Data Science Research analysis for Physical Activity Monitoring (PAM) dataset

Dataset Overview:

The physical activity monitoring(PAM) dataset comprise of 18 different physical activities performed by 9 subjects mainly employees or students at DFKI(1 female, 8 males) wearing 3 inertial measurement units (IMU) and a heart rate monitor.Detailed data on each subject was stored in individual text files. There are 54 attributes in each file, including timestamp, activity ID, heart rate, and IMU sensory data(ankle,chest and hand).

Introduction:

The importance of physical activity for a healthy life and reducing the likelihood of developing certain diseases is widely recognized. There are many different physical activities with varying effort requirements and different health benefits.The purpose of this project is to develop hardware and/or software capable of determining the amount (by using start and end times and heart rates) and type of physical activity an individual engages in and the actionable observations we can draw from this data.Below are the various steps invloved to complete the analysis using the PAM dataset

a) Data Import - Importing all the necessary libraries and files for the physical activity monitoring dataset analysis

b) Data Cleaning - Data collection, cleaning, and preprocessing are crucial for effective data analysis. This step would involve handling missing values, normalizing data, and parsing time data. Properly preparing the data helps to ensure accurate and meaningful results from the analysis

c) Exploratory Data Analysis - After addressing the missing values and normalizing the data. I would be splitting the dataset into train and test. We will be analyzing and understanding the data that focuses on discovering patterns and correlations in the data.

d) Hypothesis Testing - Per the analysis conducted, we will be performing z test between heartrate and the activities with highest intensity.

e) Modelling - Finally, we will be performing the polynomial regression and random forest algorithm

Data Import:

Firstly, import all the necessary libraries for the PAM dataset analyis. In order to load all the files and create the dataframe, a list of the file names must be created. Additionally, a dictionary should be created to store the names and numbers of each activity, so that the activity being analyzed at each stage can be understood.The MET_values dictionary maps activity IDs to values representing the metabolic equivalent of the activity. Lists for each type of IMU (chest, ankle, hand) should also be compiled to obtain the column names for the dataframe. Finally, all of these lists should be combined to generate the set of columns for the dataframe

```
#Import all the required libraries for the analysis
import pandas as pd
import numpy as np
import math
from scipy import stats
from matplotlib import pyplot as plt
%matplotlib inline
import math
import seaborn as sns
from sklearn.model_selection import train_test_split
import sklearn.model_selection as cross_validation
from pandas.plotting import scatter_matrix
pd.options.mode.chained_assignment = None

# Loading the data of 9 subjects
List_of_Subjects =
['Dataset/Protocol/subject101.dat','Dataset/Protocol/subject102.dat',

'Dataset/Protocol/subject103.dat','Dataset/Protocol/subject104.dat',

'Dataset/Protocol/subject105.dat','Dataset/Protocol/subject106.dat',

'Dataset/Protocol/subject107.dat','Dataset/Protocol/subject108.dat',
                 'Dataset/Protocol/subject109.dat' ]

subjectID = [1,2,3,4,5,6,7,8,9]

subject_weights={1:83,2:78,3:92,4:95,5:73,6:69,7:86,8:87,9:65}

MET_values = {1: 1,2: 1.8,3: 1.8,4: 3.5,5: 7.5,6: 4,7: 5.5,12:8,13:
3,16: 3.5,17: 2.3,24: 9}

# Creating a dictionary for all the activities

activityIDdict = {0: 'transient', 1: 'lying',2: 'sitting', 3:
'standing',4: 'walking', 5: 'running', 6: 'cycling',
              7: 'Nordic_walking',9: 'watching_TV',10:
'computer_work',11: 'car driving', 12: 'ascending_stairs',
              13: 'descending_stairs',16: 'vacuum_cleaning',17:
'ironing',18: 'folding_laundry',19: 'house_cleaning',
              20: 'playing_soccer',24: 'rope_jumping' }
```

```
IMUhand = ['handTemperature', 'hand_acc16_1', 'hand_acc16_2',
'hand_acc16_3','hand_acc6_1', 'hand_acc6_2',
           'hand_acc6_3', 'hand_gyro1', 'hand_gyro2',
'hand_gyro3','hand_magne1', 'hand_magne2', 'hand_magne3',
           'hand_orientation1', 'hand_orientation2',
'hand_orientation3', 'hand_orientation4']

IMUchest = ['chestTemperature', 'chest_acc16_1', 'chest_acc16_2',
'chest_acc16_3','chest_acc6_1', 'chest_acc6_2',
            'chest_acc6_3','chest_gyro1', 'chest_gyro2',
'chest_gyro3','chest_magne1', 'chest_magne2',
            'chest_magne3','chest_orientation1', 'chest_orientation2',
'chest_orientation3', 'chest_orientation4']

IMUankle = ['ankleTemperature','ankle_acc16_1', 'ankle_acc16_2',
'ankle_acc16_3', 'ankle_acc6_1',
            'ankle_acc6_2', 'ankle_acc6_3', 'ankle_gyro1',
'ankle_gyro2', 'ankle_gyro3',
            'ankle_magne1', 'ankle_magne2',
'ankle_magne3','ankle_orientation1', 'ankle_orientation2',
            'ankle_orientation3', 'ankle_orientation4']

columns=["timestamp", "activityID","heartrate"]
+IMUhand+IMUchest+IMUankle

len(columns)
```

```
54
```

DataFrame Creation: Each file in the files_list loops through and saved as CSV file that are stored in a new dataframe called phydf.A new column called subject_id is created and set to the subject ID extracted from the file name.The phydf dataframe is then appended to the pamdf dataframe using the concat function.

```
# Creating a new dataframe named pamdf using function create_dataframe
def create_dataframe(list_of_files):
    pamdf = pd.DataFrame()
    for file in list_of_files:
        df = pd.read_csv(file,sep='\s+|\s+',engine='python')
        strfile=file[17:27]+'.csv'
        df.to_csv(strfile, index=None)
        phydf=pd.read_csv(strfile)
        phydf.columns = columns
        phydf['subject_id'] = int(file[-5])
        pamdf=pd.concat([pamdf, phydf], ignore_index=True)
    return pamdf
```

```python
pamdf=create_dataframe(List_of_Subjects)
pamdf.reset_index(drop=True, inplace=True)
pamdf.head(5)
```

```
   timestamp  activityID  heartrate  handTemperature  hand_acc16_1  \
0       8.39           0        NaN             30.0       2.18837
1       8.40           0        NaN             30.0       2.37357
2       8.41           0        NaN             30.0       2.07473
3       8.42           0        NaN             30.0       2.22936
4       8.43           0        NaN             30.0       2.29959

   hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2
hand_acc6_3  ...  \
0       8.56560       3.66179      2.39494      8.55081
3.64207  ...
1       8.60107       3.54898      2.30514      8.53644
3.73280  ...
2       8.52853       3.66021      2.33528      8.53622
3.73277  ...
3       8.83122       3.70000      2.23055      8.59741
3.76295  ...
4       8.82929       3.54710      2.26132      8.65762
3.77788  ...

   ankle_gyro2  ankle_gyro3  ankle_magne1  ankle_magne2  ankle_magne3
\
0    -0.004638     0.000368      -59.8479      -38.8919      -58.5253

1     0.000148     0.022495      -60.7361      -39.4138      -58.3999

2    -0.020301     0.011275      -60.4091      -38.7635      -58.3956

3    -0.014303    -0.002823      -61.5199      -39.3879      -58.2694

4    -0.016024     0.001050      -60.2954      -38.8778      -58.3977


   ankle_orientation1  ankle_orientation2  ankle_orientation3  \
0                 1.0                 0.0                 0.0
1                 1.0                 0.0                 0.0
2                 1.0                 0.0                 0.0
3                 1.0                 0.0                 0.0
4                 1.0                 0.0                 0.0

   ankle_orientation4  subject_id
0                 0.0           1
1                 0.0           1
2                 0.0           1
3                 0.0           1
```

```
4                 0.0           1

[5 rows x 55 columns]
```

```python
# Displaying the last 5 rows
pamdf.tail(5)
```

```
         timestamp  activityID  heartrate  handTemperature  
hand_acc16_1  \
2872519     100.19           0        NaN          25.1875      -
4.71493   
2872520     100.20           0        NaN          25.1875      -
4.95932   
2872521     100.21           0        NaN          25.1875      -
4.93997   
2872522     100.22           0        NaN          25.1875      -
4.64941   
2872523     100.23           0      161.0          25.1875      -
4.09726   

         hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2  
hand_acc6_3  \
2872519      10.22250       4.66893     -5.04654      9.94944     
4.50736   
2872520      10.37130       4.12594     -4.96890     10.29620     
4.43102   
2872521       9.83615       3.70468     -5.04613     10.35690     
4.14405   
2872522       9.11129       3.51904     -5.06854      9.75268     
3.87359   
2872523       8.15642       3.29961     -4.73244      8.82870     
3.54305   

         ...  ankle_gyro2  ankle_gyro3  ankle_magne1  ankle_magne2  \
2872519  ...    -0.062676    -0.127084      -46.5153       3.58240   
2872520  ...    -0.027006    -0.089808      -45.7474       3.54453   
2872521  ...    -0.038024    -0.064709      -46.3997       4.22078   
2872522  ...    -0.025796    -0.064357      -46.5282       4.48593   
2872523  ...     0.011866    -0.042858      -46.2704       4.21475   

         ankle_magne3  ankle_orientation1  ankle_orientation2  \
2872519     -0.035995            0.598531            0.033615   
2872520      0.108583            0.598428            0.033012   
2872521      0.105504            0.598233            0.033172   
2872522      0.530240            0.598116            0.033427   
2872523      0.247798            0.598119            0.033685   

         ankle_orientation3  ankle_orientation4  subject_id  
2872519            0.799791           -0.031075           9  
2872520            0.799933           -0.030018           9  
```

```
2872521             0.800095            -0.029416           9
2872522             0.800180            -0.029208           9
2872523             0.800188            -0.028602           9

[5 rows x 55 columns]
```

As demonstrated by the above pamdf dataframe, some data cleaning is necessary. For example, activityID 0 should be completely removed from our dataset, as it represents a transitional period where the subject was not performing any specific activity (as indicated in the data_info file). The data cleaning process will be performed in the following section.

Data Cleaning

Per the data_info file, the dataset appears to have missing sensory data, due to wireless disconnections during data collection and the same are indicated with NaN. To account for this missing data and ensure that it does not impact the analysis, some data filling will be necessary. One way to address the missing values is to use interpolation, which involves constructing new data points based on known data points. Removing the columns that are not relevant for the analysis to expedite the runtime and avoid the unnecessary space.

Also, it is recommended to remove activity 0 from the data before interpolating, as it could significantly impact the output. After removing activity 0, the data can be interpolated to fill in the missing values.

```
# To Display the unique activityID
print(pamdf['activityID'].unique())
```

```
[ 0  1  2  3 17 16 12 13  4  7  6  5 24]
```

```
# Dropping the activityID = 0 from the dataframe
pamdf = pamdf.drop(pamdf[pamdf['activityID'] == 0].index)
pamdf.tail()
```

```
         timestamp  activityID  heartrate  handTemperature
hand_acc16_1  \
2872006      95.06          24        NaN           25.125
4.99466
2872007      95.07          24        NaN           25.125
5.02764
2872008      95.08          24        NaN           25.125
5.06409
2872009      95.09          24      162.0           25.125
5.13914
2872010      95.10          24        NaN           25.125
5.00812

         hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2
hand_acc6_3  \
2872006       6.01881       5.59830      4.90787      6.05780
```

```
5.68357
2872007       5.90369       5.48372      4.89090      5.95209
5.56301
2872008       5.71370       5.48491      4.97981      5.87584
5.45738
2872009       5.63724       5.48629      4.97690      5.69448
5.29167
2872010       5.40645       5.02326      4.97362      5.45272
5.14120

         ...  ankle_gyro2  ankle_gyro3  ankle_magne1  ankle_magne2  \
2872006  ...    -0.012885     0.005878      -45.7855     -0.831734
2872007  ...     0.003629    -0.004235      -46.0331     -0.817288
2872008  ...    -0.035176    -0.002309      -45.5140     -1.229410
2872009  ...    -0.036457    -0.007076      -45.9093     -0.565555
2872010  ...    -0.030195     0.018229      -46.1702     -0.812965

         ankle_magne3  ankle_orientation1  ankle_orientation2  \
2872006     -0.170139            0.522929           -0.291612
2872007      0.538134            0.522880           -0.291694
2872008      0.540438            0.522625           -0.291978
2872009      0.680109            0.522536           -0.291955
2872010     -0.313346            0.522730           -0.291275

         ankle_orientation3  ankle_orientation4  subject_id
2872006            0.705786           -0.378648           9
2872007            0.705895           -0.378450           9
2872008            0.706161           -0.378084           9
2872009            0.706426           -0.377733           9
2872010            0.706526           -0.377800           9

[5 rows x 55 columns]
```

```
# Check if the subject_id is equal to 0
pamdf[pamdf['subject_id']==0]
```

```
Empty DataFrame
Columns: [timestamp, activityID, heartrate, handTemperature,
hand_acc16_1, hand_acc16_2, hand_acc16_3, hand_acc6_1, hand_acc6_2,
hand_acc6_3, hand_gyro1, hand_gyro2, hand_gyro3, hand_magne1,
hand_magne2, hand_magne3, hand_orientation1, hand_orientation2,
hand_orientation3, hand_orientation4, chestTemperature, chest_acc16_1,
chest_acc16_2, chest_acc16_3, chest_acc6_1, chest_acc6_2,
chest_acc6_3, chest_gyro1, chest_gyro2, chest_gyro3, chest_magne1,
chest_magne2, chest_magne3, chest_orientation1, chest_orientation2,
chest_orientation3, chest_orientation4, ankleTemperature,
ankle_acc16_1, ankle_acc16_2, ankle_acc16_3, ankle_acc6_1,
ankle_acc6_2, ankle_acc6_3, ankle_gyro1, ankle_gyro2, ankle_gyro3,
ankle_magne1, ankle_magne2, ankle_magne3, ankle_orientation1,
ankle_orientation2, ankle_orientation3, ankle_orientation4,
```

```
subject_id]
Index: []

[0 rows x 55 columns]
```

```python
#Deletion of non numeric data
pamdf = pamdf.apply(pd.to_numeric, errors='coerce')

#Dropping the irrelevant coulmns
pamdf=pamdf.drop(columns=['hand_orientation1', 'hand_orientation2',
'hand_orientation3', 'hand_orientation4',
                                    'chest_orientation1',
'chest_orientation2', 'chest_orientation3', 'chest_orientation4',
                                    'ankle_orientation1',
'ankle_orientation2', 'ankle_orientation3',
                                    'ankle_orientation4'],axis=1)

pamdf.head(10)
```

```
      timestamp  activityID  heartrate  handTemperature  hand_acc16_1
\
2927      37.66           1        NaN           30.375       2.21530

2928      37.67           1        NaN           30.375       2.29196

2929      37.68           1        NaN           30.375       2.29090

2930      37.69           1        NaN           30.375       2.21800

2931      37.70           1      100.0           30.375       2.30106

2932      37.71           1        NaN           30.375       2.07165

2933      37.72           1        NaN           30.375       2.41148

2934      37.73           1        NaN           30.375       2.32815

2935      37.74           1        NaN           30.375       2.25096

2936      37.75           1        NaN           30.375       2.14107


      hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2
hand_acc6_3  ...  \
2927       8.27915       5.58753      2.24689      8.55387
5.77143  ...
2928       7.67288       5.74467      2.27373      8.14592
5.78739  ...
2929       7.14240       5.82342      2.26966      7.66268
5.78846  ...
```

```
2930       7.14365       5.89930      2.22177      7.25535
5.88000  ...
2931       7.25857       6.09259      2.20720      7.24042
5.95555  ...
2932       7.25965       6.01218      2.19238      7.21038
6.01604  ...
2933       7.59780       5.93915      2.23988      7.46679
6.03053  ...
2934       7.63431       5.70686      2.31663      7.64745
6.01495  ...
2935       7.78598       5.62821      2.28637      7.70801
5.93935  ...
2936       7.52262       5.78141      2.31538      7.72276
5.78828  ...

      ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  ankle_gyro1
ankle_gyro2  \
2927       9.63162      -1.76757      0.265761     0.002908    -
0.027714
2928       9.58649      -1.75247      0.250816     0.020882
0.000945
2929       9.60196      -1.73721      0.356632    -0.035392    -
0.052422
2930       9.58674      -1.78264      0.311453    -0.032514    -
0.018844
2931       9.64677      -1.75240      0.295902     0.001351    -
0.048878
2932       9.60177      -1.75239      0.311276     0.003793    -
0.026906
2933       9.67694      -1.76748      0.326060     0.036814    -
0.032277
2934       9.61685      -1.76749      0.326380    -0.010352    -
0.016621
2935       9.61686      -1.72212      0.326234     0.039346
0.020393
2936       9.63189      -1.70699      0.326105     0.029874    -
0.010763

      ankle_gyro3  ankle_magne1  ankle_magne2  ankle_magne3
subject_id
2927     0.001752      -61.1081      -36.8636      -58.3696
1
2928     0.006007      -60.8916      -36.3197      -58.3656
1
2929    -0.004882      -60.3407      -35.7842      -58.6119
1
2930     0.026950      -60.7646      -37.1028      -57.8799
1
2931    -0.006328      -60.2040      -37.1225      -57.8847
1
```

```
2932     0.004125      -61.3257      -36.9744      -57.7501
1
2933    -0.006866      -61.5520      -36.9632      -57.9957
1
2934     0.006548      -61.5738      -36.1724      -59.3487
1
2935    -0.011880      -61.7741      -37.1744      -58.1199
1
2936     0.005133      -60.7680      -37.4206      -58.8735
1

[10 rows x 43 columns]
```

```
# Performing the interpolation
pamdf = pamdf.interpolate()
```

```
pamdf.isnull().sum()
```

```
timestamp           0
activityID          0
heartrate           4
handTemperature     0
hand_acc16_1        0
hand_acc16_2        0
hand_acc16_3        0
hand_acc6_1         0
hand_acc6_2         0
hand_acc6_3         0
hand_gyro1          0
hand_gyro2          0
hand_gyro3          0
hand_magne1         0
hand_magne2         0
hand_magne3         0
chestTemperature    0
chest_acc16_1       0
chest_acc16_2       0
chest_acc16_3       0
chest_acc6_1        0
chest_acc6_2        0
chest_acc6_3        0
chest_gyro1         0
chest_gyro2         0
chest_gyro3         0
chest_magne1        0
chest_magne2        0
chest_magne3        0
ankleTemperature    0
ankle_acc16_1       0
ankle_acc16_2       0
ankle_acc16_3       0
```

```
ankle_acc6_1        0
ankle_acc6_2        0
ankle_acc6_3        0
ankle_gyro1         0
ankle_gyro2         0
ankle_gyro3         0
ankle_magne1        0
ankle_magne2        0
ankle_magne3        0
subject_id          0
dtype: int64
```

The heartrate column still contains NaN values because the interpolation process only calculates values based on the known data points surrounding a NaN cell. Since the first few cells in the hearrate column are NaN, it is normal for the interpolation to produce more NaN values. To rectify this issue, we can assume that the values in the first 4 cells of the hearrate column are 100,based on the values after index 4 are 100. This will allow the interpolation to properly fill in the missing value

```
# Replacing a NaN value using function
def clean_data(df, fill_value):
    df = df.fillna(fill_value)
    df.reset_index(drop=True, inplace=True)
    return df

# Replacing NaN to 100
pamdf = clean_data(pamdf, 100)
pamdf.head(10)
```

```
   timestamp  activityID  heartrate  handTemperature  hand_acc16_1  \
0      37.66           1      100.0           30.375       2.21530
1      37.67           1      100.0           30.375       2.29196
2      37.68           1      100.0           30.375       2.29090
3      37.69           1      100.0           30.375       2.21800
4      37.70           1      100.0           30.375       2.30106
5      37.71           1      100.0           30.375       2.07165
6      37.72           1      100.0           30.375       2.41148
7      37.73           1      100.0           30.375       2.32815
8      37.74           1      100.0           30.375       2.25096
9      37.75           1      100.0           30.375       2.14107

   hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2
hand_acc6_3  ...  \
0       8.27915       5.58753      2.24689      8.55387
5.77143  ...
1       7.67288       5.74467      2.27373      8.14592
5.78739  ...
2       7.14240       5.82342      2.26966      7.66268
5.78846  ...
3       7.14365       5.89930      2.22177      7.25535
```

```
5.88000  ...
4       7.25857       6.09259      2.20720      7.24042
5.95555  ...
5       7.25965       6.01218      2.19238      7.21038
6.01604  ...
6       7.59780       5.93915      2.23988      7.46679
6.03053  ...
7       7.63431       5.70686      2.31663      7.64745
6.01495  ...
8       7.78598       5.62821      2.28637      7.70801
5.93935  ...
9       7.52262       5.78141      2.31538      7.72276
5.78828  ...

   ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  ankle_gyro1  ankle_gyro2
\
0       9.63162      -1.76757      0.265761     0.002908    -0.027714

1       9.58649      -1.75247      0.250816     0.020882     0.000945

2       9.60196      -1.73721      0.356632    -0.035392    -0.052422

3       9.58674      -1.78264      0.311453    -0.032514    -0.018844

4       9.64677      -1.75240      0.295902     0.001351    -0.048878

5       9.60177      -1.75239      0.311276     0.003793    -0.026906

6       9.67694      -1.76748      0.326060     0.036814    -0.032277

7       9.61685      -1.76749      0.326380    -0.010352    -0.016621

8       9.61686      -1.72212      0.326234     0.039346     0.020393

9       9.63189      -1.70699      0.326105     0.029874    -0.010763


   ankle_gyro3  ankle_magne1  ankle_magne2  ankle_magne3  subject_id
0     0.001752      -61.1081      -36.8636      -58.3696           1
1     0.006007      -60.8916      -36.3197      -58.3656           1
2    -0.004882      -60.3407      -35.7842      -58.6119           1
3     0.026950      -60.7646      -37.1028      -57.8799           1
4    -0.006328      -60.2040      -37.1225      -57.8847           1
5     0.004125      -61.3257      -36.9744      -57.7501           1
6    -0.006866      -61.5520      -36.9632      -57.9957           1
7     0.006548      -61.5738      -36.1724      -59.3487           1
8    -0.011880      -61.7741      -37.1744      -58.1199           1
9     0.005133      -60.7680      -37.4206      -58.8735           1
```

```
[10 rows x 43 columns]

#To identify the count of null values
pamdf.isnull().sum()

timestamp           0
activityID          0
heartrate           0
handTemperature     0
hand_acc16_1        0
hand_acc16_2        0
hand_acc16_3        0
hand_acc6_1         0
hand_acc6_2         0
hand_acc6_3         0
hand_gyro1          0
hand_gyro2          0
hand_gyro3          0
hand_magne1         0
hand_magne2         0
hand_magne3         0
chestTemperature    0
chest_acc16_1       0
chest_acc16_2       0
chest_acc16_3       0
chest_acc6_1        0
chest_acc6_2        0
chest_acc6_3        0
chest_gyro1         0
chest_gyro2         0
chest_gyro3         0
chest_magne1        0
chest_magne2        0
chest_magne3        0
ankleTemperature    0
ankle_acc16_1       0
ankle_acc16_2       0
ankle_acc16_3       0
ankle_acc6_1        0
ankle_acc6_2        0
ankle_acc6_3        0
ankle_gyro1         0
ankle_gyro2         0
ankle_gyro3         0
ankle_magne1        0
ankle_magne2        0
ankle_magne3        0
subject_id          0
dtype: int64
```

Per the above summary, we do not have any missing values and the data is good to proceed with the exploratory data analysis.

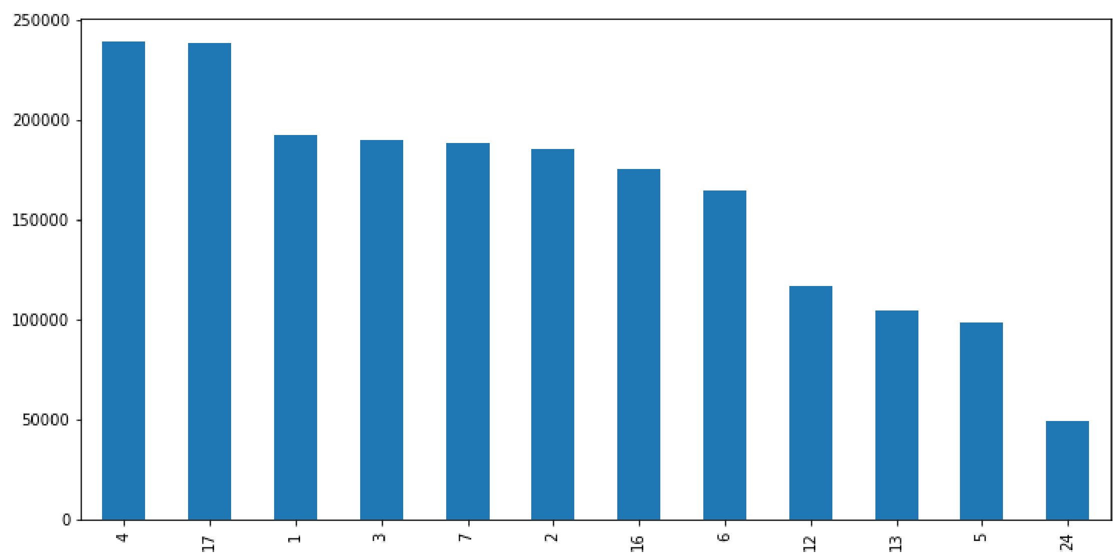
Exploratory Data Analysis(EDA)

In EDA, we are splitting the data into training and testing.This will allows us to validate the results of our analysis and ensure that our conclusions are not overly influenced by the specific data we are using.

Before dividing our data into training and testing sets, we should check if the dataframe are balanced. Because an imbalanced dataset can affect the performance of your model. If the dataframe distribution is skewed, with one dataframe much more prevalent than the other(s), the model may be biased towards the more prevalent dataframe, leading to poor performance on the minority dataframe.

In this section, we will be exploring the relationship between various physical activities and other variables by analyzing the data collected from the participants. This will include examining the different types of physical activities as well as other relevant factors such as heart rate, calories burnt, and time spent on each activity.

```
# To verify if the dataset is balanced
activity_counts = pamdf['activityID'].value_counts()

activity_counts.plot(kind = "bar", figsize = (12,6))
plt.show()
```



Per the above plot, the dataset is evenly balanced hence we are splitting the dataframe into 50% for train and 50% for test.

```
#Function to split the dataframe into train and test
def split_data(df, split_ratio):
    df = df.sample(frac=1, random_state=1)
```

```
    split_index = int(len(df) * split_ratio)
    train_df = df.iloc[:split_index]
    test_df = df.iloc[split_index:]
    return train_df, test_df

#To display the length and size of the train and test data
train_df, test_df = split_data(pamdf, 0.5)
print("Length of train_df is", len(train_df))
print('Size of the data: ', train_df.size)

print("Length of test_df is", len(test_df))
print('Size of the data: ', test_df.size)
```

```
Length of train_df is 971436
Size of the data:  41771748
Length of test_df is 971436
Size of the data:  41771748
```

Calculating the value_counts() to identify the unique actiivtyID and its respective count for both train and test data. This would be useful for getting a quick summary of the distribution of values in a categorical column.

```
train_df.activityID.value_counts()
```

```
17    119647
4     119187
1      96364
3      94933
7      93861
2      92764
16     87542
6      82516
12     58763
13     52377
5      48882
24     24600
Name: activityID, dtype: int64
```

```
test_df.activityID.value_counts()
```

```
4     119574
17    119043
1      96159
3      94998
7      94246
2      92424
16     87811
6      82084
12     58453
13     52567
5      49317
```

```
24     24760
Name: activityID, dtype: int64
```

```
# Sorting the rows by index value
train_df.sort_index()
```

```
         timestamp  activityID  heartrate  handTemperature  
hand_acc16_1  \
0            37.66           1      100.0           30.375       
2.21530   
2            37.68           1      100.0           30.375       
2.29090   
3            37.69           1      100.0           30.375       
2.21800   
5            37.71           1      100.0           30.375       
2.07165   
6            37.72           1      100.0           30.375       
2.41148   
...            ...         ...        ...              ...           .
..   
1942859      94.98          24      162.0           25.125       
4.81452   
1942861      95.00          24      162.0           25.125       
5.07290   
1942862      95.01          24      162.0           25.125       
4.95472   
1942863      95.02          24      162.0           25.125       
4.80517   
1942865      95.04          24      162.0           25.125       
4.95740   

         hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2  
hand_acc6_3  \
0             8.27915       5.58753      2.24689      8.55387      
5.77143   
2             7.14240       5.82342      2.26966      7.66268      
5.78846   
3             7.14365       5.89930      2.22177      7.25535      
5.88000   
5             7.25965       6.01218      2.19238      7.21038      
6.01604   
6             7.59780       5.93915      2.23988      7.46679      
6.03053   
...               ...           ...          ...          ...      
...   
1942859       6.51482       5.74788      4.89736      6.49594      
5.78832   
1942861       6.39761       5.59819      4.94094      6.45017      
5.63737   
1942862       6.28366       5.48134      4.93917      6.35946
```

```
5.51677
1942863      6.32311      5.51746     4.89281     6.22387
5.51711
1942865      6.28434      5.55836     4.81809     6.22448
5.59269

         ...  ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  ankle_gyro1  \
0        ...       9.63162      -1.76757      0.265761     0.002908
2        ...       9.60196      -1.73721      0.356632    -0.035392
3        ...       9.58674      -1.78264      0.311453    -0.032514
5        ...       9.60177      -1.75239      0.311276     0.003793
6        ...       9.67694      -1.76748      0.326060     0.036814
...      ...           ...           ...           ...          ...
1942859  ...       9.44267      -1.99702     -1.806020    -0.036682
1942861  ...       9.44276      -2.13311     -1.775310     0.027636
1942862  ...       9.39788      -2.13306     -1.729670    -0.005801
1942863  ...       9.48793      -2.20869     -1.745040    -0.028744
1942865  ...       9.36713      -2.42063     -1.879930    -0.041091

         ankle_gyro2  ankle_gyro3  ankle_magne1  ankle_magne2
ankle_magne3  \
0          -0.027714     0.001752      -61.1081    -36.863600    -
58.369600
2          -0.052422    -0.004882      -60.3407    -35.784200    -
58.611900
3          -0.018844     0.026950      -60.7646    -37.102800    -
57.879900
5          -0.026906     0.004125      -61.3257    -36.974400    -
57.750100
6          -0.032277    -0.006866      -61.5520    -36.963200    -
57.995700
...              ...          ...           ...           ...
...
1942859    -0.011895    -0.017897      -45.9167     -0.437698
0.254439
1942861    -0.024815    -0.022575      -46.2808     -1.320750
0.254182
1942862    -0.007817     0.009006      -45.9034     -1.211660     -
0.028281
1942863    -0.061156     0.033653      -46.0452     -0.690454     -
0.313048
1942865    -0.019494     0.014317      -45.8890     -1.596940
0.539545

         subject_id
0                 1
2                 1
3                 1
5                 1
6                 1
```

```
...           ...
1942859         9
1942861         9
1942862         9
1942863         9
1942865         9

[971436 rows x 43 columns]
```

Mapping the activity id to activity label:

The pamdatacop creates a copy of the train_df DataFrame, modify the activityID column using the values in the activityIDdict dictionary, and return the resulting pamdatacop.The purpose of this is to convert the activity IDs to more descriptive labels for easier interpretation.

```
pamdatacop=train_df.copy()
pamdatacop.activityID=pamdatacop.activityID.apply(lambda
x:activityIDdict[x])
pamdatacop
```

```
         timestamp        activityID   heartrate  handTemperature  \
312921      767.77          standing   90.000000          34.1875
141735     1926.37  ascending_stairs  167.909091          33.6875
1191085    3749.31      rope_jumping  181.000000          33.8125
1206914     255.84             lying   62.000000          33.4375
1710520     491.73           sitting   79.000000          34.3125
...            ...               ...         ...              ...
471794     3648.85           cycling  120.000000          29.7500
46078       498.44           sitting   92.000000          32.5625
495388     3961.26           running  139.000000          28.8750
1506095     783.52           ironing   74.000000          33.3750
699375      192.56             lying   72.000000          32.7500

         hand_acc16_1  hand_acc16_2  hand_acc16_3  hand_acc6_1
hand_acc6_2  \
312921       -8.52981      4.712060      0.358276    -8.254460
4.955160
141735      -11.23140      2.967660      2.325040   -10.353600
2.978210
1191085      -1.77987      3.716760     -7.388680    -0.572605
3.958700
1206914       4.89177     -0.216854      8.273420     5.134350    -
0.344501
1710520      -1.96494     -9.487270      0.889387    -1.802600    -
9.468570
...               ...           ...           ...          ...
...
471794       -6.10533      3.427980      7.365140    -6.577840
4.301450
```

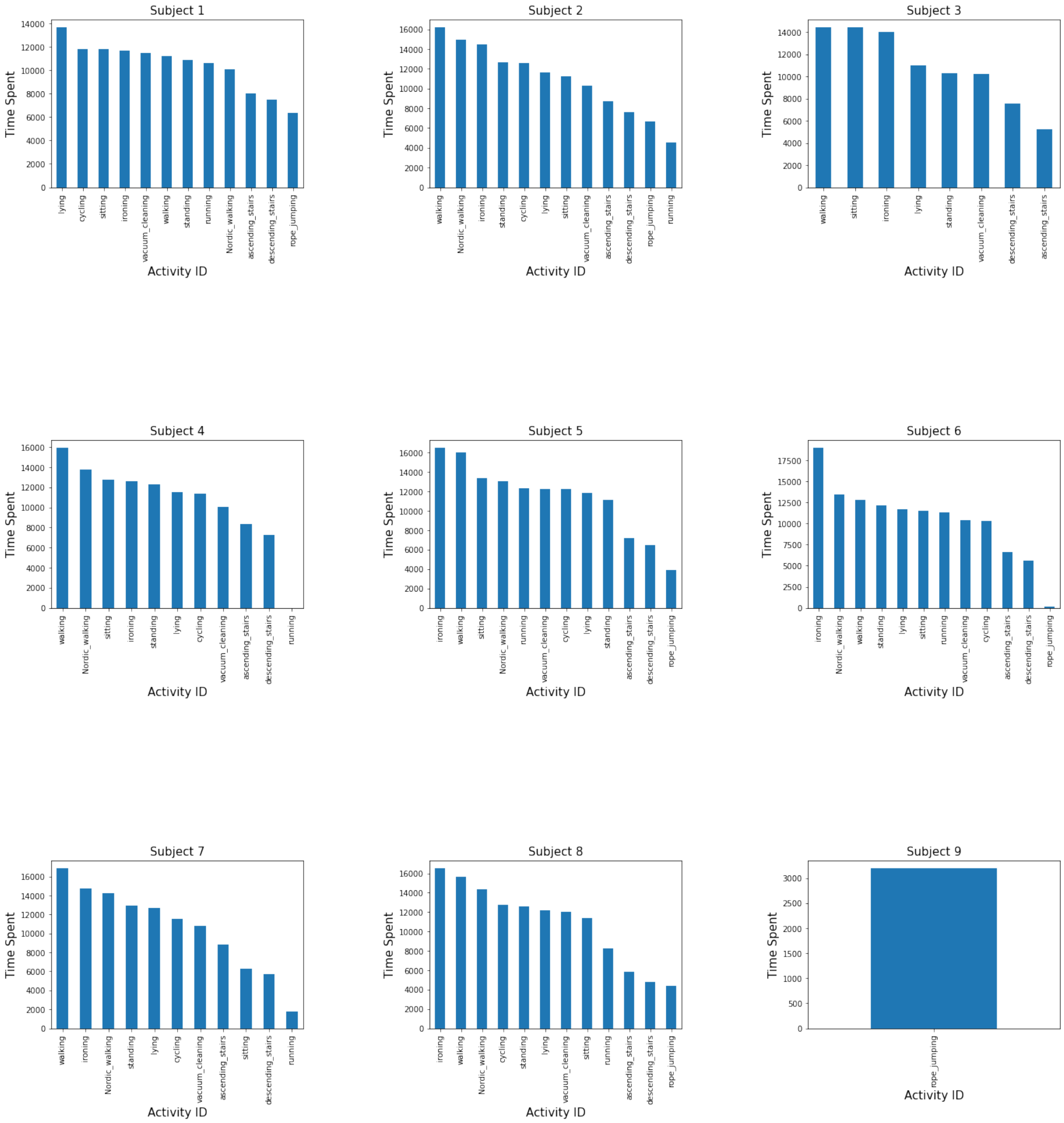
```
46078        -7.87425      4.095880      4.141940    -7.704180
4.138990
495388       -5.33359     36.979800     -3.435980    -3.119450
34.862600
1506095       2.11020     11.319200     -2.429680     2.211510
12.367400
699375        5.04452      0.198974      8.235820     5.409190
0.227261

         hand_acc6_3  ...  ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  \
312921      0.791698  ...      9.716540     -1.511980     -1.006870
141735      2.293150  ...     11.137500     -1.301760     -2.513480
1191085    -4.197230  ...     -4.136650     -4.006830      2.162320
1206914     8.596710  ...     -0.117976     -9.411680     -2.836260
1710520     1.058310  ...      9.187270      2.903000     -2.153400
...              ...  ...           ...           ...           ...
471794      9.142140  ...     10.602400     -2.752050     -1.022720
46078       4.356700  ...      9.135820      1.192120     -3.570340
495388     -2.873060  ...     11.857500      5.583830      0.381641
1506095    -2.346000  ...      9.890210     -0.561093     -2.660590
699375      8.444170  ...      2.514220     -9.420820      1.569270

         ankle_gyro1  ankle_gyro2  ankle_gyro3  ankle_magne1
ankle_magne2  \
312921      0.079272    -0.003718    -0.076595      -18.7246     -
16.87960
141735     -0.276670     0.669698    -0.143689      -51.7218
33.83850
1191085     1.877290    -0.176273    -0.654045      -44.6465     -
12.21680
1206914    -0.008012     0.002114     0.017758      -17.3221
26.44510
1710520     0.010209     0.011341     0.028921      -19.4371
15.82690
...              ...          ...          ...           ...
...
471794      0.154275     0.059888     0.106166      -41.1119
5.34881
46078       0.063705     0.008307     0.002250      -85.4454
38.14700
495388     -1.355570     1.238730    -3.278750      -51.4954      -
5.78950
1506095     0.426933    -0.741163     0.191537      -33.2568      -
1.96371
699375     -0.070824     0.009968     0.010577      -21.4016
15.80880

         ankle_magne3  subject_id
312921      36.806500           2
141735      -6.571620           1
```

```
1191085      0.272571          5
1206914     -4.782110          6
1710520     26.000300          8
...               ...        ...
471794       8.881400          2
46078       16.401700          1
495388      36.450200          2
1506095     46.112500          7
699375     -13.294100          4

[971436 rows x 43 columns]
```

Analysis of time spent on activity for individual subject:

We will be conducting a more detailed analysis by examining the amount of time each subject spent on different activities.

```
# Plotting the barchart using the function
def plot(df, sub, ax):
    df.activityID.value_counts().plot(kind='bar', ax=ax)
    df.activityID.agg(['value_counts'])
    ax.set_title('Subject {}'.format(sub), fontsize=15)
    ax.set_xlabel('Activity ID', fontsize=15)
    ax.set_ylabel('Time Spent', fontsize=15)


# Create a figure with 6 subplots arranged in a grid with 3 rows and 3
columns, with a larger size.
fig, ax = plt.subplots(3, 3, figsize=(9, 12))

# Iterate through the subject IDs
for i, sub in enumerate([1,2,3,4,5,6,7,8,9]):
    # Plot the bar chart for the subject and assign it to the
appropriate subplot
    plot(pamdatacop[pamdatacop['subject_id']==sub], sub, ax[i // 3][i
% 3])



# Adjust the spacing between the subplots
plt.subplots_adjust(left=0, bottom=0.5, right=2, top=2, wspace=0.5,
hspace=1.5)

# Show the plot
plt.show()
```



In the below code, we are grouping the training dataframe by activityID and computing the mean of the grouped data

```
df_grouped=train_df.groupby(['activityID']).mean()
df_grouped=df_grouped.reset_index()
df_grouped
```

```
    activityID    timestamp   heartrate  handTemperature  hand_acc16_1
\
0            1   205.810018   75.545557        32.728505      3.679602

1            2   506.308536   80.047179        33.258141     -1.389341

2            3   733.377930   88.536730        33.637740     -7.075932
```

```
3            4  2429.595921  112.779310        32.303069    -10.107593

4            5  3445.280586  156.609147        30.819405     -6.504239

5            6  3128.694951  124.830549        31.009282     -5.154956

6            7  2903.735748  123.775604        31.528332     -4.737611

7           12  1806.752318  129.518261        33.528458     -8.733582

8           13  1905.608580  129.094747        33.323658     -6.282657

9           16  1359.725161  104.182793        34.176997     -7.162320

10          17  1026.179980   90.045718        34.022595     -3.375883

11          24  3349.758610  161.966847        29.713111     -4.206803


    hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2  hand_acc6_3
...  \
0       2.061675      6.367305     3.791752     2.032754     6.588179
...
1       4.295544      5.173842    -1.262264     4.294076     5.383078
...
2       3.223117      2.675270    -6.939606     3.266560     2.893323
...
3       2.515623      1.922580   -10.100182     2.545913     2.092860
...
4       6.728369      0.287698    -6.624070     6.352816     0.388420
...
5       2.507518      7.122948    -5.158149     2.506534     7.262646
...
6       5.079735      2.543156    -4.736760     5.078315     2.692999
...
7       3.658778      1.603061    -8.643621     3.695112     1.801738
...
8       2.910228      3.656226    -6.184478     2.924561     3.853605
...
9       3.575758      1.913401    -7.031500     3.595148     2.132941
...
10      3.752365      5.447087    -3.246341     3.745548     5.665965
...
11      5.390808     -0.530344    -4.326484     5.410021    -0.539632
...

    ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  ankle_gyro1  ankle_gyro2
\
0       0.543936     -6.226228     -3.316962     0.010893    -0.005854
```

1	8.809982	-0.216851	-2.089557	0.006796	-0.005329
2	9.380298	-0.741048	-1.643213	0.004367	-0.004551
3	11.959444	0.632620	-2.693545	-0.005349	-0.113771
4	13.193339	2.880917	-3.189012	0.002644	-0.100271
5	9.130031	2.129800	-1.068567	0.085572	0.038773
6	12.262297	0.872126	-2.906883	-0.004517	-0.140149
7	9.788651	1.604495	-2.708451	0.385221	0.133281
8	10.742766	1.154140	-2.114203	-0.405388	-0.242240
9	9.563327	0.435375	-1.281936	-0.002207	0.005179
10	9.601066	-0.401777	-1.457873	0.011071	-0.002973
11	9.907947	1.084461	-2.047257	0.009809	-0.009883

	ankle_gyro3	ankle_magne1	ankle_magne2	ankle_magne3	subject_id
0	0.006003	-17.937375	20.577229	0.134306	4.485731
1	0.005010	-22.525800	2.503881	21.125203	4.306876
2	0.004309	-22.656027	-0.594149	24.427203	4.593166
3	-0.000048	-36.985424	-0.592640	15.840951	4.623625
4	-0.037299	-36.532773	-8.301071	13.229515	4.663884
5	0.003845	-38.848188	-6.929358	12.873313	4.701040
6	0.006245	-37.386417	-0.656277	14.345998	4.851653
7	-0.006582	-35.736146	-4.829318	13.768898	4.402396
8	0.111473	-36.826082	-4.451120	20.214430	4.166409
9	0.004223	-24.416449	4.187497	11.367237	4.550262
10	0.002770	-38.835212	5.579286	34.094910	4.726989

```
11     0.012182    -40.243773     -8.196516     23.698923    4.221707


[12 rows x 43 columns]
```

```
# Creating a new column named activity_name and grouping by
activityID, activity_name and heartrate

df_grouped['activity_name']=1
for i in range(len(df_grouped['activityID'])):
    df_grouped['activity_name']
[i]=activityIDdict[df_grouped['activityID'][i]]

df_grouped[['activityID','activity_name','heartrate']]
```

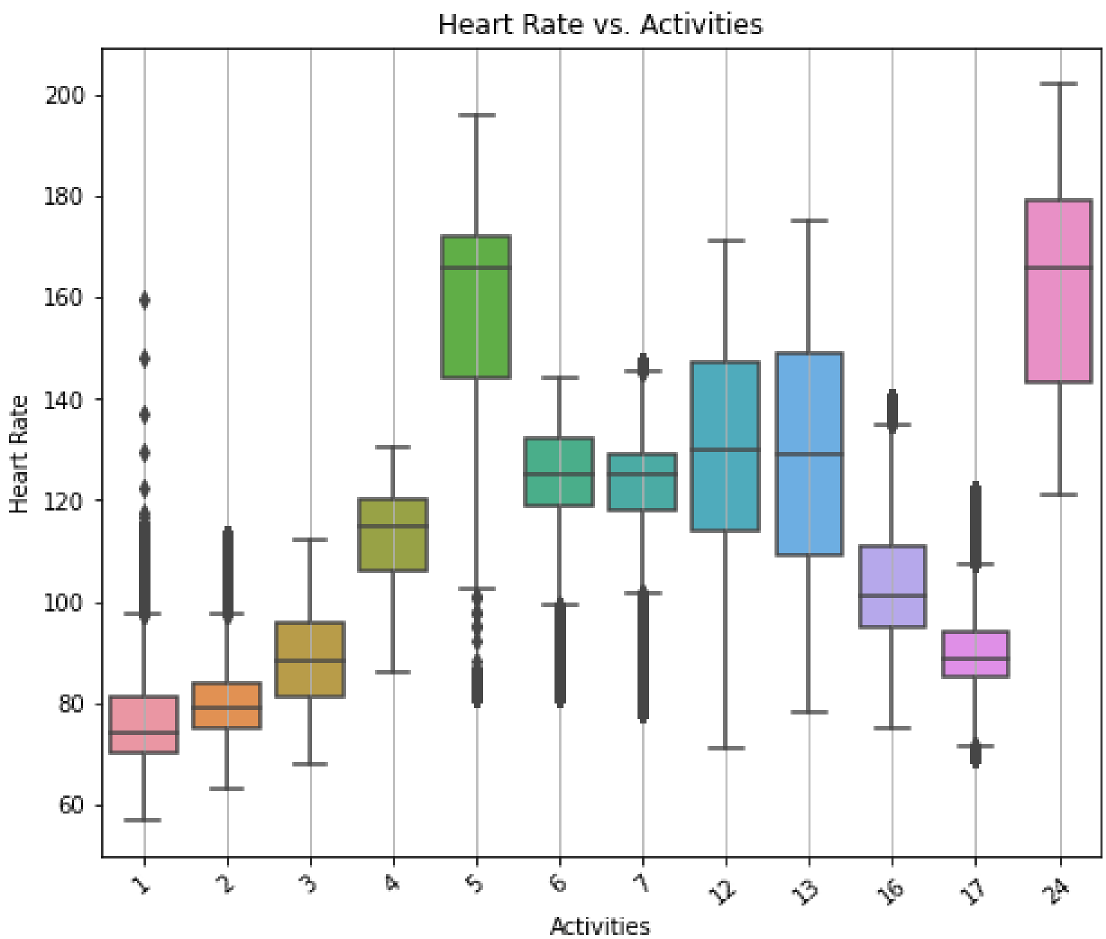
```
    activityID      activity_name   heartrate
0            1              lying   75.545557
1            2            sitting   80.047179
2            3           standing   88.536730
3            4            walking  112.779310
4            5            running  156.609147
5            6            cycling  124.830549
6            7     Nordic_walking  123.775604
7           12   ascending_stairs  129.518261
8           13  descending_stairs  129.094747
9           16    vacuum_cleaning  104.182793
10          17            ironing   90.045718
11          24       rope_jumping  161.966847
```

Boxplot for heartrate v/s activities:

Creating a box plot by showing the relationship between heartrate and activity using the Seaborn library's boxplot()

```
import seaborn as sns

# create a figure with a single subplot
fig, ax = plt.subplots(figsize=(7, 5))

# adjust the spacing between the subplots
plt.subplots_adjust(2, 1, 5, 2)
plt.subplot(131)

# create a boxplot of the data
sns.boxplot(x='activityID', y='heartrate',
data=train_df[['activityID', 'heartrate']])
```

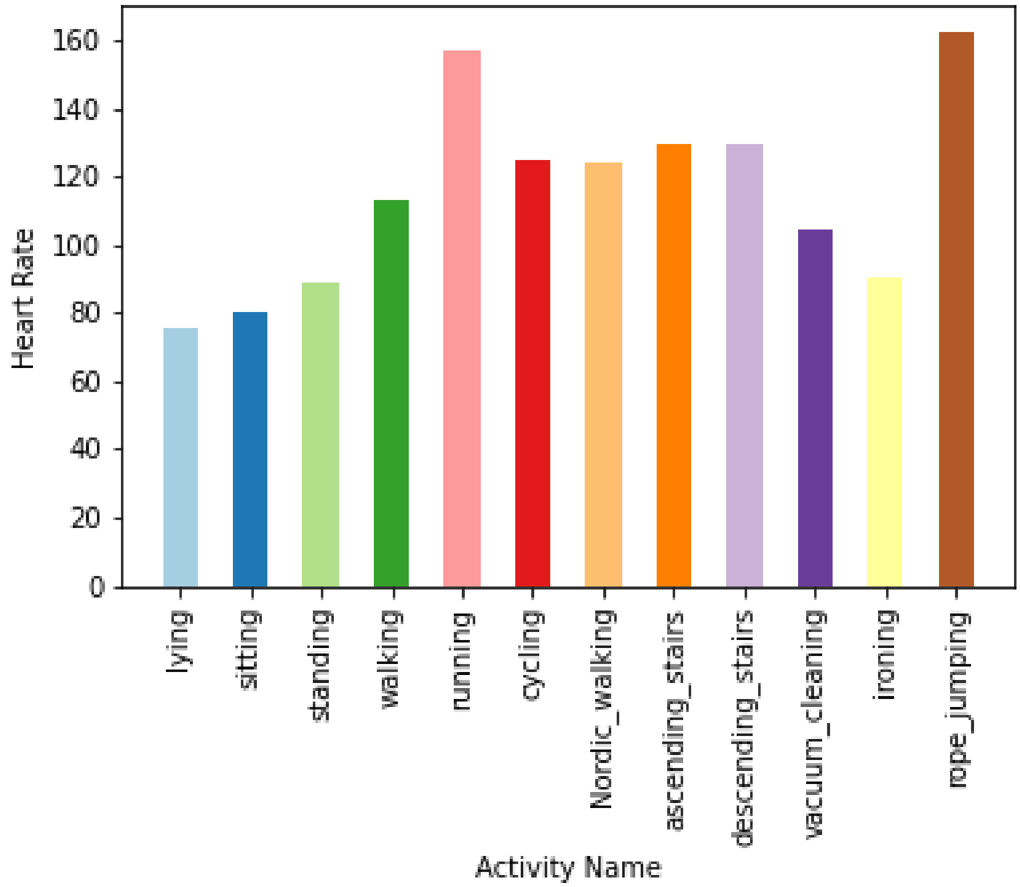
```
# rotate the x-axis labels
plt.xticks(rotation=40)

# show a grid on the x-axis
plt.grid(axis='x')

# label the x and y axes
plt.xlabel('Activities')
plt.ylabel('Heart Rate')

# add a title to the plot
plt.title('Heart Rate vs. Activities')

# show the plot
plt.show()
```



Barchart representation of Activity Name and Heartrate:

```
size = range(len(df_grouped))
plt.bar(df_grouped['activity_name'],df_grouped['heartrate'],
```

```
width=0.5, color=plt.cm.Paired(size))
plt.xticks(rotation=90)
plt.xlabel("Activity Name")
plt.ylabel("Heart Rate")
plt.show()
```



Per the above representations, rope jumping and running are the most strenuous activities among the ones represented in the chart.

Analysis of calories burnt v/s activities:

In this analysis, we are interested in the number of calories burnt by subjects during different activities. To calculate this, we are using the metabolic equivalent (MET) value of each activity and the weight of each subject.

```
def calorie_cal(i):

#Set the value of t= 60
    T=60
# Look up MET value for the current activity
    MET=MET_values[act_sub_grp['activityID'][i]]
```

```
  # Look up subject weight for the current subject
    W=subject_weights[act_sub_grp['subject_id'][i]]

   # Calculate number of calories burnt
    calories= T * MET * 3.5 * W / (200 * 60)

    return calories
```

Creating a new dataframe named act_sub_grp from train_df by grouping activityID and subjectID by applying the mean, max, and min functions to the timestamp column for each group, and adding calories burnt to that dataframe. Another dataframe calorie_act is created by grouping the activityID and subjectID of act_sub_grp and taking their mean. The activityID is then mapped to an activityName for clear interpretation of the data

```
# Group train_df by activityID and subject_id, and calculate mean,
max, and min values of timestamp
act_sub_grp=train_df.groupby(['activityID','subject_id'])
['timestamp'].agg([np.mean,max,min])
act_sub_grp=act_sub_grp.reset_index()
act_sub_grp['calorie_burnt']=0
# Iterate over activityID column of act_sub_grp
for i in range(len(act_sub_grp['activityID'])):
 # Calculate calorie_burnt for current activity and subject, and store
in act_sub_grp
    act_sub_grp['calorie_burnt'][i]=calorie_cal(i)

calorie_act=act_sub_grp.groupby(['activityID','subject_id']).mean()
calorie_act=calorie_act.reset_index()
# Replace values in activityID column with corresponding descriptions
using activityIDdict
calorie_act.activityID=calorie_act.activityID.apply(lambda
x:activityIDdict[x])
calorie_act
```

```
      activityID  subject_id         mean      max      min  
calorie_burnt
0          lying           1   174.045415   309.52    37.66  
1.4525
1          lying           2   171.966434   289.49    55.20  
1.3650
2          lying           3   276.886762   386.54   166.14  
1.6100
3          lying           4   190.067999   305.69    75.26  
1.6625
4          lying           5   223.699684   341.45   104.53  
1.2775
..           ...         ...          ...      ...      ...  
...
86  rope_jumping           2  4178.977066  4245.68  4113.08  
12.2850
```

```
87  rope_jumping          5  3714.966443  3753.50  3676.20
11.4975
88  rope_jumping          6  3622.808686  3624.05  3621.50
10.8675
89  rope_jumping          8  3844.586316  3888.41  3800.36
13.7025
90  rope_jumping          9    63.153766    95.04    31.22
10.2375

[91 rows x 6 columns]
```

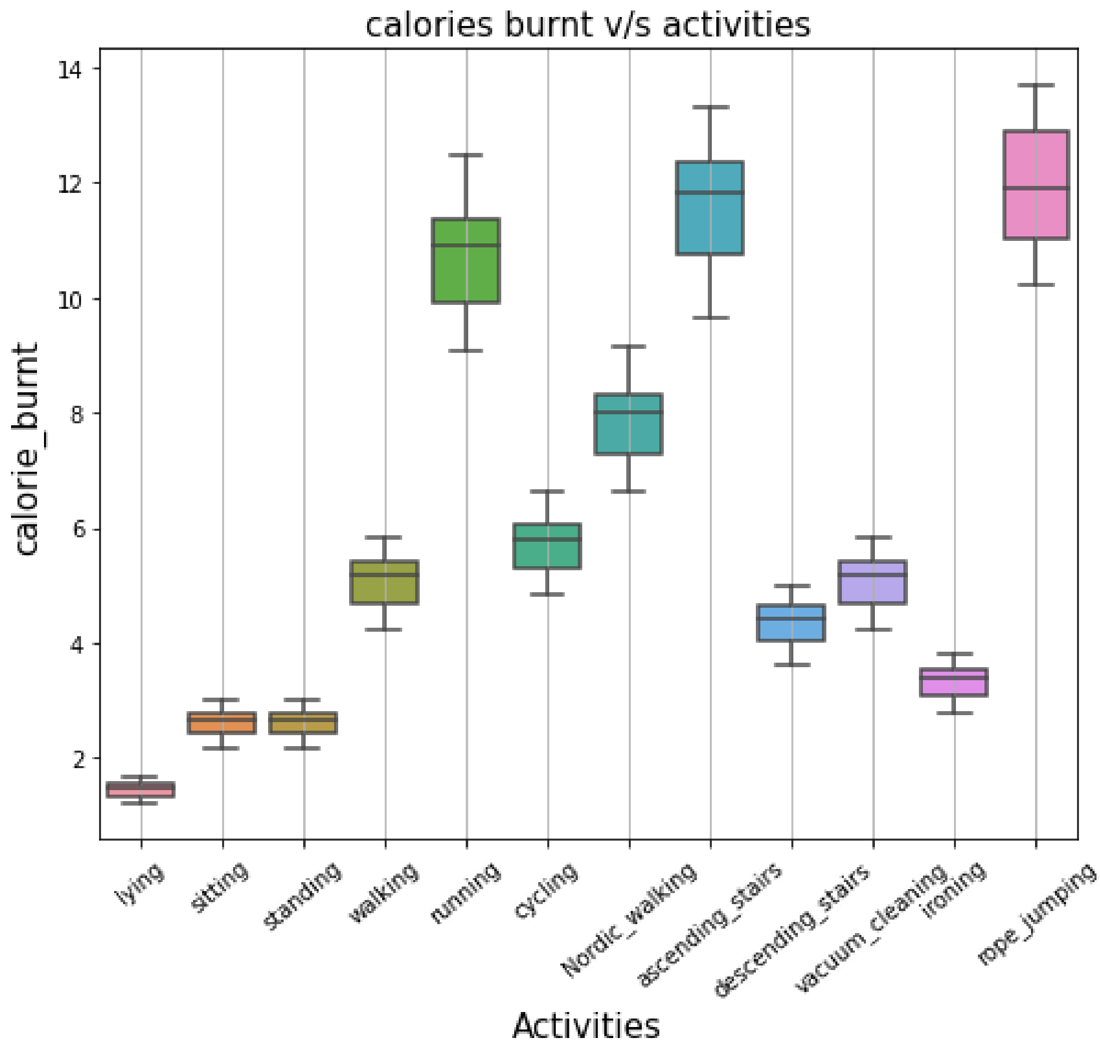
```
#Group calorie_act by activityID and calculate mean value
calorie_df_grouped=calorie_act.groupby(['activityID']).mean()
calorie_df_grouped.reset_index(drop=False, inplace=True)
calorie_df_grouped
```

```
           activityID  subject_id         mean          max
min  \
0      Nordic_walking    4.714286  2885.966026  3020.291429
2751.590000
1    ascending_stairs    4.500000  1807.821634  1999.022500
1615.946250
2             cycling    4.714286  3124.079746  3241.817143
3006.704286
3   descending_stairs    4.500000  1901.256871  2058.310000
1753.810000
4             ironing    4.500000  1023.305778  1172.390000
874.068750
5               lying    4.500000   206.958464   327.148750
86.527500
6        rope_jumping    5.166667  3164.427640  3205.508333
3123.270000
7             running    4.714286  3432.242946  3502.458571
3362.197143
8             sitting    4.500000   499.172845   614.961250
383.506250
9            standing    4.500000   736.287875   854.913750
617.530000
10    vacuum_cleaning    4.500000  1358.443971  1468.221250
1249.055000
11            walking    4.500000  2426.017357  2575.397500
2276.978750

    calorie_burnt
0        7.851250
1       11.602500
2        5.710000
3        4.350937
4        3.335719
5        1.450312
```

```
6       11.943750
7       10.706250
8        2.610562
9        2.610562
10       5.076094
11       5.076094
```

Boxplot for calories burnt v/s activities:

```
import seaborn as sns
plt.figure(figsize=(7,5))
plt.subplots_adjust(2,1,5,2)
plt.subplot(131)
dat1=calorie_act[['activityID','calorie_burnt']]
plt.xticks(rotation=40)
sns.boxplot(x='activityID',y='calorie_burnt',data=dat1)
plt.grid(axis='x')
plt.ylabel('calorie_burnt',fontsize=15)
plt.xlabel('Activities',fontsize=15)
plt.title('calories burnt v/s activities',fontsize=15)
plt.show()
```



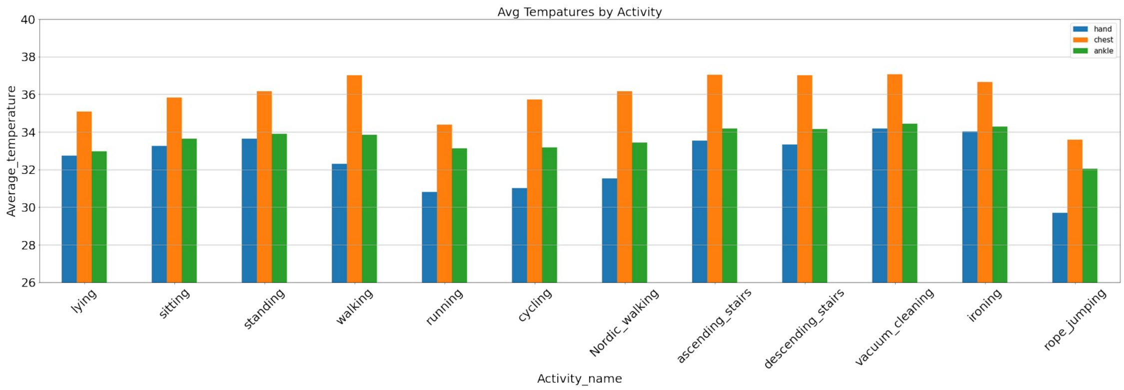
According to the boxplot, rope jumping and ascending stairs have the highest quartile for calories burnt, while lying, sitting, and standing have the lowest values.

Comparison of average temperature for hand,chest and ankle:

We are examining the average temperatures of different body parts (hand, chest, and ankle) while engaging in various activities.

```
Avg_Temp = pd.DataFrame()
Avg_Temp['hand'] = df_grouped['handTemperature']
Avg_Temp['chest'] = df_grouped['chestTemperature']
Avg_Temp['ankle'] = df_grouped['ankleTemperature']

ax = Avg_Temp.plot(kind='bar', figsize=(40,10), title='Avg Tempatures by Activity',fontsize=25)
a = ax.set_xticklabels(df_grouped['activity_name'],rotation=45)
b = ax.legend(fontsize = 15)
```

```python
c = ax.set_xticks(np.arange(len(Avg_Temp)))
plt.title('Avg Tempatures by Activity',fontsize=25)
plt.xlabel('Activity_name',fontsize=25)
plt.ylabel('Average_temperature',fontsize=25)
plt.ylim(26, 40)
plt.grid(axis='y')
```



Per the above bar chart, the chest temperature is significantly higher than the temperatures of the other body parts.

```python
# Grouping by activity name
df_grouped['activity_name']=1
for i in range(len(df_grouped['activityID'])):
    df_grouped['activity_name']
[i]=activityIDdict[df_grouped['activityID'][i]]
df_grouped
```

```
    activityID    timestamp   heartrate  handTemperature  hand_acc16_1
\
0            1   205.810018   75.545557        32.728505      3.679602

1            2   506.308536   80.047179        33.258141     -1.389341

2            3   733.377930   88.536730        33.637740     -7.075932

3            4  2429.595921  112.779310        32.303069    -10.107593

4            5  3445.280586  156.609147        30.819405     -6.504239

5            6  3128.694951  124.830549        31.009282     -5.154956

6            7  2903.735748  123.775604        31.528332     -4.737611

7           12  1806.752318  129.518261        33.528458     -8.733582

8           13  1905.608580  129.094747        33.323658     -6.282657

9           16  1359.725161  104.182793        34.176997     -7.162320
```

```
10          17  1026.179980   90.045718        34.022595     -3.375883

11          24  3349.758610  161.966847        29.713111     -4.206803


    hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2  hand_acc6_3
...  \
0       2.061675      6.367305     3.791752     2.032754     6.588179
...
1       4.295544      5.173842    -1.262264     4.294076     5.383078
...
2       3.223117      2.675270    -6.939606     3.266560     2.893323
...
3       2.515623      1.922580   -10.100182     2.545913     2.092860
...
4       6.728369      0.287698    -6.624070     6.352816     0.388420
...
5       2.507518      7.122948    -5.158149     2.506534     7.262646
...
6       5.079735      2.543156    -4.736760     5.078315     2.692999
...
7       3.658778      1.603061    -8.643621     3.695112     1.801738
...
8       2.910228      3.656226    -6.184478     2.924561     3.853605
...
9       3.575758      1.913401    -7.031500     3.595148     2.132941
...
10      3.752365      5.447087    -3.246341     3.745548     5.665965
...
11      5.390808     -0.530344    -4.326484     5.410021    -0.539632
...

    ankle_acc6_2  ankle_acc6_3  ankle_gyro1  ankle_gyro2  ankle_gyro3
\
0      -6.226228     -3.316962     0.010893    -0.005854     0.006003

1      -0.216851     -2.089557     0.006796    -0.005329     0.005010

2      -0.741048     -1.643213     0.004367    -0.004551     0.004309

3       0.632620     -2.693545    -0.005349    -0.113771    -0.000048

4       2.880917     -3.189012     0.002644    -0.100271    -0.037299

5       2.129800     -1.068567     0.085572     0.038773     0.003845

6       0.872126     -2.906883    -0.004517    -0.140149     0.006245
```

```
7        1.604495     -2.708451     0.385221     0.133281    -0.006582

8        1.154140     -2.114203    -0.405388    -0.242240     0.111473

9        0.435375     -1.281936    -0.002207     0.005179     0.004223

10      -0.401777     -1.457873     0.011071    -0.002973     0.002770

11       1.084461     -2.047257     0.009809    -0.009883     0.012182


    ankle_magne1  ankle_magne2  ankle_magne3  subject_id
activity_name
0     -17.937375     20.577229      0.134306    4.485731
lying
1     -22.525800      2.503881     21.125203    4.306876
sitting
2     -22.656027     -0.594149     24.427203    4.593166
standing
3     -36.985424     -0.592640     15.840951    4.623625
walking
4     -36.532773     -8.301071     13.229515    4.663884
running
5     -38.848188     -6.929358     12.873313    4.701040
cycling
6     -37.386417     -0.656277     14.345998    4.851653
Nordic_walking
7     -35.736146     -4.829318     13.768898    4.402396
ascending_stairs
8     -36.826082     -4.451120     20.214430    4.166409
descending_stairs
9     -24.416449      4.187497     11.367237    4.550262
vacuum_cleaning
10    -38.835212      5.579286     34.094910    4.726989
ironing
11    -40.243773     -8.196516     23.698923    4.221707
rope_jumping

[12 rows x 44 columns]
```

Correlation coefficient:

Correlation coefficient is used to assess the strength of associations between data variables.In this, we are analysing the correlation coefficient for the train_df.

```
df_correlation=train_df.copy()
df_correlation.reset_index(drop=True, inplace=True)
df_correlation
```

```
        timestamp  activityID   heartrate  handTemperature  
hand_acc16_1  \
0          767.77           3   90.000000          34.1875      -
8.52981   
1         1926.37          12  167.909091          33.6875     -
11.23140   
2         3749.31          24  181.000000          33.8125      -
1.77987   
3          255.84           1   62.000000          33.4375       
4.89177   
4          491.73           2   79.000000          34.3125      -
1.96494   
...           ...         ...         ...              ...           .
..   
971431    3648.85           6  120.000000          29.7500      -
6.10533   
971432     498.44           2   92.000000          32.5625      -
7.87425   
971433    3961.26           5  139.000000          28.8750      -
5.33359   
971434     783.52          17   74.000000          33.3750       
2.11020   
971435     192.56           1   72.000000          32.7500       
5.04452   

        hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2  
hand_acc6_3  \
0           4.712060      0.358276    -8.254460     4.955160     
0.791698   
1           2.967660      2.325040   -10.353600     2.978210     
2.293150   
2           3.716760     -7.388680    -0.572605     3.958700    -
4.197230   
3          -0.216854      8.273420     5.134350    -0.344501     
8.596710   
4          -9.487270      0.889387    -1.802600    -9.468570     
1.058310   
...              ...           ...          ...          ...          
...   
971431      3.427980      7.365140    -6.577840     4.301450     
9.142140   
971432      4.095880      4.141940    -7.704180     4.138990     
4.356700   
971433     36.979800     -3.435980    -3.119450    34.862600    -
2.873060   
971434     11.319200     -2.429680     2.211510    12.367400    -
2.346000   
971435      0.198974      8.235820     5.409190     0.227261     
8.444170   
```

```
        ...  ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  ankle_gyro1  \
0       ...      9.716540     -1.511980     -1.006870     0.079272
1       ...     11.137500     -1.301760     -2.513480    -0.276670
2       ...     -4.136650     -4.006830      2.162320     1.877290
3       ...     -0.117976     -9.411680     -2.836260    -0.008012
4       ...      9.187270      2.903000     -2.153400     0.010209
...     ...           ...           ...           ...          ...
971431  ...     10.602400     -2.752050     -1.022720     0.154275
971432  ...      9.135820      1.192120     -3.570340     0.063705
971433  ...     11.857500      5.583830      0.381641    -1.355570
971434  ...      9.890210     -0.561093     -2.660590     0.426933
971435  ...      2.514220     -9.420820      1.569270    -0.070824

        ankle_gyro2  ankle_gyro3  ankle_magne1  ankle_magne2
ankle_magne3  \
0         -0.003718    -0.076595      -18.7246     -16.87960
36.806500
1          0.669698    -0.143689      -51.7218      33.83850     -
6.571620
2         -0.176273    -0.654045      -44.6465     -12.21680
0.272571
3          0.002114     0.017758      -17.3221      26.44510     -
4.782110
4          0.011341     0.028921      -19.4371      15.82690
26.000300
...             ...          ...           ...           ...
...
971431     0.059888     0.106166      -41.1119       5.34881
8.881400
971432     0.008307     0.002250      -85.4454      38.14700
16.401700
971433     1.238730    -3.278750      -51.4954      -5.78950
36.450200
971434    -0.741163     0.191537      -33.2568      -1.96371
46.112500
971435     0.009968     0.010577      -21.4016      15.80880    -
13.294100

        subject_id
0                2
1                1
2                5
3                6
4                8
...            ...
971431           2
971432           1
971433           2
971434           7
971435           4
```

```
[971436 rows x 43 columns]
```

The function axis_reducer takes a dataframe and three column names as input and creates a new column by concatenating the values in the specified columns. It then removes the original columns from the dataframe and returns the modified dataframe.

```
def axis_reducer(df,a,b,c,d):
    ref=df.copy()
    ref[d]=1
    for i in range(len(ref[a])):

        ref[d][i]=(ref[a][i]**2 + ref[b][i]**2 + ref[c][i]**2)**0.5

    ref=ref.drop([a,b,c], axis=1)



    return ref
```

```
# Calling the function axis_reducer and combining the various
variables of hand, chest and ankle in a single column

ref=axis_reducer(df_correlation,'hand_acc16_1','hand_acc16_2','hand_ac
c16_3','Hand_Acceleration_16')
ref=axis_reducer(ref,'hand_acc6_1','hand_acc6_2','hand_acc6_3','Hand_A
cceleration_6')
ref=axis_reducer(ref,'hand_gyro1', 'hand_gyro2',
'hand_gyro3','Hand_Gyrometer')
ref=axis_reducer(ref,'hand_magne1','hand_magne2','hand_magne3','Hand_m
agnetometer')

ref=axis_reducer(ref,'ankle_acc16_1','ankle_acc16_2','ankle_acc16_3','
Ankle_Acceleration_16')
ref=axis_reducer(ref,'ankle_acc6_1', 'ankle_acc6_2',
'ankle_acc6_3','Ankle_Acceleration_6')
ref=axis_reducer(ref,'ankle_gyro1', 'ankle_gyro2',
'ankle_gyro3','Ankle_Gyrometer')
ref=axis_reducer(ref,'ankle_magne1','ankle_magne2','ankle_magne3','Ank
le_Magnetometer')

ref=axis_reducer(ref,'chest_acc16_1', 'chest_acc16_2',
'chest_acc16_3','Chest_Acceleration_16')
ref=axis_reducer(ref,'chest_acc6_1', 'chest_acc6_2',
'chest_acc6_3','Chest_Acceleration_6')
ref=axis_reducer(ref,'chest_gyro1', 'chest_gyro2',
'chest_gyro3','Chest_Gyrometer')
ref=axis_reducer(ref,'chest_magne1',
'chest_magne2','chest_magne3','Chest_Magnetometer')
ref
```

```
        timestamp  activityID   heartrate  handTemperature
chestTemperature  \
0          767.77           3   90.000000          34.1875
37.5000
1         1926.37          12  167.909091          33.6875
36.9375
2         3749.31          24  181.000000          33.8125
36.4375
3          255.84           1   62.000000          33.4375
35.5625
4          491.73           2   79.000000          34.3125
37.6250
...           ...         ...         ...              ...
...
971431    3648.85           6  120.000000          29.7500
34.7500
971432     498.44           2   92.000000          32.5625
34.1875
971433    3961.26           5  139.000000          28.8750
32.2500
971434     783.52          17   74.000000          33.3750
36.1250
971435     192.56           1   72.000000          32.7500
34.7500

        ankleTemperature  subject_id  Hand_Acceleration_16  \
0                34.8125           2              9.751386
1                34.9375           1             11.847243
2                34.1250           5              8.460191
3                34.6250           6              9.613840
4                33.9375           8              9.729352
...                  ...         ...                   ...
971431           33.6250           2             10.162253
971432           32.9375           1              9.794678
971433           31.7500           2             37.520111
971434           32.5625           7             11.767777
971435           33.4375           4              9.659995

        Hand_Acceleration_6  Hand_Gyrometer  Hand_magnetometer  \
0                  9.660047        0.164967          34.928264
1                 11.014777        2.181952          48.546151
2                  5.797924        4.643583          46.379032
3                 10.019164        0.062214          48.484174
4                  9.696556        0.057963          29.258687
...                     ...             ...                ...
971431            12.056085        0.268151          53.481398
971432             9.770694        0.041158          56.686076
971433            35.119600        2.651422          56.899400
971434            12.780731        4.275455          44.155289
971435            10.030702        0.039176          53.783521
```

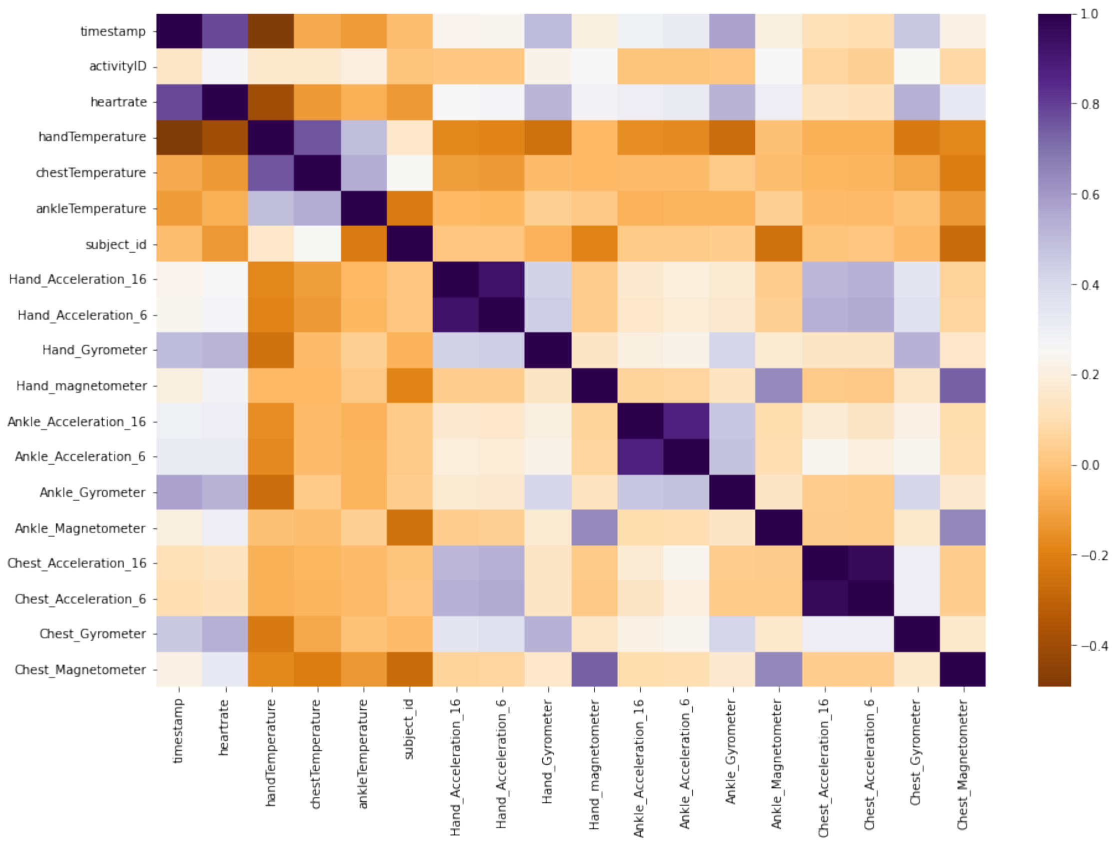
```
        Ankle_Acceleration_16  Ankle_Acceleration_6
Ankle_Gyrometer  \
0                    9.853628              9.884888         0.110294

1                   11.880628             11.491565         0.738707

2                    7.628115              6.151600         1.995762

3                   10.163821              9.830463         0.019596

4                    9.926956              9.872713         0.032699

...                       ...                   ...              ...

971431              10.488048             11.001392         0.196618

971432              10.033660              9.880875         0.064284

971433              11.042984             13.112022         3.757955

971434              10.479384             10.257184         0.876516

971435               9.848968              9.876019         0.072300


        Ankle_Magnetometer  Chest_Acceleration_16
Chest_Acceleration_6  \
0                44.612218              10.060474
9.855084
1                62.156053              13.478711
12.883012
2                46.288600              11.259950
7.621981
3                31.972911               9.440225
9.771592
4                36.115194               9.970973
9.974286
...                    ...                    ...                   ..
.
971431           42.399025              12.667246
12.987450
971432           95.000662               9.771799
9.806011
971433           63.355439              30.175062
24.763849
971434           56.887903              10.583624
10.518973
971435           29.743566               9.600552
```

```
9.884871

        Chest_Gyrometer  Chest_Magnetometer
0              0.084096           27.869222
1              0.474735           43.174891
2              2.515189           49.537940
3              0.034585           49.229344
4              0.083730           26.310660
...                 ...                 ...
971431         0.233825           41.916091
971432         0.033836           74.998591
971433         4.356266           53.524002
971434         1.351819           37.290062
971435         0.055825           49.388521

[971436 rows x 19 columns]
```

Heatmap Generation:

Heatmap is used to visually represent data in order to identify patterns and correlations. The heatmap provides a graphical representation of the data, using size and color to show multiple pieces of data in a way that is easy to understand and interpret.

```
df_corr = ref.corr(method ='pearson')
df_corr = df_corr.drop(['activityID'], axis = 1)

f, ax = plt.subplots(figsize=(15, 10))
sns.heatmap(df_corr,cmap = "PuOr", mask=np.zeros_like(df_corr,
dtype=bool))
plt.show()
```



The heatmap suggests a strong correlation between heartrate and the gyrometer readings for the hand, chest, and ankle, as evidenced by the dark colors in the corresponding cells. Additionally, there is a negative correlation between the hand temperature and heartrate.

This concludes the data analysis of various attributes and proceeding to hypothesis testing.

Hypothesis Testing:

Based on the comparison of different physical activities that have been analyzed, it has been determined that the two most intense activities are rope jumping and ascending stairs. In order to further investigate the intensity of these activities and make a decision about the true intensity level, we will be conducting a hypothesis test.

Hypothesis: IF, the subject is performing intense activities like rope jumping and ascending stairs THEN average heartrate of the subject will be more than 105."

- Independent Variable :Rope jumping and ascending stairs
- Dependent Variable : Average Heart rate

H0 : The average heartrate of the subject while performing intense activities is less than or equal to 105 (μ <= 105).

H1 : Intense physical activities have an impact on the heartrate readings ie,(μ > 105).

```
ascending_stairs = train_df[train_df["activityID"] == 12]
ropejumping= train_df[train_df["activityID"] == 24]
Intense_activity=pd.DataFrame()
Intense_activity =
pd.concat([Intense_activity,ascending_stairs,ropejumping],
ignore_index=False)
Intense_activity
```

```
         timestamp  activityID   heartrate  handTemperature
hand_acc16_1  \
141735     1926.37          12  167.909091          33.6875    -
11.231400
812554     1554.29          12  149.000000          32.5625    -
10.057200
1326831    1716.24          12  153.000000          33.6250     -
6.199510
822542     1797.06          12  115.000000          32.6250    -
10.322200
827744     1849.08          12  150.000000          32.5625     -
9.374930
...            ...         ...         ...              ...
...
247533     3602.14          24  181.000000          30.1875     -
1.027870
511742     4230.05          24  179.000000          28.5000
0.634975
1184329    3681.75          24  139.000000          33.8750     -
9.025420
501024     4122.87          24  123.000000          28.3125     -
0.539344
237828     3505.09          24  129.000000          30.1875     -
1.603240

         hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2
hand_acc6_3  \
141735       2.967660      2.325040   -10.353600     2.978210
2.293150
812554       5.136370     -0.204857    -9.799700     5.101200    -
0.309397
1326831      5.790310      4.161310    -6.227590     5.925460
4.305980
822542       2.592290      3.571520   -10.719300     2.529440
4.076400
827744       1.932020      2.702730    -9.607850     1.644200
2.703030
...               ...           ...          ...          ...
...
247533       7.147790     -4.925800    -1.094920     6.650410    -
4.489510
511742      -0.968486      1.130520     1.424530    -2.712040    -
```

```
0.137652
1184329      3.777730      1.508630    -8.812540     3.887880
1.821460
501024      -1.067640    -10.933300    -0.241368     0.023247   -
11.120200
237828       7.814620      1.221310    -1.494780     7.609710
0.824271

         ...  ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  ankle_gyro1  \
141735   ...      11.13750     -1.301760     -2.513480    -0.276670
812554   ...      10.93450      0.167377     -0.837120    -0.042031
1326831  ...       9.51498     -0.772813     -2.567110     0.062419
822542   ...       9.74073      5.095370     -2.874750     0.051499
827744   ...      17.52620      4.584040     -1.915530    -1.356800
...      ...           ...           ...           ...          ...
247533   ...       2.47699     -2.874380     -0.615926     1.523360
511742   ...       7.19682      1.316780     -0.230692    -0.532160
1184329  ...       9.24221      0.678307     -3.236270    -0.143172
501024   ...       5.25898      5.559860      6.713020    -0.230153
237828   ...      25.24460     -2.021600     12.276700     0.664214

         ankle_gyro2  ankle_gyro3  ankle_magne1  ankle_magne2
ankle_magne3  \
141735      0.669698    -0.143689     -51.72180      33.83850     -
6.571620
812554     -0.114733    -0.994883     -38.26050     -18.97500     -
0.548117
1326831    -0.087208     0.030497     -25.77890      13.95410
25.838400
822542      0.139639     1.070840      -6.90424     -15.04980
27.214900
827744     -0.280870    -3.004700     -55.06430     -12.10490     -
6.997640
...              ...          ...           ...           ...
...
247533      0.133100     2.189300     -51.87950     -34.34670
36.148700
511742     -0.233961    -1.513260     -38.23570      -1.37933
34.655900
1184329    -0.223484    -0.306898     -46.12190      19.94060
9.165500
501024      0.232471     2.192700     -36.09650     -12.53920
30.187100
237828      0.278529     1.631470     -65.96960      12.86130    -
17.571400

         subject_id
141735            1
812554            4
1326831           6
```

```
822542            4
827744            4
...             ...
247533            1
511742            2
1184329           5
501024            2
237828            1

[83363 rows x 43 columns]
```

```
Intense_activity_mean=Intense_activity['heartrate'].mean()
Intense_activity_std=Intense_activity['heartrate'].std()
Intense_activity_count=Intense_activity['heartrate'].count()
Intense_activity_count
```

```
83363
```

```
Intense_activity_std
```

```
25.820132755931425
```

```
Intense_activity_mean
```

```
139.09367431726955
```

```
z=(Intense_activity_mean-105)/(Intense_activity_std/
np.sqrt(Intense_activity_count))
pValue=(1-stats.norm.cdf(z))

if pValue>0.1:

    print ("The pValue is: ",pValue, "and h1 is rejected")
else:
    print ("The pValue is: ", pValue, "and h0 is rejected")
```

```
The pValue is:  0.0 and h0 is rejected
```

In this analysis, a significance level of 5% will be used to guide the decision making process. After conducting a z-test, the obtained p-value is 0.05 which means that there is only a 5% chance that the results are due to chance. Since this p-value is less than the significance level of 5%, we must reject the null hypothesis which states that the average heart rate of the subject while performing intense activities such as rope jumping and ascending stairs is less than 105 beats per minute. Instead, we will accept the alternative hypothesis, which asserts that the average heart rate is greater than 105 beats per minute.

Modelling:

For the purpose of predicting heart rate, we will focus on the chest readings. Our exploratory data analysis showed that there is a strong correlation between heart rate and the chest_acc16_2 and chest_magne3 variables. Therefore, we will create an independent variable X that includes both chest_acc16_2 and chest_magne3 and a dependent variable Y that includes heart rate.

```
pamdf.corr()
```

```
                 timestamp  activityID  heartrate  handTemperature  \
timestamp         1.000000    0.143712   0.781720        -0.493674
activityID        0.143712    1.000000   0.265348         0.159352
heartrate         0.781720    0.265348   1.000000        -0.395932
handTemperature  -0.493674    0.159352  -0.395932         1.000000
hand_acc16_1     -0.321940   -0.129905  -0.298458         0.059130
hand_acc16_2      0.039321    0.038609   0.065728        -0.053288
hand_acc16_3     -0.215706   -0.116724  -0.268278         0.077812
hand_acc6_1      -0.332616   -0.129179  -0.307739         0.073200
hand_acc6_2       0.036865    0.042104   0.063252        -0.048332
hand_acc6_3      -0.226055   -0.118394  -0.277938         0.094127
hand_gyro1        0.011050    0.019513   0.012257        -0.025622
hand_gyro2        0.047300    0.027951   0.059773        -0.023345
hand_gyro3        0.002810    0.001211  -0.001140        -0.001049
hand_magne1       0.337544    0.053005   0.322611        -0.096713
hand_magne2      -0.095922   -0.188841  -0.134859        -0.024213
hand_magne3       0.084492   -0.023507   0.106754        -0.022195
chestTemperature -0.084272    0.159571  -0.127371         0.757514
chest_acc16_1    -0.027998   -0.150098  -0.018564         0.005239
chest_acc16_2     0.157065    0.105353   0.151197         0.034396
chest_acc16_3    -0.469099   -0.428336  -0.409800         0.098116
chest_acc6_1     -0.026889   -0.147677  -0.018807         0.019417
chest_acc6_2      0.158540    0.103335   0.153499         0.032130
chest_acc6_3     -0.471208   -0.429389  -0.412970         0.107511
chest_gyro1       0.005108    0.000238   0.007120        -0.001823
chest_gyro2       0.025507   -0.020597   0.016387        -0.013415
chest_gyro3      -0.018143    0.003579  -0.021720         0.004979
chest_magne1     -0.189442   -0.241078  -0.157438        -0.043054
chest_magne2     -0.368725   -0.298399  -0.403923        -0.001476
chest_magne3      0.499251    0.265201   0.449597        -0.180812
ankleTemperature -0.123084    0.195023  -0.062178         0.495376
ankle_acc16_1     0.324924    0.133338   0.283501        -0.062394
ankle_acc16_2     0.200998    0.097188   0.193728        -0.047370
ankle_acc16_3     0.006021    0.083855  -0.004164        -0.041712
ankle_acc6_1      0.343229    0.144746   0.297939        -0.061476
ankle_acc6_2      0.218812    0.103893   0.211036        -0.053106
ankle_acc6_3      0.003066    0.099199  -0.009076        -0.042125
ankle_gyro1       0.001594   -0.004876   0.001228        -0.000417
ankle_gyro2      -0.045435    0.012900  -0.028440         0.033085
ankle_gyro3      -0.001125    0.001905   0.001550         0.001882
ankle_magne1     -0.308442   -0.199631  -0.325755         0.092230
ankle_magne2     -0.234251   -0.069626  -0.247339         0.112293
ankle_magne3     -0.041953    0.209897  -0.056973         0.154059
subject_id       -0.022949   -0.001891  -0.127993         0.154094
```

```
                  hand_acc16_1  hand_acc16_2  hand_acc16_3
hand_acc6_1  \
timestamp            -0.321940      0.039321     -0.215706    -
0.332616
activityID           -0.129905      0.038609     -0.116724    -
0.129179
heartrate            -0.298458      0.065728     -0.268278    -
0.307739
handTemperature       0.059130     -0.053288      0.077812
0.073200
hand_acc16_1          1.000000     -0.085483      0.257433
0.978695
hand_acc16_2         -0.085483      1.000000     -0.067639    -
0.070139
hand_acc16_3          0.257433     -0.067639      1.000000
0.254849
hand_acc6_1           0.978695     -0.070139      0.254849
1.000000
hand_acc6_2          -0.080322      0.945219     -0.067663    -
0.075857
hand_acc6_3           0.261275     -0.070711      0.964548
0.258160
hand_gyro1            0.020572      0.181804     -0.028435
0.026733
hand_gyro2           -0.094400     -0.007236     -0.023842    -
0.070609
hand_gyro3            0.030422      0.018463     -0.084107
0.026827
hand_magne1          -0.509092     -0.043630     -0.192428    -
0.513296
hand_magne2           0.055012     -0.430874      0.145735
0.054776
hand_magne3          -0.210788      0.103437     -0.506023    -
0.212629
chestTemperature     -0.161275     -0.054717     -0.030131    -
0.151743
chest_acc16_1         0.020788      0.039300     -0.102546
0.022197
chest_acc16_2        -0.424136      0.229222     -0.068293    -
0.422319
chest_acc16_3         0.436363     -0.120776      0.143100
0.439236
chest_acc6_1          0.019807      0.036088     -0.105307
0.018233
chest_acc6_2         -0.430063      0.243264     -0.068802    -
0.431257
chest_acc6_3          0.436456     -0.127795      0.142996
0.439120
chest_gyro1          -0.071869      0.060359      0.008060    -
```

```
0.056542
chest_gyro2         -0.048675      0.001573     -0.058523    -
0.040470
chest_gyro3          0.080158      0.033692      0.031157
0.074419
chest_magne1         0.248302     -0.052192      0.175672
0.248276
chest_magne2         0.340194     -0.088579      0.199710
0.342390
chest_magne3        -0.343251      0.055371     -0.136653    -
0.347580
ankleTemperature    -0.134254      0.074205     -0.024140    -
0.127711
ankle_acc16_1       -0.284793      0.048992     -0.158171    -
0.279327
ankle_acc16_2       -0.091788      0.112923     -0.069671    -
0.090862
ankle_acc16_3       -0.009622      0.008100      0.019459    -
0.009980
ankle_acc6_1        -0.320398      0.060863     -0.164375    -
0.314685
ankle_acc6_2        -0.101164      0.127373     -0.075457    -
0.100935
ankle_acc6_3        -0.013885      0.007775      0.023768    -
0.013800
ankle_gyro1          0.038499     -0.004623      0.012820
0.039194
ankle_gyro2         -0.072116     -0.031042     -0.011710    -
0.070733
ankle_gyro3          0.093352     -0.037152      0.023692
0.082080
ankle_magne1         0.084533     -0.049024      0.060863
0.085498
ankle_magne2         0.206595     -0.096015      0.108253
0.211722
ankle_magne3        -0.042602      0.041010     -0.011762    -
0.039436
subject_id          -0.030536     -0.281382     -0.002032    -
0.028407

                  hand_acc6_2  hand_acc6_3  ...  ankle_acc6_1
ankle_acc6_2  \
timestamp            0.036865    -0.226055  ...      0.343229
0.218812
activityID           0.042104    -0.118394  ...      0.144746
0.103893
heartrate            0.063252    -0.277938  ...      0.297939
0.211036
handTemperature     -0.048332     0.094127  ...     -0.061476     -
0.053106
```

```
hand_acc16_1        -0.080322     0.261275  ...     -0.320398     -
0.101164
hand_acc16_2         0.945219    -0.070711  ...      0.060863
0.127373
hand_acc16_3        -0.067663     0.964548  ...     -0.164375     -
0.075457
hand_acc6_1         -0.075857     0.258160  ...     -0.314685     -
0.100935
hand_acc6_2          1.000000    -0.057245  ...      0.057046
0.130811
hand_acc6_3         -0.057245     1.000000  ...     -0.170112     -
0.075637
hand_gyro1           0.127434    -0.018062  ...      0.024639
0.007832
hand_gyro2          -0.018114    -0.046556  ...      0.056241
0.011126
hand_gyro3           0.047140    -0.081414  ...     -0.029619
0.067769
hand_magne1         -0.042100    -0.193052  ...      0.183855
0.133716
hand_magne2         -0.448265     0.148578  ...     -0.177566     -
0.100271
hand_magne3          0.108379    -0.508750  ...      0.120442
0.101405
chestTemperature    -0.051863    -0.018037  ...      0.117682
0.063606
chest_acc16_1        0.040047    -0.100536  ...     -0.057189
0.034964
chest_acc16_2        0.225210    -0.071394  ...      0.350883
0.123827
chest_acc16_3       -0.126056     0.148525  ...     -0.351082     -
0.245606
chest_acc6_1         0.038100    -0.103556  ...     -0.056746
0.050902
chest_acc6_2         0.240070    -0.070809  ...      0.346028
0.127759
chest_acc6_3        -0.132640     0.147878  ...     -0.332360     -
0.252244
chest_gyro1          0.039675     0.002391  ...     -0.015835
0.033178
chest_gyro2          0.004696    -0.057784  ...      0.050787
0.080787
chest_gyro3          0.040766     0.034933  ...     -0.080334
0.067386
chest_magne1        -0.055658     0.177742  ...     -0.277185     -
0.172659
chest_magne2        -0.092798     0.203156  ...     -0.376887     -
0.206097
chest_magne3         0.058728    -0.142154  ...      0.274268
0.208268
```

```
ankleTemperature      0.077988    -0.014209  ...      0.103990
0.079747
ankle_acc16_1         0.046976    -0.162560  ...      0.865992
0.136564
ankle_acc16_2         0.113708    -0.071240  ...      0.173266
0.830114
ankle_acc16_3         0.009954     0.018408  ...      0.026540     -
0.131144
ankle_acc6_1          0.057046    -0.170112  ...      1.000000
0.160784
ankle_acc6_2          0.130811    -0.075637  ...      0.160784
1.000000
ankle_acc6_3          0.009622     0.021927  ...     -0.011388     -
0.149378
ankle_gyro1          -0.002519     0.012257  ...     -0.026946
0.119640
ankle_gyro2          -0.032122    -0.011419  ...      0.019751     -
0.067823
ankle_gyro3          -0.013802     0.025473  ...     -0.064276
0.080396
ankle_magne1         -0.052534     0.062323  ...     -0.214861     -
0.119502
ankle_magne2         -0.099627     0.112929  ...     -0.122585     -
0.055327
ankle_magne3          0.043655    -0.009647  ...      0.125014
0.104612
subject_id           -0.299075     0.003984  ...      0.000232     -
0.016663

                  ankle_acc6_3  ankle_gyro1  ankle_gyro2  ankle_gyro3
\
timestamp             0.003066     0.001594    -0.045435    -0.001125

activityID            0.099199    -0.004876     0.012900     0.001905

heartrate            -0.009076     0.001228    -0.028440     0.001550

handTemperature      -0.042125    -0.000417     0.033085     0.001882

hand_acc16_1         -0.013885     0.038499    -0.072116     0.093352

hand_acc16_2          0.007775    -0.004623    -0.031042    -0.037152

hand_acc16_3          0.023768     0.012820    -0.011710     0.023692

hand_acc6_1          -0.013800     0.039194    -0.070733     0.082080

hand_acc6_2           0.009622    -0.002519    -0.032122    -0.013802
```

hand_acc6_3	0.021927	0.012257	-0.011419	0.025473
hand_gyro1	-0.015512	0.029073	-0.034873	0.138108
hand_gyro2	-0.036083	0.049503	0.098756	-0.034400
hand_gyro3	0.040395	-0.049996	-0.017408	-0.168897
hand_magne1	0.077258	0.049010	-0.084133	0.055356
hand_magne2	-0.071410	0.072884	-0.017045	0.051156
hand_magne3	-0.025222	0.007752	-0.047867	0.011586
chestTemperature	-0.058572	-0.001082	-0.003040	0.002490
chest_acc16_1	-0.077985	0.069913	-0.042724	-0.042076
chest_acc16_2	-0.012490	-0.038563	0.027827	-0.110185
chest_acc16_3	-0.150507	0.012151	-0.022057	0.069666
chest_acc6_1	-0.085095	0.059285	-0.031758	-0.042266
chest_acc6_2	-0.008315	-0.043967	0.031758	-0.100427
chest_acc6_3	-0.148667	0.011869	-0.020204	0.063494
chest_gyro1	-0.017601	0.017133	-0.018961	-0.046690
chest_gyro2	-0.015170	0.125529	0.066578	-0.100307
chest_gyro3	0.007946	0.148984	-0.126513	0.274436
chest_magne1	-0.046764	0.004302	0.014061	0.000785
chest_magne2	-0.011048	-0.001941	0.019354	-0.002909
chest_magne3	0.157367	0.007541	0.004453	0.002670
ankleTemperature	0.109705	-0.011002	0.008862	0.005253
ankle_acc16_1	-0.127484	-0.005604	-0.049563	-0.036485
ankle_acc16_2	-0.100162	0.149641	-0.052329	0.141475
ankle_acc16_3	0.680237	-0.090419	-0.070966	-0.019770

ankle_acc6_1	-0.011388	-0.026946	0.019751	-0.064276
ankle_acc6_2	-0.149378	0.119640	-0.067823	0.080396
ankle_acc6_3	1.000000	-0.082210	0.011974	-0.015923
ankle_gyro1	-0.082210	1.000000	-0.066244	0.323950
ankle_gyro2	0.011974	-0.066244	1.000000	0.021793
ankle_gyro3	-0.015923	0.323950	0.021793	1.000000
ankle_magne1	-0.034605	-0.022029	0.021425	-0.005423
ankle_magne2	-0.065216	0.055971	-0.028698	0.010361
ankle_magne3	-0.024594	-0.016857	-0.025478	-0.017639
subject_id	-0.165449	0.015888	-0.006887	-0.004433

	ankle_magne1	ankle_magne2	ankle_magne3	subject_id
timestamp	-0.308442	-0.234251	-0.041953	-0.022949
activityID	-0.199631	-0.069626	0.209897	-0.001891
heartrate	-0.325755	-0.247339	-0.056973	-0.127993
handTemperature	0.092230	0.112293	0.154059	0.154094
hand_acc16_1	0.084533	0.206595	-0.042602	-0.030536
hand_acc16_2	-0.049024	-0.096015	0.041010	-0.281382
hand_acc16_3	0.060863	0.108253	-0.011762	-0.002032
hand_acc6_1	0.085498	0.211722	-0.039436	-0.028407
hand_acc6_2	-0.052534	-0.099627	0.043655	-0.299075
hand_acc6_3	0.062323	0.112929	-0.009647	0.003984
hand_gyro1	0.012532	-0.066293	-0.000633	-0.027199
hand_gyro2	-0.007820	-0.003256	-0.014572	0.004104
hand_gyro3	-0.062962	0.128163	0.036106	-0.001531

hand_magne1	-0.067752	-0.293229	-0.157725	0.050043
hand_magne2	0.237363	0.152785	-0.307834	0.307351
hand_magne3	0.164784	-0.204457	0.176941	0.031566
chestTemperature	0.003403	-0.023325	0.137156	0.250471
chest_acc16_1	0.031638	0.034772	0.022563	0.054456
chest_acc16_2	-0.095474	-0.158283	0.177911	0.055383
chest_acc16_3	0.241516	0.291592	-0.123356	0.160612
chest_acc6_1	0.029862	0.032687	0.025687	0.061794
chest_acc6_2	-0.096078	-0.163369	0.176566	0.054672
chest_acc6_3	0.241334	0.293496	-0.120431	0.168062
chest_gyro1	0.011577	-0.009814	-0.001900	-0.001165
chest_gyro2	-0.047100	0.071318	-0.007137	0.014272
chest_gyro3	0.036327	-0.013329	-0.016635	-0.012408
chest_magne1	0.173865	0.298685	-0.561354	0.047993
chest_magne2	0.544165	0.189275	-0.276592	0.084797
chest_magne3	-0.162404	-0.500480	-0.100202	-0.096166
ankleTemperature	-0.046966	-0.078861	0.092491	-0.215867
ankle_acc16_1	-0.206725	-0.113579	0.118556	0.002171
ankle_acc16_2	-0.111039	-0.051984	0.096681	-0.015104
ankle_acc16_3	-0.027784	-0.054434	-0.020188	-0.142385
ankle_acc6_1	-0.214861	-0.122585	0.125014	0.000232
ankle_acc6_2	-0.119502	-0.055327	0.104612	-0.016663
ankle_acc6_3	-0.034605	-0.065216	-0.024594	-0.165449
ankle_gyro1	-0.022029	0.055971	-0.016857	0.015888

```
ankle_gyro2            0.021425    -0.028698    -0.025478   -0.006887
ankle_gyro3           -0.005423     0.010361    -0.017639   -0.004433
ankle_magne1           1.000000     0.062097    -0.031615    0.193853
ankle_magne2           0.062097     1.000000     0.020617    0.105505
ankle_magne3          -0.031615     0.020617     1.000000    0.045997
subject_id             0.193853     0.105505     0.045997    1.000000

[43 rows x 43 columns]
```

```
X=pamdf[['chest_acc16_2','chest_magne3']]
tar = pamdf['heartrate']
```

During the exploratory data analysis phase, we computed the correlation between variables and determined that there is no linear relationship between them. As a result, we cannot use linear regression for modeling. Instead, we will use polynomial regression and random forest classification for our modeling approach.

Polynomial Regression

Polynomial regression is a type of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an nth degree polynomial. This is useful when the data does not fit a linear relationship.

In our case, the independent variable X includes chest_acc16_2 and chest_magne3, and the dependent variable tar is heart rate.

To fit a polynomial regression model, we first transform the independent variable X by calculating it to the nth degree polynomial, where n is set to 8. We then split the data into training and test sets using the train_test_split function. The model is fit using the X_train and y_train data, and the corresponding model is created.

To evaluate the performance of the model, we compute the root mean squared error and mean squared error. As an example, we also provide the predicted heart rate for two values of X from the dataframe."

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=8, include_bias=False)
poly_features = poly.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(poly_features,
tar, test_size=0.4,train_size=0.6,random_state=4798)
```

```
from sklearn.linear_model import LinearRegression
poly_reg_model = LinearRegression()
poly_reg_model.fit(X_train, y_train)
```

```
LinearRegression()
```

```
poly_reg_y_predicted = poly_reg_model.predict(X_test)
from sklearn.metrics import mean_squared_error
poly_reg_rmse = np.sqrt(mean_squared_error(y_test,
poly_reg_y_predicted))
print('The root mean squared error is {}'.format(poly_reg_rmse))
poly_mse=mean_squared_error(y_test, poly_reg_y_predicted)
print('The mean squared error is {}'.format(poly_mse))
poly_feat = poly.fit_transform([[1.930140,-54.463000]])
predict_heart= poly_reg_model.predict(poly_feat)
predict_heart
```

```
The root mean squared error is 20.62432356012397
The mean squared error is 425.36272231268464
```

```
array([92.5212244])
```

Random Forest Algorithm

The random forest algorithm is a machine learning method that can be used for both classification and regression tasks. It is called a "random forest" because it is made up of a collection of decision trees, which are generated randomly. The more trees in the forest, the more accurate the model's predictions will be.

To use the random forest algorithm, we first create a dataframe called data by dropping the activityID and timestamp columns. We also create a target variable that includes the activityID. We then split the data into training and test sets for both data and the target variable.

Next, we import the RandomForestClassifier and create a Gaussian classifier called clf. We train the model using the training data, and then evaluate its performance by calculating the root mean squared error and the accuracy of the model."

```
data=pamdf.drop(['activityID','timestamp'], axis=1)
target = pamdf['activityID']
train_df,test_df,train_target,test_target =
cross_validation.train_test_split(data,target,test_size=0.4,train_size
=0.6,random_state=12345)
```

```
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
```

```
clf.fit(train_df,train_target)

y_pred=clf.predict(test_df)

rand_rmse = np.sqrt(mean_squared_error(test_target, y_pred))
print('The root mean squared error for Random Forest Classifier is
{}'.format(rand_rmse))
```

```
The root mean squared error for Random Forest Classifier is
0.12921658888913773
```

```
from sklearn import metrics

print("Accuracy:",metrics.accuracy_score(test_target, y_pred))
```

```
Accuracy: 0.9998198543651218
```

```
# Testing with SubjectID

test=pamdf[pamdf['subject_id']==1]
test
```

```
        timestamp  activityID   heartrate  handTemperature
hand_acc16_1  \
0           37.66           1  100.000000          30.3750
2.21530
1           37.67           1  100.000000          30.3750
2.29196
2           37.68           1  100.000000          30.3750
2.29090
3           37.69           1  100.000000          30.3750
2.21800
4           37.70           1  100.000000          30.3750
2.30106
...           ...         ...         ...              ...            .
..
249952    3626.33          24  162.923077          30.1875
2.51550
249953    3626.34          24  156.230769          30.1875
2.50643
249954    3626.35          24  149.538462          30.1875
2.54102
249955    3626.36          24  142.846154          30.1875
2.65866
249956    3626.37          24  136.153846          30.1875
2.51044

        hand_acc16_2  hand_acc16_3  hand_acc6_1  hand_acc6_2
hand_acc6_3  \
0            8.27915       5.58753      2.24689      8.55387
5.77143
```

```
1             7.67288       5.74467      2.27373      8.14592
5.78739
2             7.14240       5.82342      2.26966      7.66268
5.78846
3             7.14365       5.89930      2.22177      7.25535
5.88000
4             7.25857       6.09259      2.20720      7.24042
5.95555
...               ...           ...          ...          ...
...
249952        7.02650       5.78869      2.44962      7.57075
5.77337
249953        6.30465       5.67552      2.53332      6.84517
5.69937
249954        5.84908       5.67758      2.55875      6.18058
5.80651
249955        5.88715       5.79468      2.57228      5.87855
5.92796
249956        6.11629       5.83017      2.58812      5.95396
5.94288

        ...  ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  ankle_gyro1  \
0       ...       9.63162      -1.76757      0.265761     0.002908
1       ...       9.58649      -1.75247      0.250816     0.020882
2       ...       9.60196      -1.73721      0.356632    -0.035392
3       ...       9.58674      -1.78264      0.311453    -0.032514
4       ...       9.64677      -1.75240      0.295902     0.001351
...     ...           ...           ...           ...          ...
249952  ...       9.61441       1.92197     -0.517913    -0.018273
249953  ...       9.71932       1.95217     -0.579109    -0.037873
249954  ...       9.83979       1.89179     -0.503878     0.059283
249955  ...       9.88503       1.84648     -0.458567     0.049326
249956  ...       9.89983       1.95229     -0.519527     0.051950

        ankle_gyro2  ankle_gyro3  ankle_magne1  ankle_magne2
ankle_magne3  \
0         -0.027714     0.001752      -61.1081      -36.8636      -
58.3696
1          0.000945     0.006007      -60.8916      -36.3197      -
58.3656
2         -0.052422    -0.004882      -60.3407      -35.7842      -
58.6119
3         -0.018844     0.026950      -60.7646      -37.1028      -
57.8799
4         -0.048878    -0.006328      -60.2040      -37.1225      -
57.8847
...             ...          ...           ...           ...
...
249952     0.000864     0.018507      -56.3324      -29.6397
34.1311
```

```
249953    -0.022418    -0.017999      -55.7786      -29.1093
34.2560
249954    -0.050002     0.005630      -55.5371      -29.5769
35.6106
249955    -0.016209     0.016162      -56.4328      -30.1943
34.6211
249956    -0.011002    -0.017655      -56.3426      -29.1916
33.6409

        subject_id
0                1
1                1
2                1
3                1
4                1
...            ...
249952           1
249953           1
249954           1
249955           1
249956           1

[249957 rows x 43 columns]
```

```
test=pamdf.iloc[[2932]]
test1=test.drop(['activityID','timestamp'], axis=1)
pred=clf.predict(test1)
pred
test1
```

```
       heartrate  handTemperature  hand_acc16_1  hand_acc16_2
hand_acc16_3  \
2932  100.363636          30.6875       6.05492       4.98862
5.88707

      hand_acc6_1  hand_acc6_2  hand_acc6_3  hand_gyro1
hand_gyro2  ...  \
2932      6.13627      4.96176      5.89612   -0.019734
0.018454  ...

      ankle_acc6_1  ankle_acc6_2  ankle_acc6_3  ankle_gyro1
ankle_gyro2  \
2932     -0.627138       -7.7219      -6.18387     0.005983    -
0.039797

      ankle_gyro3  ankle_magne1  ankle_magne2  ankle_magne3
subject_id
2932     0.020553      -14.5606       44.6709      -7.13134
1
```

```
[1 rows x 41 columns]
```

Summary:

The analysis of the Physical Activity Monitoring (PAM) dataset was carried out in a five-step process. To begin, we imported all the necessary libraries and data files. Next, we performed data cleaning by dropping unnecessary columns, removing null values through the use of the interpolate method, and converting all non-numeric data to numeric format. In the Exploratory Data Analysis (EDA) phase, we verified if the dataset was balanced and divided it into training and testing sets. The following analyses were conducted as part of the EDA:

- Examination of the amount of time each subject spent on different activities
- Investigation of the relationship between heartrate and different activities, including the use of bar charts to visualize the data
- Analysis of the correlation between calorie burn and activities
- Comparison of temperatures measured at the hand, chest, and ankle
- Use of heatmaps to visualize correlations between various data points.

Based on our observations, we carried out a z-test to compare heartrate and the activities with the highest intensity, specifically rope jumping and climbing stairs. Finally, we chose to use a polynomial regression model rather than a linear regression model as the exploratory data analysis revealed that there was no linear correlation between the variables. In addition, I also applied a random forest algorithm and achieved an accuracy of 0.99 in my predictions.

Based on the analyses and information discussed above, we can develop hardware or software that can identify the quantity and type of physical activity an individual engages in using heart rates.