed and structured in some different formats, such as graphics, tables, etc.

2 Related work

In order to do our experimental work, we had to investigate to organize our process.

For that we have analysed [1] which gave us the idea of variety of sensors that we have in our smartphones. This part helped us a lot when to choose the sensors that we wanted to read from in order to get our idea done. This reference also gave us the notion why do we need to record mobile sensors and in which way it can helps

us. The Android OS itself records output of various sensors in mobile phone automatically or on our request to assist us while using many of our applications in our device.

Android sensors can also be recorder by the users to use the recorded output in the implementation of the application that we desire to develop. An example of this kind of applications is to control the robots and machines, to identify metallic or non-metallic objects, to study vibration of a machine part and, most importantly to our case, to control some activity based on mobile orientation.

The reference [1] gave us a notion on how to record sensors data and process it offline or online.

[2] which is supported by [3] also helps us to understand better what is human activity recognition and how to approach this theme.

Like said before, human activity recognition is the problem of predicting what a person is doing based on a trace of their movements using sensors. Sensors are often located on the subject, such as a smartphone or vest, and is always recording accelerometer data in three dimensions (x,y,z).

The major idea is that once the subject's activity is recognized and known, an intelligent computer system can offer assistance.

This is a challenging problem because there is no clear way to relate the sensor data to specific actions in a general way. It is very challenging due to the large volume of sensor data collected and the use of hand-crafted features and heuristics from this data in developing predictive models.

[4] A Public Domain Dataset for Human Activity Recognition Using Smartphones gave us a even better understanding of what human activity recognition really is.

3 Experimental Work

Our idea is to use sensor data coming from a smartphone to recognize human activities. For that we used the accelerometer and gyroscope sensors from a smartphone.

3.1 Extracting the data

To extract the accelerometer and gyroscope's data, we have used an android app that is accessible to everyone in Play Store called '**Physics Toolbox Suite**'. This app gives us the data in .CSV file format from the moment that we start recording to the moment we stop it (.CSV files from this phase of extracting data will be given in the attachments to this file).

- **Accelerometer**

  The linear accelerometer measures acceleration in a straight-line in three different dimensions.

  Linear acceleration changes whenever the mobile device speeds up, slows down or changes direction. When the mobile device is at rest with respect to the surface of the earth, it reads acceleration values of (0,0,0).

  Linear acceleration is derived from the accelerometer, but also uses the gyroscope and the magnetometer to negate the effects of the earth's gravitational field on the sensor.

  The accelerometer contains either a cantilever or comb-like inertial massed attached to springs with one dimension.

  Sensor Collection Rate: 125Hz

- **Gyroscope**

  The gyroscope measures rotational velocity around three axes (x,y,z).

  The gyroscope is often composed of one or a paired set of comb-like inertial masses that are free to move along a plane. A small electrical current causes the frames of the comb-like structures to resonate along a single dimension. When the mobile device rotates, the comb-like structure begins to move out of the

single direction of resonant vibration. This is due to Coriolis effect, often attributed to the Coriolis force (a false force) that results from different frames of reference.

As the mobile device spins, the comb-like mass moves about in the presence of a capacitor plates between the comb-like structures, varying the electrical potential across the plates. The changes in electric potential due to the difference in direction of the vibration compared to a non-rotating system is interpreted as rotational velocity.

Sensor Collection Rate: 500 Hz

3.2 Plots of the raw data from sensors

To plot the raw data extracted from the accelerometer and gyroscope, we used a python library "*matplotlib.pyplot*" and have implemented a small script to generate the plot for all activities. The activities are name A, B, C, D and E (the description of the activities is in the activitiesLayout.txt file).

The script to generate the plot of the raw data and respective plots of each sensor are shown below:

```
import pandas as pd #library pandas
import matplotlib.pyplot as plt


activities = ['A', 'B', 'C', 'D', 'E']

for i in activities:

    data = pd.read_excel(str(i)+'_gyro.xlsx')
    pd.DataFrame(data)

    # by analysing the min and max values of each column,
    #we can determine the time of record, max and of ax, ay, az and at
    data.describe()

    # In here we will plot the data coming from the gyroscope sensor data
    #Get data from each attribute

    #time iterations
    time = data.iloc[:,0].values
    #wx  (rad/s)
    wx = data.iloc[:,1].values
    #wy  (rad/s)
    wy = data.iloc[:,2].values
    #wz  (rad/s)
    wz = data.iloc[:,3].values

    #plot the data
    plt.plot(wx, 'r', wy, 'g', wz, 'b')
    plt.title('Activity_' + str(i) + '_Gyroscope')
    plt.xlabel('Time(ms)')
    plt.ylabel('Angular_velocity_(rad/s)')
    plt.show()
#-------------------------------------------------------------------------------------------------------------------------#
for i in activities:

    data = pd.read_excel(str(i)+'_acc.xlsx')
    pd.DataFrame(data)

    # by analysing the min and max values of each column,
    # we can determine the time of record, max and of ax, ay, az and at
    data.describe()

    # In here we will plot the data coming from the gyroscope sensor data
    #Get data from each attribute

    #time iterations
    time = data.iloc[:,0].values
    #wx  (rad/s)
    wx = data.iloc[:,1].values
    #wy  (rad/s)
    wy = data.iloc[:,2].values
    #wz  (rad/s)
    wz = data.iloc[:,3].values

    #plot the data
    plt.plot(wx, 'r', wy, 'g', wz, 'b')
    plt.title('Activity_' + str(i) + '_Accelerometer')
    plt.xlabel('Time(ms)')
    plt.ylabel('Acceleration_(m/s^2)')
    plt.show()
```

3.3 Applying Machine Learning

In order to apply machine learning to our extracted data, we had to prepare our files from "***rawData***" folder to start our analysis.

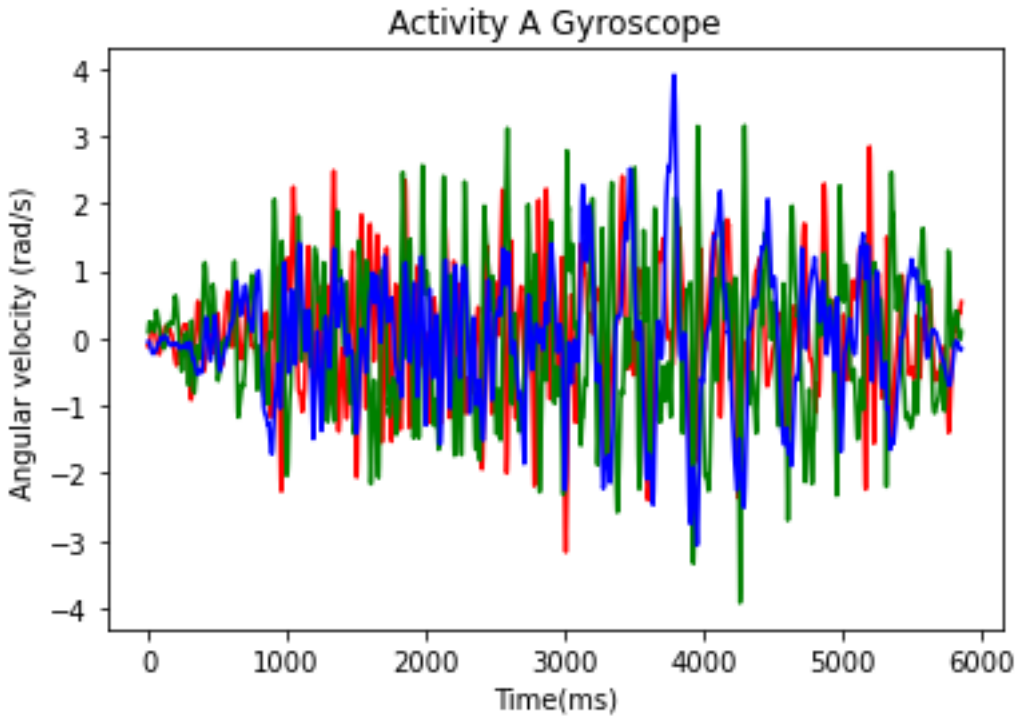

Figure 1: Plot of Activity A (Gyroscope)

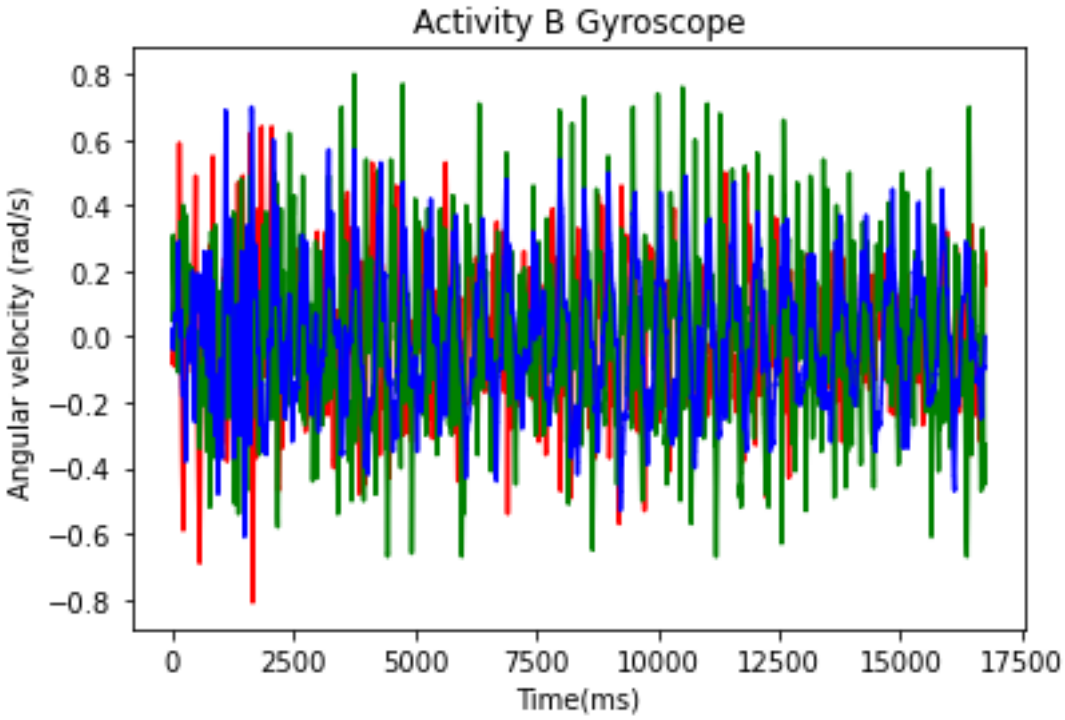

Figure 2: Plot of Activity B (Gyroscope)

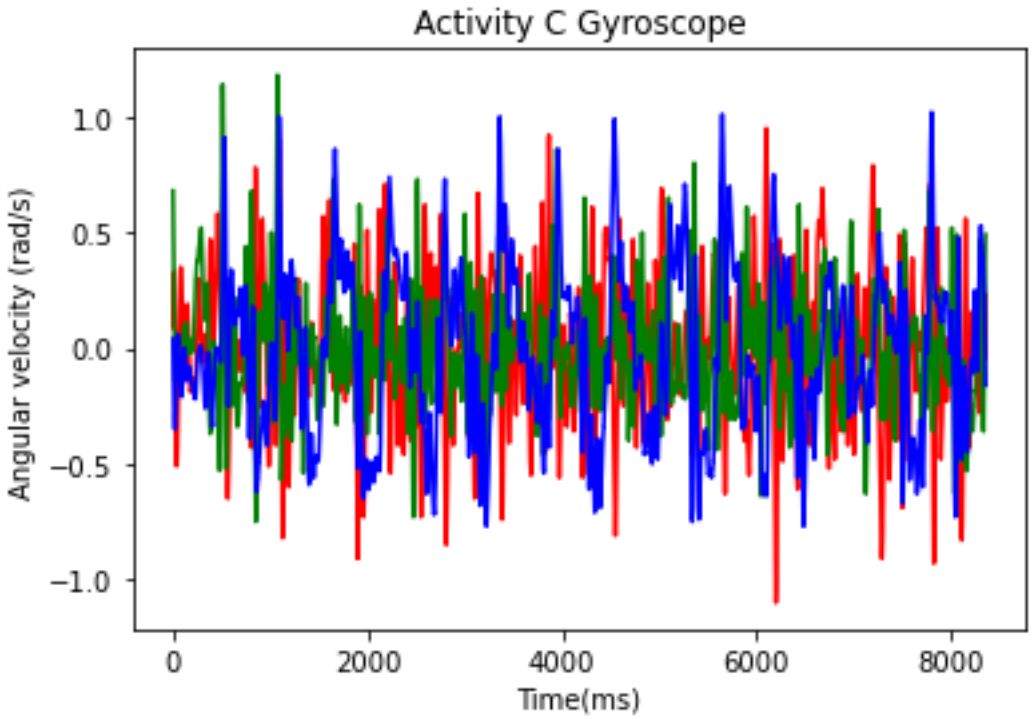

Figure 3: Plot of Activity C (Gyroscope)

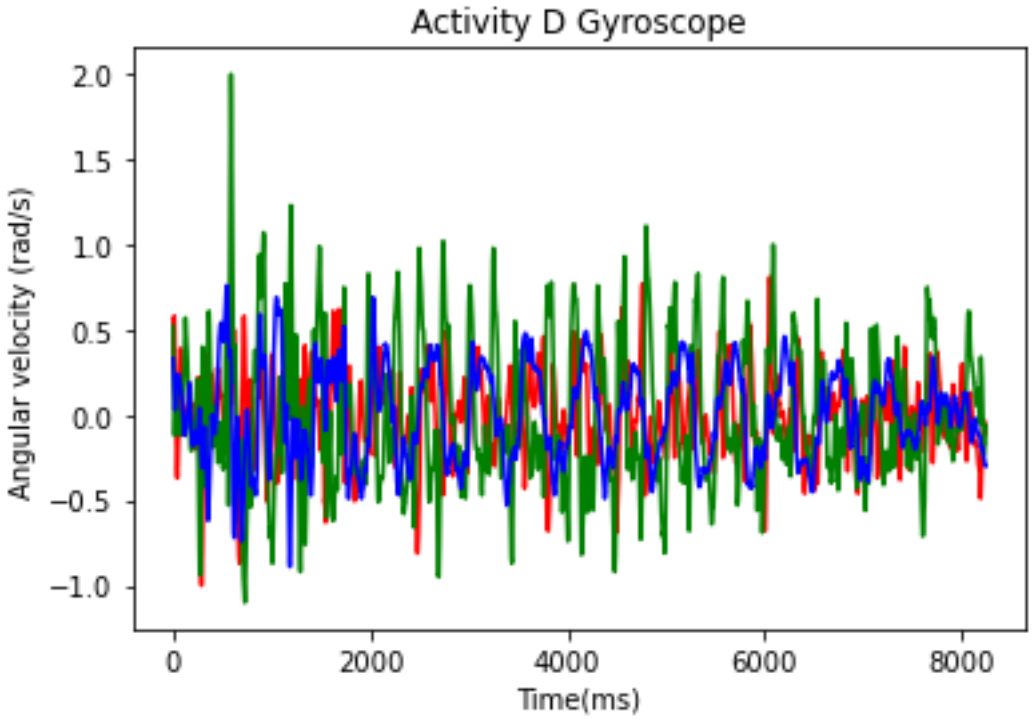

Figure 4: Plot of Activity D (Gyroscope)

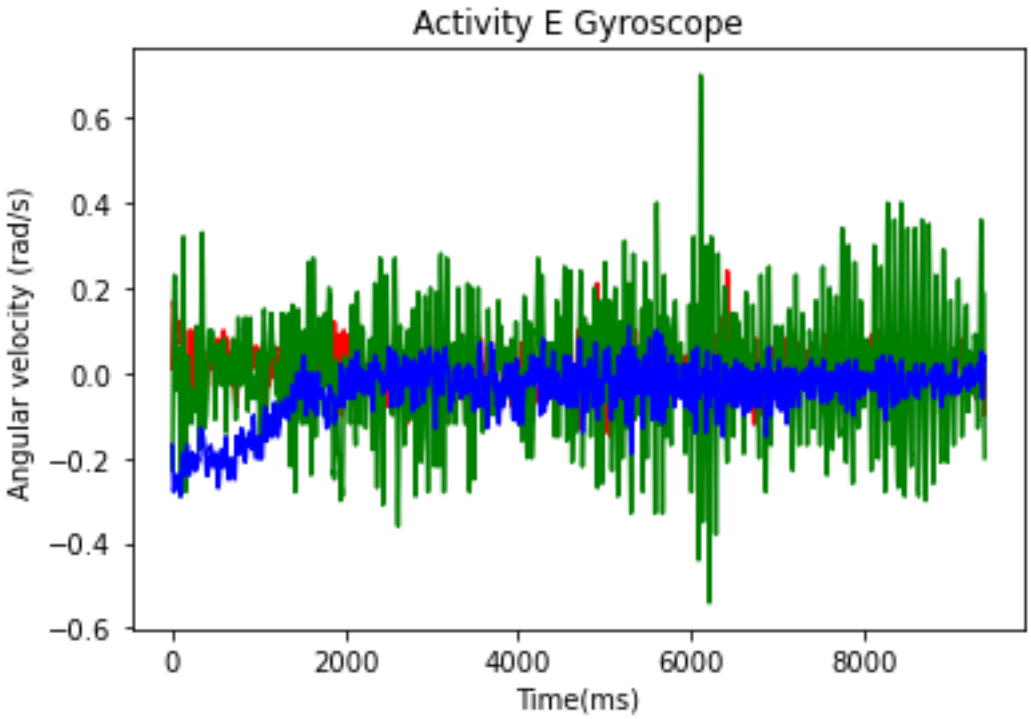

Figure 5: Plot of Activity E (Gyroscope)

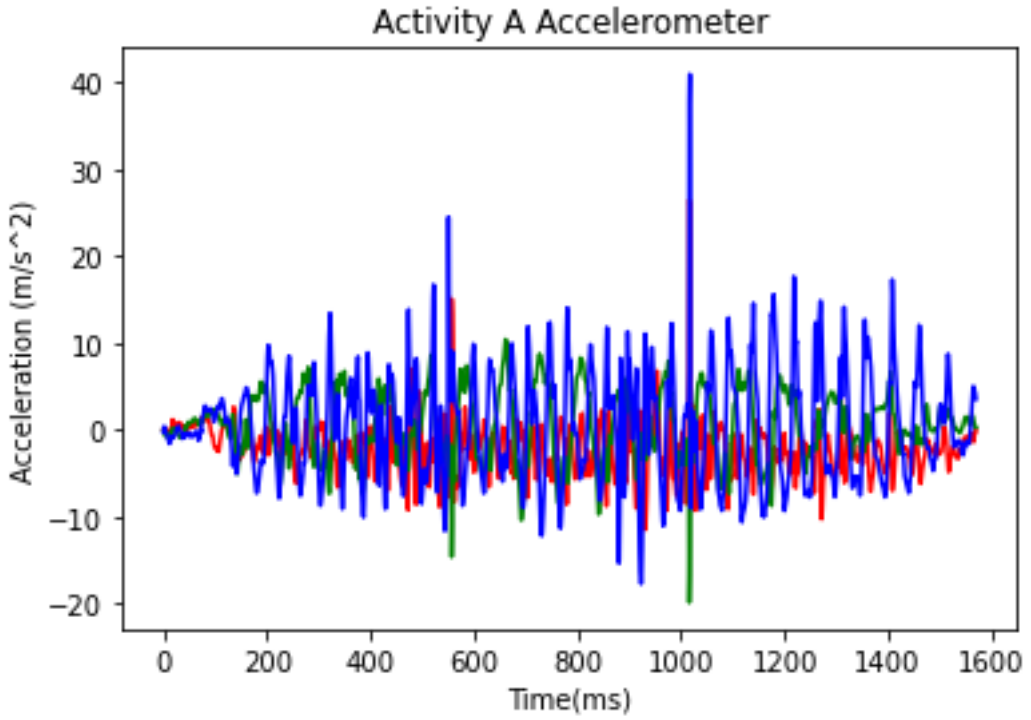

Figure 6: Plot of Activity A (Accelerometer)

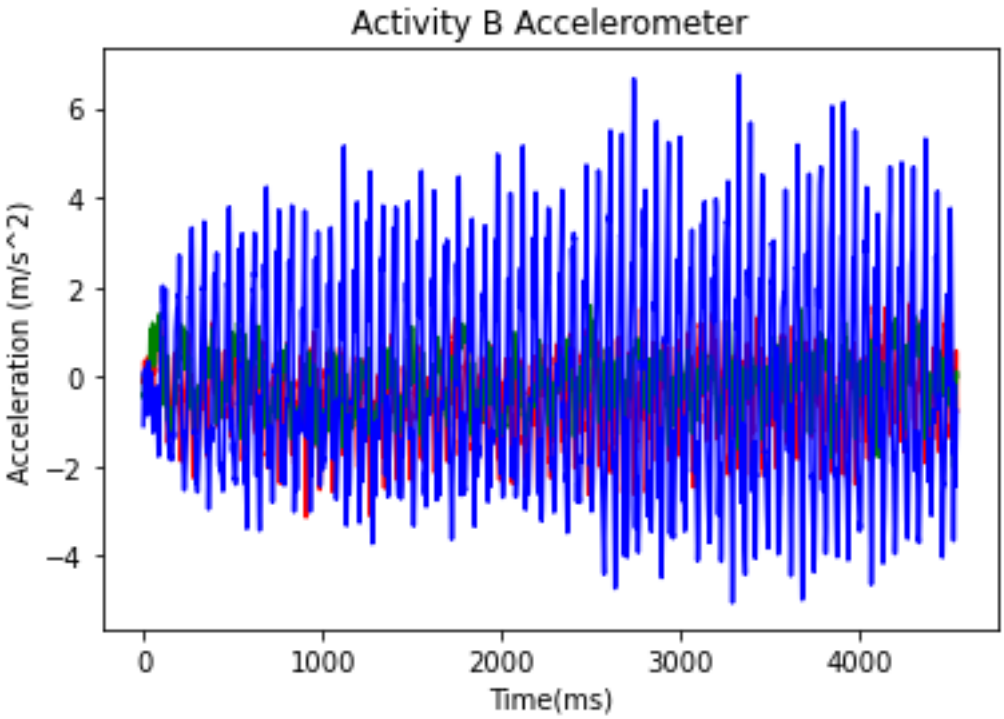

Figure 7: Plot of Activity B (Accelerometer)

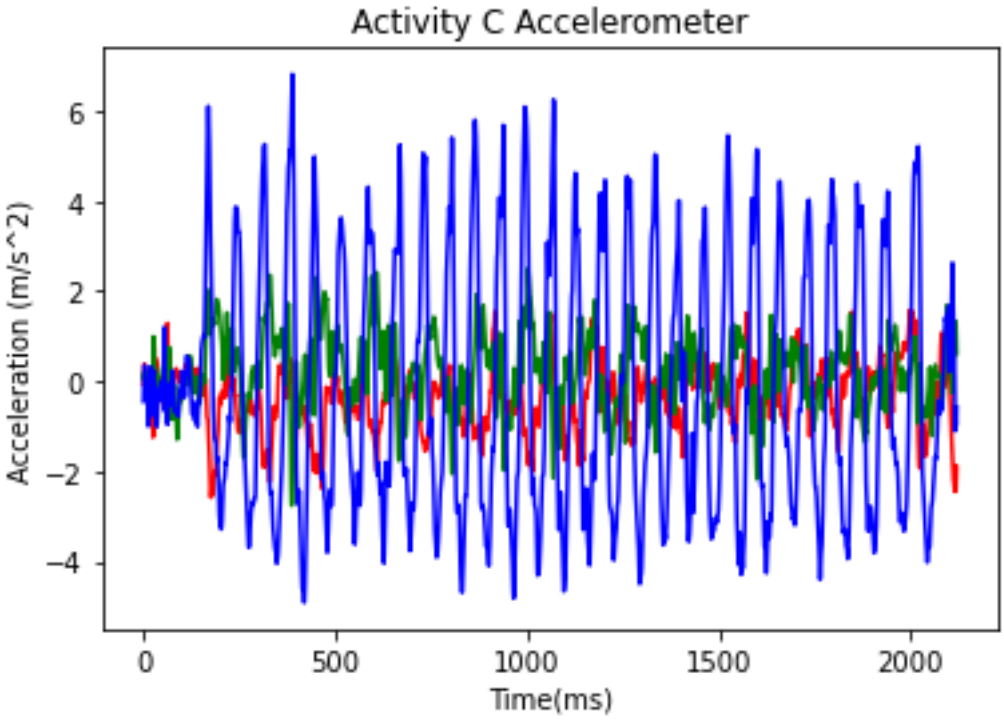

Figure 8: Plot of Activity C (Accelerometer)

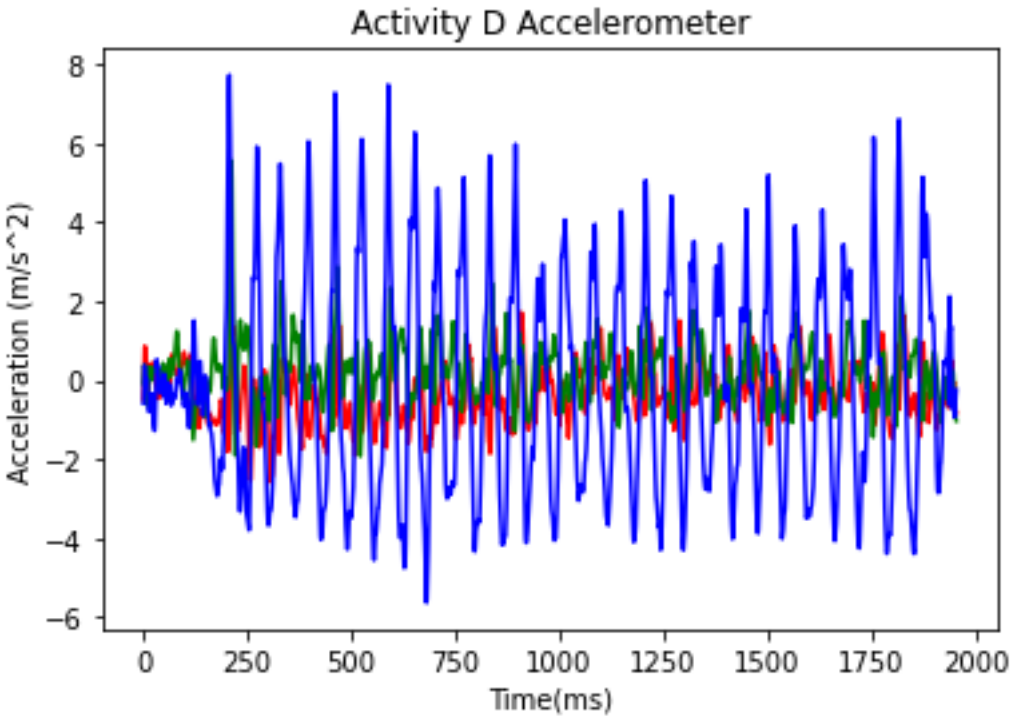

Figure 9: Plot of Activity D (Accelerometer)

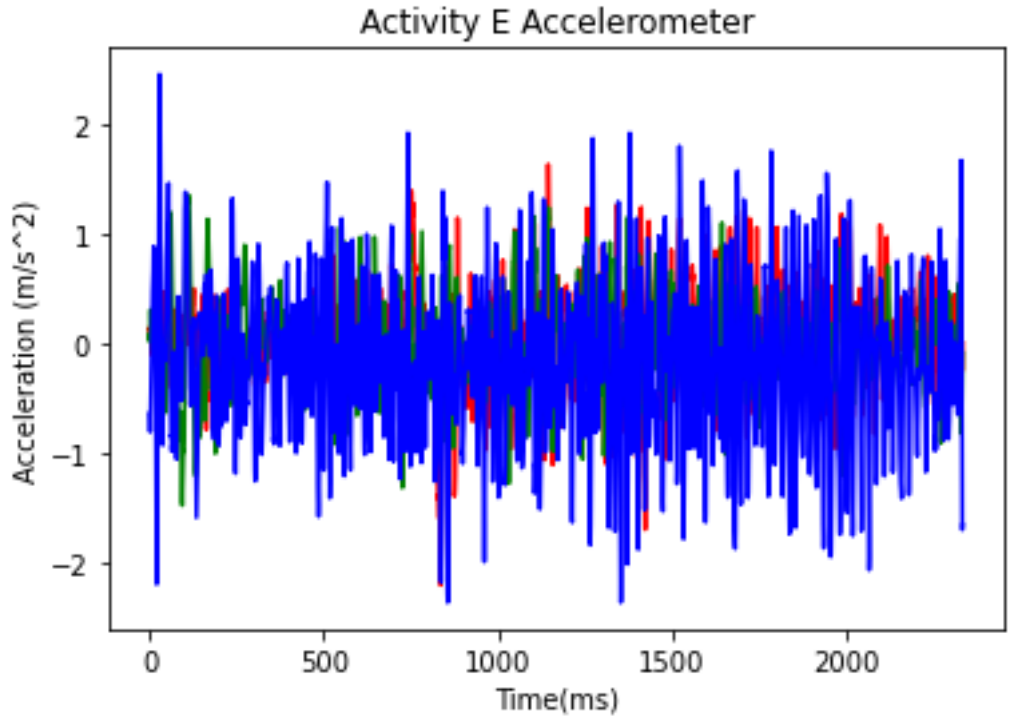

Figure 10: Plot of Activity E (Accelerometer)

For each file (each activity for each sensor) we have deleted the 'Time' attribute (it is not relevant to the analysis that we needed to do), added a column for the corresponding activity and renamed the files in the following way: **data_X_Y**, where X = [A...E] and Y = ['acc','gyro']. These files can be found the in folder named "***Data Preparation***".

After this preparation of our data, we combined all the data from accelerometer in one file (***data_acc.xlxs***) and the data from the gyroscope in another file (***data_gyro.xlxs***). In order to shuffle all the data in each file, we used the function *RAND()* from excel to obtain a more realistic data set to apply machine learning algorithms. These files can be found in folder named "***Data Set***".

In order to apply various algorithms of machine learning to our data sets, we have developed a script in python that reads the dataset, splits the data (descriptive and target), trains the test split, standardizes the data and call the various machine algorithms that we applied, depending on the sensor we want to evaluate (accelerometer or gyroscope). The script is named ***appliedML.py*** and is shown below:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

#-------------------------------------------------------------------------------#
def NaivesBayes(descriptive_train, descriptive_test, target_train, target_test):
    if(i == 'acc'):
        # call Naive Bayes Algorithm
        classifierAcc = GaussianNB()
        classifierAcc.fit(descriptive_train, target_train)

        predictionNBAcc = classifierAcc.predict(descriptive_test)

        # accuracy and Matrix
        accuracyNBAcc = accuracy_score(target_test, predictionNBAcc)
        matrixNBAcc = confusion_matrix(target_test, predictionNBAcc)

        print('\n\n\n\n')
        print('.....................................................................\n')
        print('Accelerometer: Accuracy using NaivesBayes algorithm is {}\n'.format(float(accuracyNBAcc)))

        return accuracyNBAcc, matrixNBAcc

    else:
        # call Naive Bayes Algorithm
        classifierGyro = GaussianNB()
        classifierGyro.fit(descriptive_train, target_train)

        predictionNBGyro = classifierGyro.predict(descriptive_test)

        # accuracy and Matrix
        accuracyNBGyro = accuracy_score(target_test, predictionNBGyro)
        matrixNBGyro = confusion_matrix(target_test, predictionNBGyro)

        print('Gyroscope: Accuracy using NaivesBayes algorithm is {}\n'.format(float(accuracyNBGyro)))

        return accuracyNBGyro, matrixNBGyro
#-------------------------------------------------------------------------------#
def DecisionTree(descriptive_train, descriptive_test, target_train, target_test):
    if(i == 'acc'):
        # call Decision tree Algorithm
        classifierDTAcc = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
        classifierDTAcc.fit(descriptive_train, target_train) # modleo criado

        predictionDTAcc = classifierDTAcc.predict(descriptive_test)

        # accuracy and Matrix
        accuracyDTAcc = accuracy_score(target_test, predictionDTAcc)
        matrixDTAcc = confusion_matrix(target_test, predictionDTAcc)

        print('Accelerometer: Accuracy using Decision Tree algorithm is {}\n'.format(float(accuracyDTAcc)))

        return accuracyDTAcc, matrixDTAcc

    else:
        # call Decision tree Algorithm
        classifierDTGyro = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
        classifierDTGyro.fit(descriptive_train, target_train) # modleo criado

        predictionDTGyro= classifierDTGyro.predict(descriptive_test)

        # accuracy and Matrix
        accuracyDTGyro = accuracy_score(target_test, predictionDTGyro)
        matrixDTGyro = confusion_matrix(target_test, predictionDTGyro)

        print('Gyroscope: Accuracy using Decision Tree algorithm is {}\n'.format(float(accuracyDTGyro)))

        return accuracyDTGyro, matrixDTGyro
#-------------------------------------------------------------------------------#
def RandomForest(descriptive_train, descriptive_test, target_train, target_test):
    if(i == 'acc'):
        # call Random Forest Algorithm
```

```
        classifierRFAcc = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', random_state = 0)
        classifierRFAcc.fit(descriptive_train, target_train) # modleo criado

        predictionRFAcc = classifierRFAcc.predict(descriptive_test)

        # accuracy and Matrix
        accuracyRFAcc = accuracy_score(target_test, predictionRFAcc)
        matrixRFAcc = confusion_matrix(target_test, predictionRFAcc)

        print('Accelerometer: Accuracy using Random Forest algorithm is {} \n'.format(float(accuracyRFAcc)))

        return accuracyRFAcc, matrixRFAcc

    else:
        # call Random Forest Algorithm
        classifierRFGyro = RandomForestClassifier(n_estimators = 20, criterion = 'entropy', random_state = 0)
        classifierRFGyro.fit(descriptive_train, target_train) # modleo criado

        predictionRFGyro = classifierRFGyro.predict(descriptive_test)

        # accuracy and Matrix
        accuracyRFGyro = accuracy_score(target_test, predictionRFGyro)
        matrixRFGyro = confusion_matrix(target_test, predictionRFGyro)

        print('Gyroscope: Accuracy using Random Forest algorithm is {} \n'.format(float(accuracyRFGyro)))

        return accuracyRFGyro, matrixRFGyro
#-------------------------------------------------------------------------------------------------------------#
def kNN(descriptive_train, descriptive_test, target_train, target_test):
    if(i == 'acc'):
        # call kNN Algorithm
        classifierKnnAcc = KNeighborsClassifier(n_neighbors = 9, metric = 'minkowski', p = 2)
        classifierKnnAcc.fit(descriptive_train,target_train)

        predictionKnnAcc = classifierKnnAcc.predict(descriptive_test)

        # accuracy and Matrix
        accuracyKnnAcc = accuracy_score(target_test, predictionKnnAcc)
        matrixKnnAcc = confusion_matrix(target_test, predictionKnnAcc)

        print('Accelerometer: Accuracy using kNN algorithm is {} \n'.format(float(accuracyKnnAcc)))

        return accuracyKnnAcc, matrixKnnAcc

    else:
        # call kNN Algorithm
        classifierKnnGyro = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)
        classifierKnnGyro.fit(descriptive_train,target_train)

        predictionKnnGyro = classifierKnnGyro.predict(descriptive_test)

        # accuracy and Matrix
        accuracyKnnGyro = accuracy_score(target_test, predictionKnnGyro)
        matrixKnnGyro = confusion_matrix(target_test, predictionKnnGyro)

        print('Gyroscope: Accuracy using kNN algorithm is {} \n'.format(float(accuracyKnnGyro)))
        print('.....................................................\n')

        return accuracyKnnGyro, matrixKnnGyro
#-------------------------------------------------------------------------------------------------------------#
sensors = ['acc', 'gyro']

for i in sensors:

    data = pd.read_excel('data_' + str(i) + '.xlsx')
    # to observe our dataset
    pd.DataFrame(data)

    #data.describe()

    if(i == 'acc'):
        # splitting the data: descriptive and Target
        descriptiveAcc = data.iloc[:,0:4].values
        targetAcc = data.iloc[:,-1]
    else:
        descriptiveGyro = data.iloc[:,0:3].values
        targetGyro = data.iloc[:,-1]


    if(i == 'acc'):
        # train test split
        descriptive_train_acc, descriptive_test_acc, target_train_acc, target_test_acc =
        train_test_split(descriptiveAcc, targetAcc, test_size = 0.3, random_state = 0)

        # Data Standarzization
        standard_scaler_acc = StandardScaler()
        descriptive_train_acc[:,:] = standard_scaler_acc.fit_transform(descriptive_train_acc[:,:])
        descriptive_test_acc[:,:] = standard_scaler_acc.fit_transform(descriptive_test_acc[:,:])

        # Call algorithms
        NaivesBayes(descriptive_train_acc,descriptive_test_acc,target_train_acc,target_test_acc)
        DecisionTree(descriptive_train_acc,descriptive_test_acc,target_train_acc,target_test_acc)
        RandomForest(descriptive_train_acc,descriptive_test_acc,target_train_acc,target_test_acc)
        kNN(descriptive_train_acc,descriptive_test_acc,target_train_acc,target_test_acc)
    else:
        # train test split
        descriptive_train_gyro, descriptive_test_gyro, target_train_gyro, target_test_gyro =
        train_test_split(descriptiveGyro, targetGyro, test_size = 0.3, random_state = 0)

        # Data Standarzization
        standard_scaler_gyro = StandardScaler()
        descriptive_train_gyro[:,:] = standard_scaler_acc.fit_transform(descriptive_train_gyro[:,:])
        descriptive_test_gyro[:,:] = standard_scaler_acc.fit_transform(descriptive_test_gyro[:,:])
```

```
# Call algorithms
NaivesBayes(descriptive_train_gyro, descriptive_test_gyro, target_train_gyro, target_test_gyro)
DecisionTree(descriptive_train_gyro, descriptive_test_gyro, target_train_gyro, target_test_gyro)
RandomForest(descriptive_train_gyro, descriptive_test_gyro, target_train_gyro, target_test_gyro)
kNN(descriptive_train_gyro, descriptive_test_gyro, target_train_gyro, target_test_gyro)
```

To test the prediction Model, we use 30% of our dataset and 70% to train.

The algorithms that we have chosen to use to analyse our dataset are: Naives Bayes, Decision Tree, Random Forest and kNN. For each algorithm, we have implemented a function that return the accuracy values and confusion matrix. But the values that we really want are the accuracy values. These values will be the ones that will helps us analyse our results.

3.4 Results and Conclusions

The results coming from ***appliedML.py*** are the following:

Table 1: Accuracy values

ML Algorithms	**Sensors**	
	Accelerometer	***Gyroscope***
Naives Bayes	0.5721128259712613	0.6137595171136566
Decision Tree	0.510111761575306	0.7056725426984019
Random Forest	0.584353379457158	0.7345496947664449
kNN	0.6125598722724853	0.7551958296179436

The conclusions that we take of our experimental work, based on the accuracy levels of the machine learning algorithms that we applied, is that the kNN algorithm gives us the best results, i.e., since the values of accuracy are higher than the other algorithms, it means that using this algorithm our prediction is more correct than using the others. Using the kNN algorithm, the prediction in the **accelerometer** is ***61.26%*** correct and for the **gyroscope** is ***75.52%*** correct.

References

[1] https://www.youtube.com/watch?v=AZUIcXcDpcMt=980s

[2] https://machinelearningmastery.com/how-to-model-human-activity-from-smartphone-data/

[3] https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

[4] Davide Anguita, Alessandro Ghio, Luca Onet, Xavier Parra, Jorge L. Reyes-Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones"