# Exploratory Data Analysis of HAR

Table of Contents

[Exploratory Data Analysis of HAR 1](#_Toc130016909)

[Introduction: 2](#_Toc130016910)

[Data Cleaning (Pre-Processing): 4](#_Toc130016911)

[Normalization: 4](#_Toc130016912)

[Denoising: 5](#_Toc130016913)

[Up sampling & Down sampling: 5](#_Toc130016914)

[Descriptive Statistics (Feature Extraction): 5](#_Toc130016915)

[min 5](#_Toc130016916)

[max 6](#_Toc130016917)

[mean 6](#_Toc130016918)

[median 6](#_Toc130016919)

[first order mean 6](#_Toc130016920)

[second order mean 6](#_Toc130016921)

[variance 7](#_Toc130016922)

[standard deviation 7](#_Toc130016923)

[percentile 20 7](#_Toc130016924)

[percentile 50 8](#_Toc130016925)

[percentile 80 8](#_Toc130016926)

[skewness 8](#_Toc130016927)

[Data Visualization: 10](#_Toc130016928)

[SmartPhone Accelerometer 10](#_Toc130016929)

[SmartPhone Gyroscope: 11](#_Toc130016930)

[SmartPhone Gravity: 12](#_Toc130016931)

[SmartPhone Linear Acceleration: 13](#_Toc130016932)

[SmartPhone Magnetometer: 14](#_Toc130016933)

[SmartWatch Accelerometer: 14](#_Toc130016934)

[SmartWatch Gyroscope: 15](#_Toc130016935)

[SmartGlasses Accelerometer: 16](#_Toc130016936)

[SmartGlasses Gyroscope: 16](#_Toc130016937)

[Classification 17](#_Toc130016938)

[Support Vector Classifier: 17](#_Toc130016939)

[Random Forest Classifier: 19](#_Toc130016940)

## Introduction:

In this document, we will perform an exploratory data analysis (EDA) on a dataset of Human activity Recognition which contains a total of 55 activities and are performed 4572 times given in a range of sensors which include Mobile (Accelerometer, Gyroscope, Gravity, Linear Acceleration and Magnetometer), Smart Watch(Accelerometer, Gyroscope) and Jins Glasses(Accelerometer and Gyroscope).

These 55 activities are:



Since The Frequency of each sensor differs, we got different time windows for each sensor. Each activity is 4 seconds long. So for a 200 Hz frequency sensor, 1 activity window would be of size 800,3. In which 800 describes the data of 4 seconds and 3 describes the axis (x-axis, y-axis, z-axis).

The goal of this analysis is to gain insights and understanding of the data through descriptive statistics and data visualization.

Before proceeding with the analysis, we will discuss the context and background of the dataset and the research questions that we aim to answer. This will help provide a clear understanding of the dataset and the potential implications of the analysis.

We will also describe the data cleaning and preprocessing steps taken to ensure the data is ready for analysis. This includes removing missing or duplicate values, handling outliers, and transforming variables if necessary.

Overall, this EDA aims to provide a comprehensive and informative exploration of the dataset, laying the groundwork for further analysis and research.

## Data Cleaning (Pre-Processing):

### Normalization:

There are various methods by which we can perform normalization like:

* Min Max values normalization

In this method we subtract the minimum value from each data point and then divide by with difference between maximum and minimum values(range). As the result, it transforms the data in the range [0, 1].

* Normalization with Min Max Scalar

This method normalizes the data similarly to the min max values method but the only difference is that it allows the transformed data to be lied in a specified range.

* Normalization with sklearn library

By using the normalize function of sklearn library we can transforms our data on the same scale. This function uses Euclidean normalization which scales the data in such a way that sum of squares of the data points is equal to 1.

We have used the sklearn’s normalize function as it has given highest accuracy from other normalization methods.

#### Why do we use Normalization?

Since the data is in a different range, some axes range from 0 to 1 and some from 60-100. We have to do normalization to allow similar comparison/processing on each data.

Otherwise it would generate a bias result and incorrect model performance.

### Denoising:

It is the process of removing or reducing the unwanted data(noise) from the dataset. This unwanted data refers to any information that is been introduced during the acquisition or the storage of the data.

During preprocessing of the dataset, we also applied denoising but it did not have any effect on the accuracy of the classifier’s results so we removed it.

### Up sampling & Down sampling:

Up sampling is the process of increasing the number of samples in a signal or we can say that by up sampling, we increase the resolution of the dataset. Down sampling, on the other hand is the process of decreasing the number of samples in a dataset.

In the Cog Age dataset, we have data collected from three different sensors that is mobile, watch and glasses. All of these sensors have a different sampling rate. So, we tried and make the sampling rate of all sensors’ data to 100Hz.

For the mobile data we down sampled it from 800 to 400 and for the glasses and watch data, we had up sampled it to 400 so that all the data has an equal sample rate of 100Hz.

After applying the classifier on this dataset of equal sampling rate, we had observed that the accuracy was not affected by it. So, we decided to remove the up sampling and down sampling as well.

## Descriptive Statistics (Feature Extraction):

Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.

We calculated 17 features. Detail of them is given below.

### min

The feature "min" likely refers to the minimum value of some measurement.

In our project, this feature provides information about the lower bounds of the data we're working with. This features calculates min on axis=0 and provide minimum value from each axis i.e. x, y, z and return them as an array[].

### max

The feature "max" likely refers to the maximum value of some measurement.

In our project, this feature provides information about the upper bounds of the data we're working with. This feature calculates max on axis=0 and provide maximum value from each axis i.e. x, y, z and return them as an array[].

### mean

The feature "mean" likely refers to the average value of some measurement.

In our project, this feature provides information about the central tendency of the data we're working with. This feature calculates mean from each axis (x, y, z) and return them as an array[].

### median

The feature "median" likely refers to the median value of some measurement. The median is the middle value of a set of data, such that half of the values in the set are higher and half are lower.

In our project, this feature provides information about the central tendency of the data you're working with, similar to the mean feature.

### first order mean

The feature "first order mean" likely refers to the mean of the first-order difference of some measurement. In mathematical terms, the first-order difference of a time series is calculated by subtracting each value from the value that precedes it. The first-order mean is then the average of these differences.

This feature provides information about the rate of change in the data we're working with.

For example, if you're working with accelerometer data from a mobile device, the "first order mean" feature might provide information about the average rate of change in acceleration the device experienced during a certain time period. The accuracy of your machine learning model could be impacted by the "first order mean" feature because, if the feature is not representative of the rate of change in the data, it could lead to bias in your model

### second order mean

The feature "second order mean" likely refers to the mean of the second-order difference of some measurement. In mathematical terms, the second-order difference of a time series is calculated by subtracting the first-order difference of each value from the first-order difference that precedes it. The second-order mean is then the average of these differences.

This feature provides information about the rate of change in the rate of change of the data we're working with.

### variance

The feature "variance" likely refers to the variance of some measurement. Variance is a statistical measure of the spread of a set of data. It provides information about how much the values in the set deviate from the mean.

In our project, this feature provides information about the spread of the data we're working with. The accuracy of your machine learning model could be impacted by the "variance" feature because, if the feature is not representative of the spread of the data, it could lead to bias in your model. For example, if the "variance" feature is significantly different from the actual variance of the data, this could lead to incorrect assumptions about the distribution of the data and lower accuracy. Additionally, if the variance is too high or too low, it could lead to overfitting or underfitting, which can also negatively impact model accuracy.

### standard deviation

The feature "standard deviation" likely refers to the standard deviation of some measurement. Standard deviation is a statistical measure of the spread of a set of data, similar to variance. It is calculated as the square root of the variance and provides information about how much the values in the set deviate from the mean.

In our project, this feature could provide information about the spread of the data we're working with.

### percentile 20

The feature "percentile 20" likely refers to the 20th percentile of some measurement. The 20th percentile is a statistical value that separates the lowest 20% of the data from the rest of the data. In other words, 20% of the data points have values lower than the 20th percentile and 80% of the data points have values higher than it.

In our project, this feature could provide information about the distribution of the data we're working with.

For example, if you're working with accelerometer data from a mobile device, the "percentile 20" feature might provide information about the lowest 20% of the acceleration values the device experienced during a certain time period.

### percentile 50

The feature "percentile 50" likely refers to the 50th percentile, also known as the median, of some measurement. The median is a statistical value that separates the data into two equal halves, with 50% of the data points having values lower than the median and 50% of the data points having values higher than it.

### percentile 80

The feature "percentile 80" likely refers to the 80th percentile of some measurement. The 80th percentile is a statistical value that separates the highest 20% of the data from the rest of the data. In other words, 80% of the data points have values lower than the 80th percentile and 20% of the data points have values higher than it.

### skewness

The feature "skewness" refers to a measure of the asymmetry of a probability distribution of a real-valued random variable. In other words, skewness is a measure of how much a data distribution is "leaning" to one side or another. A positive skewness means that the tail of the distribution is extended to the right, while a negative skewness means that the tail of the distribution is extended to the left. A symmetrical distribution has a skewness of 0.

In our project, the skewness feature could provide important information about the shape of the data we're working with. For example, if you're working with accelerometer data from a mobile device, the skewness feature might provide information about the asymmetry of the acceleration values the device experienced during a certain time period.

#### kurtosis

The feature "kurtosis" refers to a measure of the "peakedness" of a probability distribution of a real-valued random variable. Kurtosis is a measure of how much a data distribution is peaked or flat relative to a normal distribution. A distribution with high kurtosis is called a "leptokurtic" distribution and has a high peak and thick tails, while a distribution with low kurtosis is called a "platykurtic" distribution and has a low peak and thin tails. A normal distribution has a kurtosis of 3.

#### zero crossing point

The feature "zero crossing point" refers to the number of times that a signal changes from positive to negative or vice versa. In other words, it is the number of times that a signal crosses the zero line. This feature is often used in signal processing and can be a useful indicator of the frequency content of a signal.

#### inter quartile

The feature "interquartile range (IQR)" is a measure of the variability of a dataset. It is defined as the range between the first quartile (25th percentile) and the third quartile (75th percentile) of the data. The IQR is used as a measure of dispersion in a dataset, as it gives an idea of how spread out the values are.

#### spectral enregy

The feature "spectral energy" refers to the distribution of energy across different frequencies in a signal. It is a measure of the amount of energy present in different frequency bands and is often used in signal processing and feature extraction.

In the context of our machine learning project, the spectral energy feature could provide important information about the frequency content of the sensors data we're working with. For example, if you're working with accelerometer data from a mobile device, the spectral energy feature might provide information about the distribution of energy across different frequencies in the acceleration signal. The accuracy of your machine learning model could be impacted by the spectral energy feature because, if the data has a lot of high frequency content, this could lead to noise in the data and lower accuracy. Additionally, if the spectral energy feature is not well understood, this could impact the ability to accurately train the machine learning model, leading to lower accuracy. It is important to consider the spectral energy feature when choosing and training machine learning models, as well as when interpreting the results of the models.

#### spectral entropy

The feature "spectral entropy" refers to a measure of the randomness or unpredictability of a signal's frequency content. It is used to quantify the diversity or spread of the energy across different frequencies in a signal.

In the context of our machine learning project, the spectral entropy feature could provide important information about the diversity of the frequency content in the sensors data you're working with. For example, if you're working with accelerometer data from a mobile device, the spectral entropy feature might provide information about how evenly the energy is distributed across different frequencies in the acceleration signal.

In our code, extract\_feature () take a 2D array((800,3) or (268,3) or (80,3)) of data and extract 17 features from it. Extend them in an array and return it. Thats how we get 17\*3 values from each array.

Moreover, calculateFeature () takes sensor data as an input. That could be any sensor like acceleromter, gyroscope etc. This function loop through each time window and extract feature of that particular activity. Now for each window we have features associate with it.

Next, we concatenate all the sensors data and pass it to model.

## Data Visualization:

For different devices(mobile, smart watch and glasses) we have different frequency rates associated with it. The Frequency of each sensor is written below.

SmartPhone:

Accelerometer : 200Hz

Gyroscope : 200Hz

Gravity : 200Hz

Linear Acceleration: 200Hz

Magnetometer : 50Hz

SmartWatch:

Accelerometer : 67Hz

Gyroscope : 67Hz

SmartGlasses:

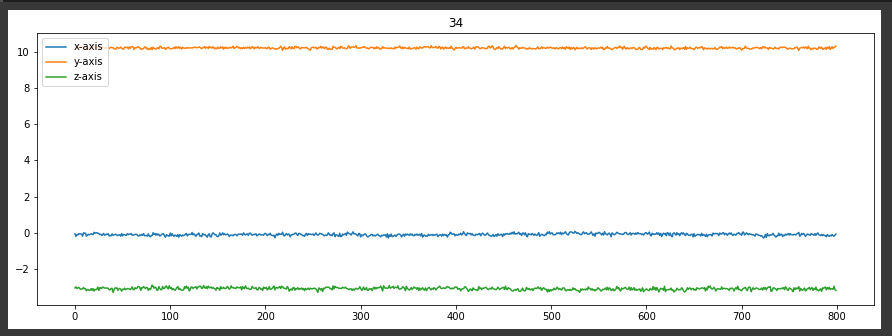
Accelerometer : 20Hz

Gyroscope : 20Hz

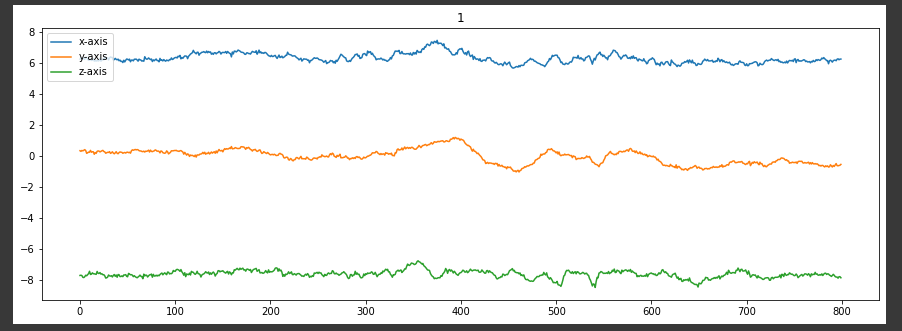
Since there exist so many activities, we show the difference between 2 different activities here.

### SmartPhone Accelerometer

Activity 34 (Read):

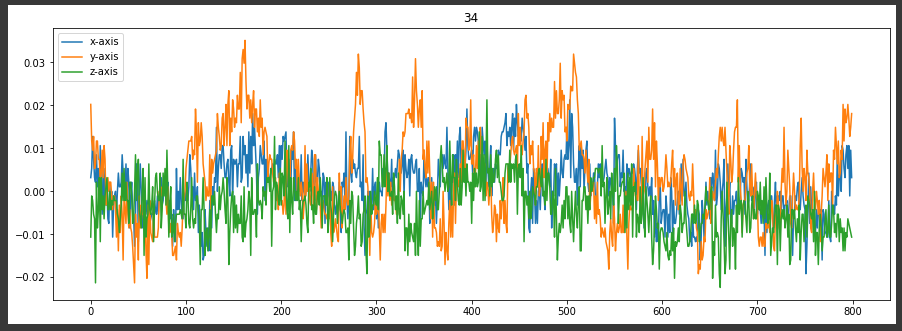


Activity 1 (Clean Floor):

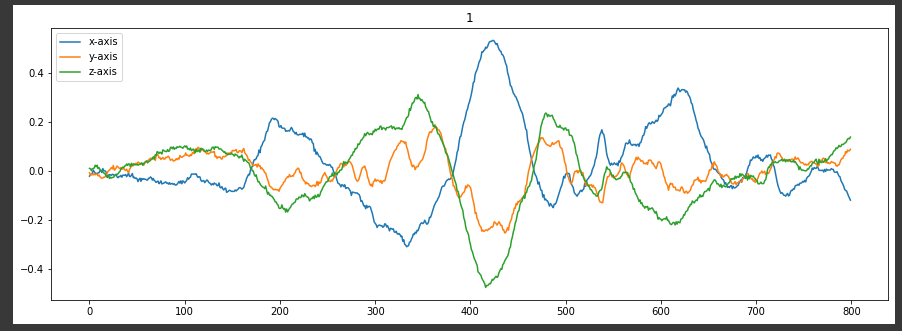


### SmartPhone Gyroscope:

Activity 34 (Read):

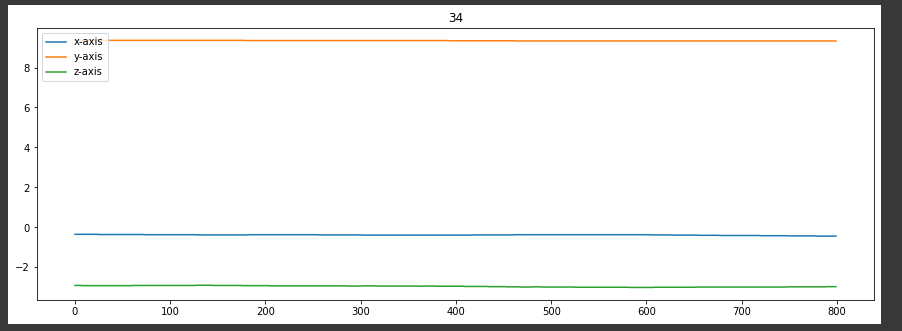


Activity 1 (Clean Floor):

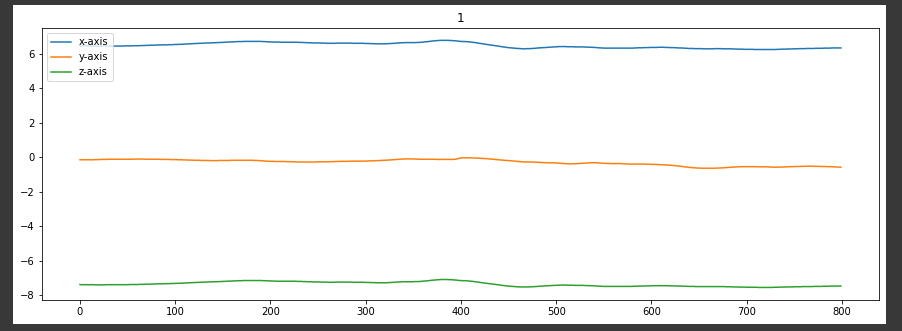


### SmartPhone Gravity:

Activity 34 (Read):

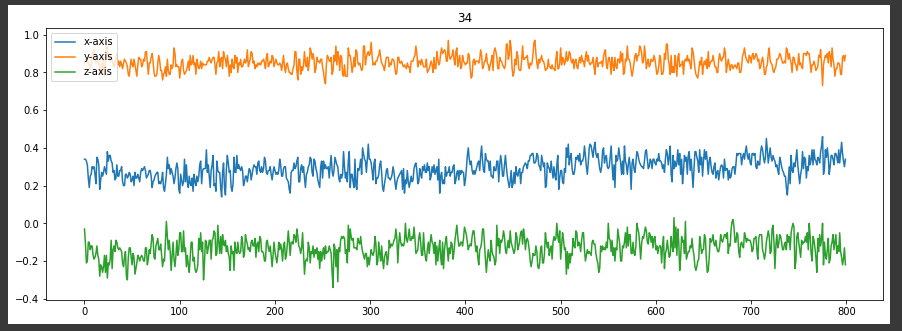


Activity 1 (Clean Floor):

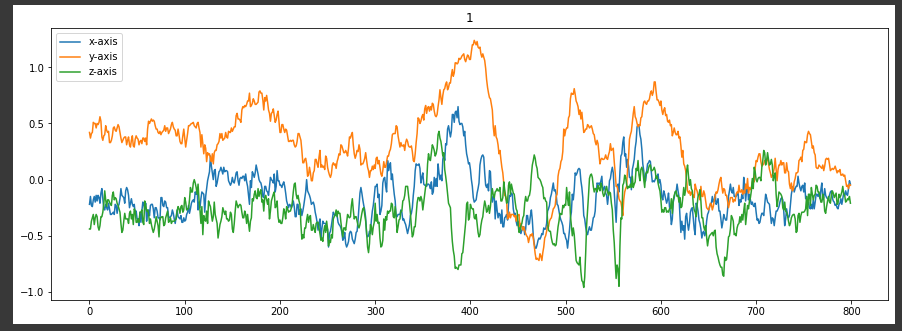


### SmartPhone Linear Acceleration:

Activity 34 (Read):

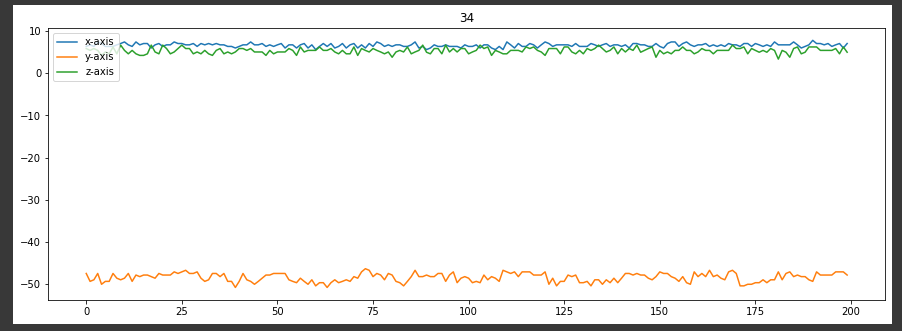


Activity 1 (Clean Floor):

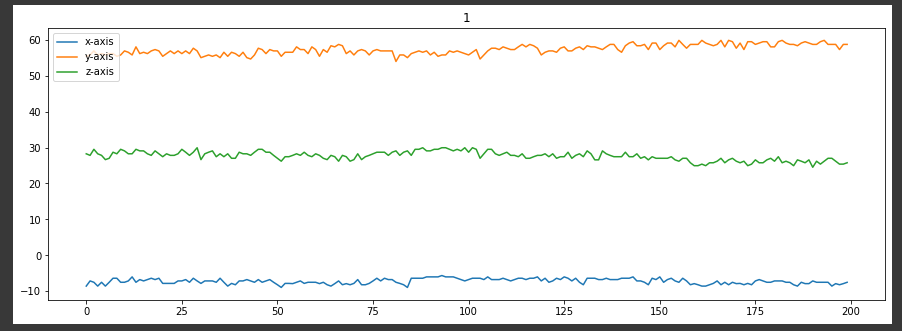


### SmartPhone Magnetometer:

Activity 34 (Read)

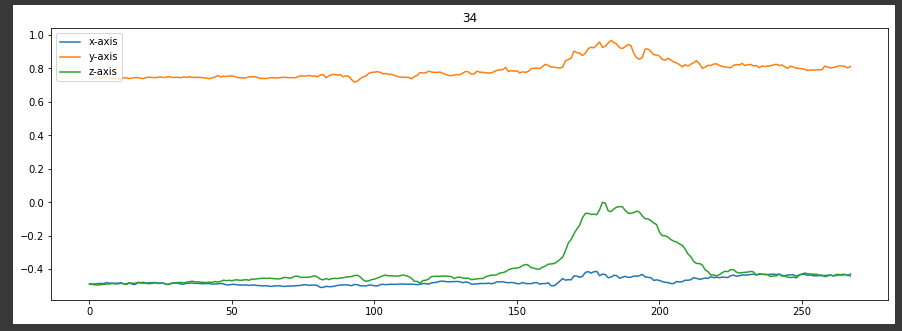


Activity 1 (Clean Floor):

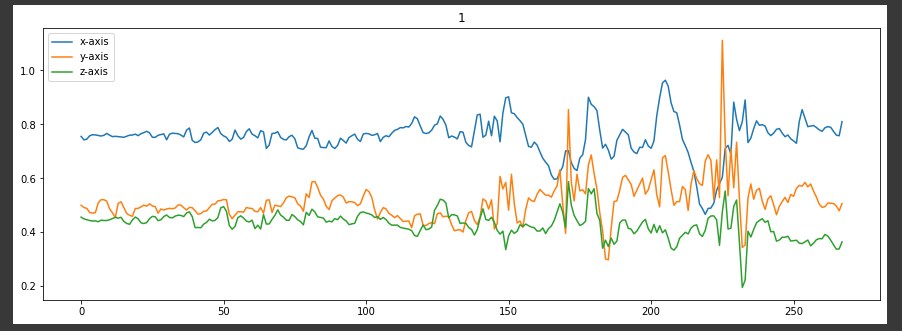


### SmartWatch Accelerometer:

Activity 34 (Read):

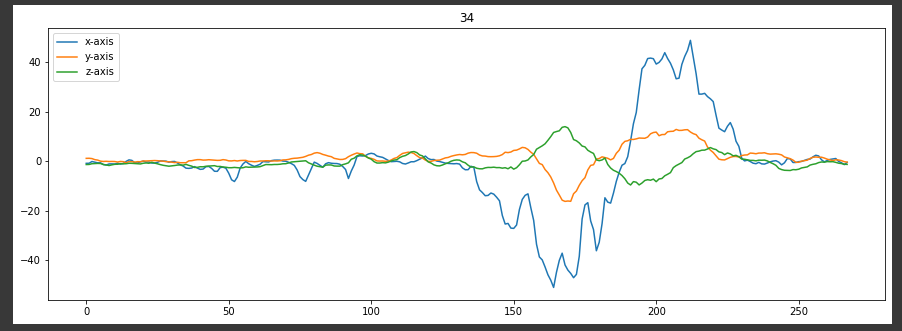


Activity 1 (Clean Floor):

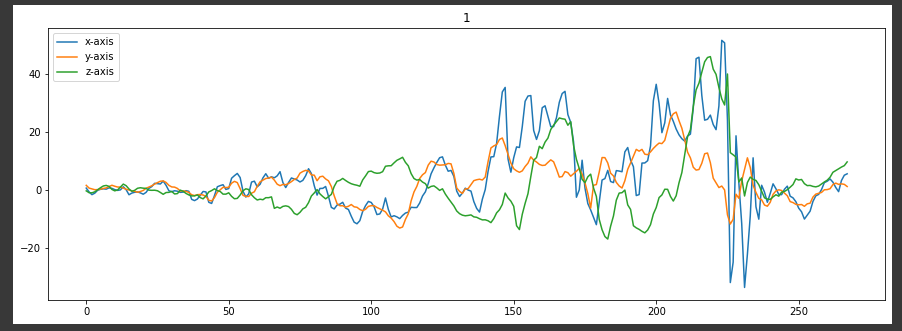


### SmartWatch Gyroscope:

Activity 34 (Read):

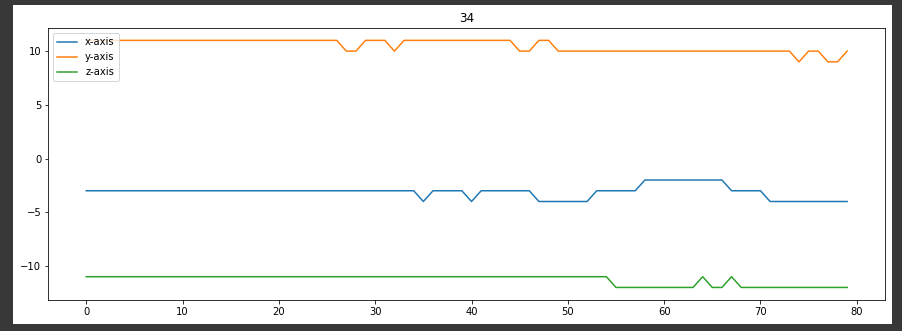


Activity 1 (Clean Floor):

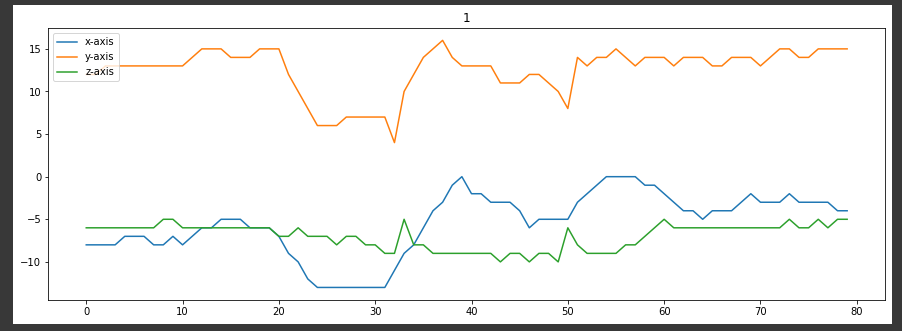


### SmartGlasses Accelerometer:

Activity 34 (Read):

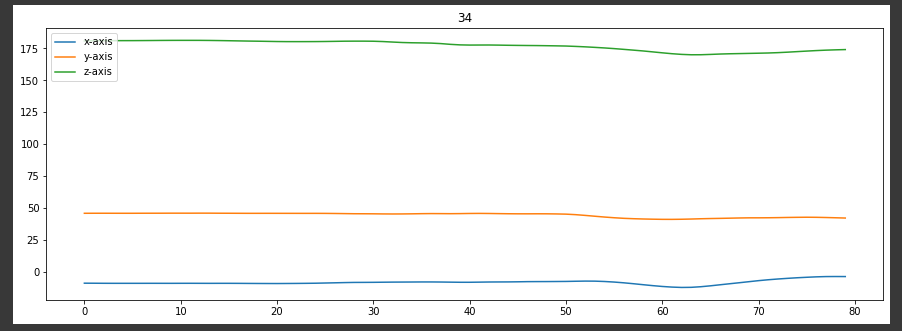


Activity 1 (Clean Floor):

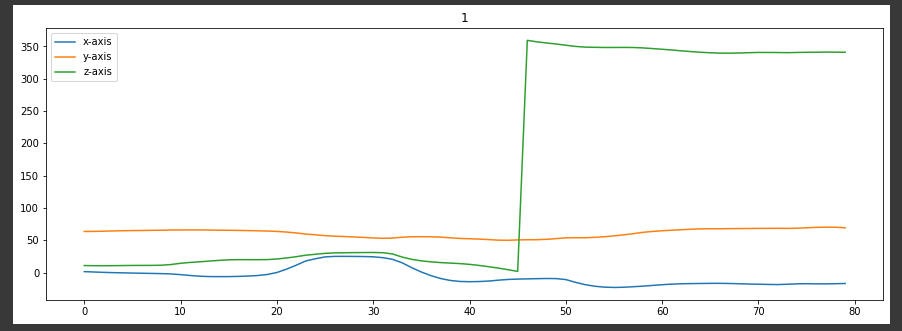


### SmartGlasses Gyroscope:

Activity 34 (Read):



Activity 1 (Clean Floor):



## Classification

We tried two classifiers respectively for training the model:

1. Support Vector Classifier (SVC)
2. Random Forest Classifier

### Support Vector Classifier:

It is used for classification tasks. It belongs to the family of Support Vector Machines (SVM), which are commonly used in supervised learning problems, particularly in binary classification tasks where the goal is to separate two classes of data using a hyperplane.

The idea behind the SVM/SVC algorithm is to find the hyperplane that best separates the two classes in the feature space, while maximizing the margin between them. The margin is defined as the distance between the hyperplane and the closest data points from each class, and the optimal hyperplane is the one that maximizes this distance.

To make a prediction for a new data point, the SVC algorithm computes the distance between the point and the hyperplane, and assigns it to the class that is on the other side of the hyperplane. The algorithm can be extended to handle multiclass classification problems by either training several binary classifiers and combining their predictions or by using a single classifier that separates all the classes at once.

One of the advantages of SVC over other classification algorithms is that it can handle non-linearly separable data by using a kernel function to map the data into a higher-dimensional space where the classes become separable. The most commonly used kernel functions are the linear, polynomial, and radial basis function (RBF) kernels.

SVC has been successfully applied in many real-world applications, such as image classification, text classification, and bioinformatics. However, it can be computationally expensive, especially when dealing with large datasets, and requires careful selection of hyperparameters such as the kernel type and its parameters.

### General Code Description:

* The SVM classifier is imported from the scikit-learn library and is initialized with a linear kernel and a regularization parameter C=1.
* The classifier is then trained on the provided training data, which consists of feature vectors and corresponding labels.
* After the classifier is trained, it's used to predict the labels of a test dataset.
* The performance of the classifier is evaluated using various metrics, including accuracy, weighted F1-score, macro F1-score, and individual F1-scores for each class.
* The accuracy is calculated as the ratio of correctly classified instances to the total number of instances.
* The F1-score is a measure of the balance between precision and recall and provides a single value to represent the performance of the classifier. The weighted F1-score is an average of the F1-scores weighted by the number of instances in each class, while the macro F1-score is the unweighted average of the F1-scores for each class.
* Finally, a confusion matrix is computed to provide more detailed information about the performance of the classifier.
* The rows of the matrix represent the true labels, while the columns represent the predicted labels. The values in each cell represent the number of instances with a given true and predicted label.

### Accuracy Details:

* SVM classifier with linear kernal achieved an accuracy of 23% using 12 HandCrafted Features, we use simple Min-Max Normalization in this case. (svm->min max norm->12 features->23%)
* By including 3 more Hand Crafted Features, we experimented SVM classifier (linear kernel) with Normalization using sklearn library and also segmented the data to improve the accuracy. As a result, accuracy was increased from 23% to 57%. (svm->sklearn norm->15 features-> small window-->57%)
* We introduce 2 more features (spectral entropy and enerygy) respectively, now having total of 17 features. Initially with accuracy of 57% we have highly segmented data, as a result SVM classifier ran indefinitely and didn't produce any results.
* However, after removing the segmentation and keeping the data to original size, SVM with sklearn normalization and 17 features give accuracy of 48%.
* SVM with non-linear kernal (rbf) give accuracy of 18% without denosing and accuracy of 28% with denoising.

### Random Forest Classifier:

It is used for classification tasks. It belongs to the family of ensemble learning methods, which combine multiple models to improve the overall performance and reduce overfitting.

The basic idea behind RFC is to create a large number of decision trees, where each tree is trained on a random subset of the training data and a random subset of the features. During training, each tree in the forest is constructed by recursively splitting the data into smaller subsets based on the values of the selected features, until a stopping criterion is reached.

To make a prediction for a new data point, the RFC algorithm aggregates the predictions of all the decision trees in the forest, either by taking the majority vote (for classification tasks) or by averaging the outputs (for regression tasks). This way, the ensemble model can better capture the underlying patterns in the data and reduce the impact of noisy or irrelevant features.

RFC has several advantages over other classification algorithms, such as being able to handle both categorical and numerical data, being less prone to overfitting, and providing a measure of feature importance that can help with feature selection. It has been successfully applied in many real-world applications, such as image and text classification, anomaly detection, and credit risk assessment.

However, RFC also has some limitations, such as being less interpretable than some other models, and being computationally expensive to train and evaluate, especially when dealing with large datasets or complex feature spaces.

### General Code Description:

* The Random Forest classifier is imported from the sklearn.ensemble library. The Random Forest Classifier is initialized with a maximum depth of 20 and random state 0.
* The max\_depth parameter of the RandomForestClassifier sets the maximum depth of the decision trees in the forest. A deeper tree is more complex and may result in overfitting, while a shallow tree may underfit the data. Setting the max\_depth to 20 means that the trees in the forest can have a maximum of 20 levels of splits. This value is set to 20 in this code to balance the model's complexity and accuracy. The test accuracy decreases from 68% by decreasing the max-depth and remains constant from 20 onward.
* The random\_state parameter is used to initialize the random number generator, which is used to select random samples of the data to train each tree in the forest. Setting the random\_state to 0 ensures that the same random samples are selected every time the code is run, making the results of the model reproducible.

### Accuracy Details with RF:

* With 17 features, Segmentation, denoising and Normalization using sklearn library the accuracy improved from 57% to 66%.
* After removing denoising factor from pre-processing, the accuracy jumped from 66% to 68%
* However, after testing different random states, our accuracy increase from 68% to 69% at random state = 440.