



Strength Exercise Tracking

Using
Machine Learning on Sensory Data

Under Guidance of
Prof. Prithish Varadwaj

Indian Institute of Information Technology Allahabad

Department of Information Technology

5th Semester - Mini Project

Group Members

Arnav Uppal (IIT2021006)
Tushar Singh (IIT2021032)
Chirag Nain (IIT2021079)

Nitish Kushwaha (IIT2021086)
Amit Kumar (IIT2021088)

Introduction

- ❖ **Context**
- ❖ **Problem Definition**
- ❖ **Output of Project**
- ❖ **Application**
- ❖ **Scope**
- ❖ **Challenges**

Introduction

❖ Context

- Quantified Self
- Current Situation– Cardio Based Tracking
- No Strength Exercises Based Tracking Product Exist.



Introduction

❖ Problem Definition

- To create Python scripts to process, visualize, and model accelerometer and gyroscope data to create a machine learning model that can classify strength exercises and count repetitions.



Introduction

❖ Output

- Given a series of sensor data we will have a model that can predict what kind of strength exercise are you doing and how many repetition have you done.
- In ideal situation , an application or backend would continuously monitor the accelerometer and gyroscope data and it would automatically track the exercise.

Introduction

❖ Application

Fitness Tracking and Goal Setting:

- Fitness tracking is possible with the model's workout recognition and repetition counting. Users can create and track fitness goals to ensure their strength training programs meet their aims.

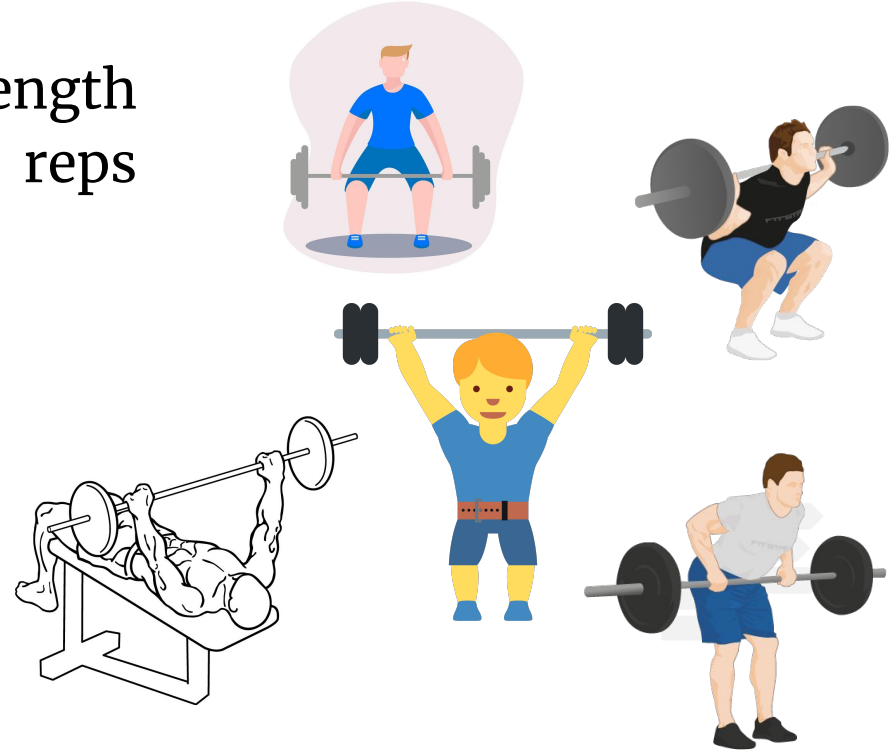
Context-Aware Applications:

- The broader scope involves the development of context-aware applications for strength training. The model lays the foundation for creating digital personal trainers that can adapt to users' workout contexts, providing tailored guidance and feedback.

Introduction

❖ Scope

- Can classify 5 different strength exercises and count the reps performed.
- Strength exercises are–
 - Bench Press
 - Squats
 - Bent over rows
 - Deadlift
 - Overhead press



Introduction



Limitation

- The model has been trained on relatively small dataset.
- Dedicated sensors not used for dataset generation.

Introduction

❖ Challenges Faced

- Identifying the right device for this type of problem.
- Difficulty in obtaining large dataset.
- Despite high accuracy , challenges were faced in distinguishing between exercises with similar movements such as bench press & overhead press or deadlift & row
- Further reducing miscount rate in counting reps

Related Works & Gaps

Study 1: Van Laerhoven (2000) – Activity Recognition Using Wearable Sensors

- Summary: This study, conducted by Van Laerhoven in 2000, was one of the earliest works on activity recognition using wearable sensors. The researchers connected accelerometers to a pair of pants and used machine learning techniques such as Kohonen maps and probabilistic models to recognize daily activities. The system they built was able to interpret raw sensor data and identify various activities.

Gaps:

- The details of the machine learning techniques used are not specified, making it challenging to understand the model's intricacies.
- The study's focus seems broad on recognizing daily activities, and it doesn't delve into specifics about exercise types or free weight training.
- The study's technological context might be outdated, and advancements in wearable technology could potentially impact the generalizability of the findings.

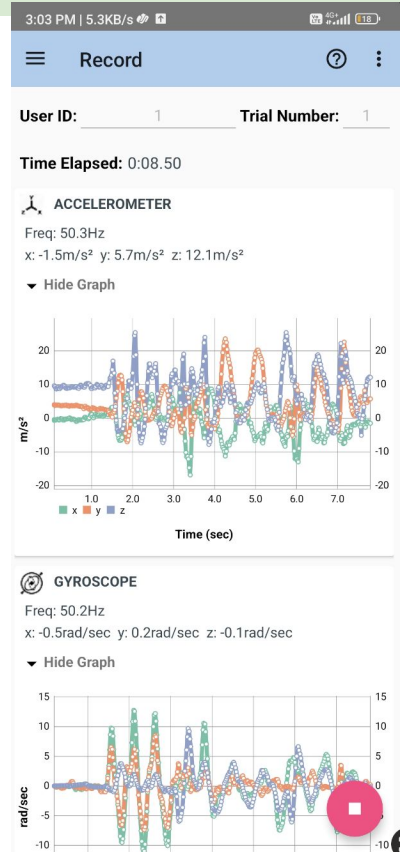
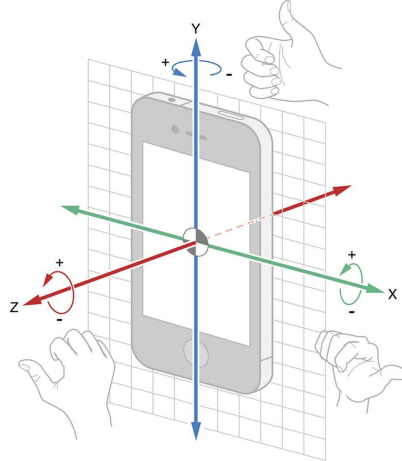
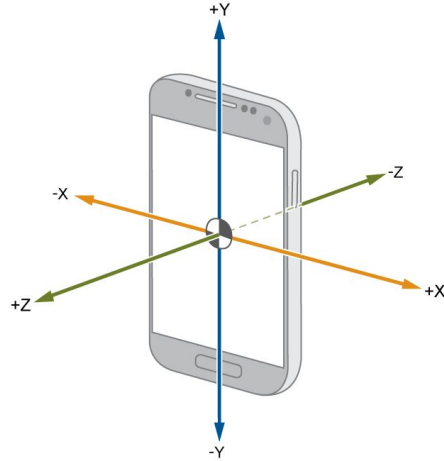
Related Works & Gaps

Study 2: Chang et al. (Year not specified) – Recognizing Exercises through Wearable Sensors

- Summary: Chang et al. utilized accelerometers in a workout glove and on a user's waist to track hand movements and body posture during gym exercises. They evaluated two methods, Naive Bayes Classifier, achieving an overall recognition accuracy of around 90% based on single-user data.
- **Gaps:**
 - The study focuses on recognizing exercises, but it doesn't address key aspects of strength training.
 - The study's evaluation involved a relatively small sample size, potentially limiting the generalizability of the results.
 - The miscount rate in the repetition counting algorithm is mentioned (around 7.5%), but the reasons behind miscounts or potential improvements are not discussed.

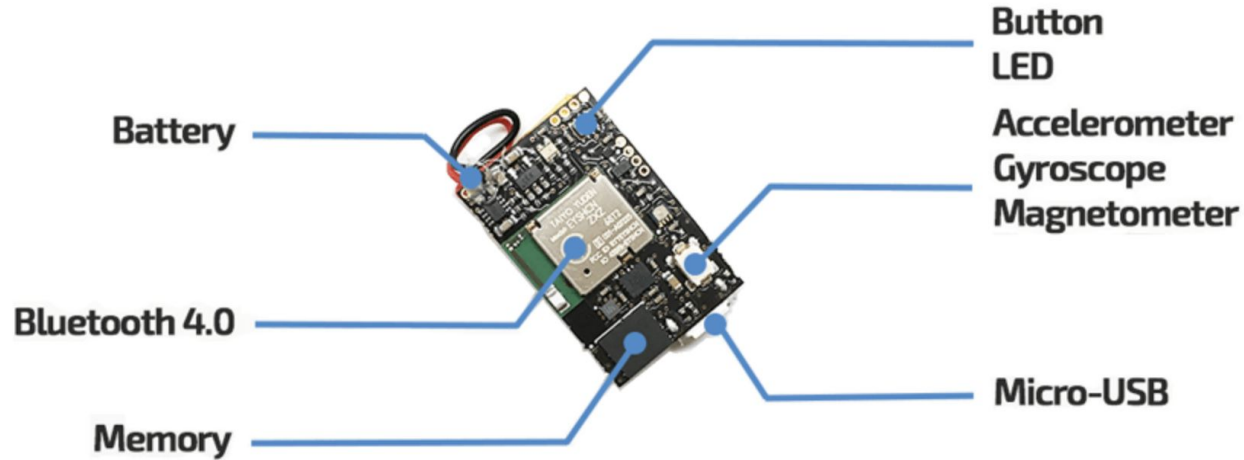
Experimental Setup

- 5 participants.
- To classify exercises we need data from 2 sensors.
- Data Sensor App for measuring and recording Gyroscope and Accelerometer data.



Experimental Setup

- Meta motion Sensor



































Data Acquisition

- Recording of Data.



Data Acquisition

- CSV files generated.

Name	Status
 A-bench-heavy_2023_Accelerometer.csv	
 A-bench-heavy_2023_Gyroscope.csv	
 A-bench-heavy2_2023_Accelerometer.csv	
 A-bench-heavy2_2023_Gyroscope.csv	
 A-bench-heavy2-rpe8_2023_Acceleromet...	
 A-bench-heavy2-rpe8_2023_Gyroscope.csv	
 A-bench-heavy3-rpe8_2023_Acceleromet...	
 A-bench-heavy3-rpe8_2023_Gyroscope.csv	
 A-dead-heavy_2023_Accelerometer.csv	
 A-dead-heavy_2023_Gyroscope.csv	
 A-dead-medium_2023_Accelerometer.c...	
 A-dead-medium_2023_Gyroscope.csv	
 A-dead-medium1_2023_Accelerometer.c...	
 A-dead-medium1_2023_Gyroscope.csv	
 A-dead-medium1-rpe6_2023_Accelerom...	
 A-dead-medium1-rpe6_2023_Gyroscope....	

epoch (ms)	time (01:00)	elapsed (s)	x-axis (g)	y-axis (g)	z-axis (g)
1697117221400	2023-10-12T13:27:01.400000	0.0	-0.181	0.796	-0.383
1697117221480	2023-10-12T13:27:01.480000	0.08	-0.192	0.823	-0.424
1697117221560	2023-10-12T13:27:01.560000	0.16	-0.203	0.808	-0.421
1697117221640	2023-10-12T13:27:01.640000	0.24	-0.217	0.833	-0.439
1697117221720	2023-10-12T13:27:01.720000	0.32	-0.206	0.845	-0.446
1697117221800	2023-10-12T13:27:01.800000	0.4	-0.209	0.867	-0.454
1697117221880	2023-10-12T13:27:01.880000	0.48	-0.247	1.047	-0.495
1697117221960	2023-10-12T13:27:01.960000	0.56	-0.359	1.422	-0.63
1697117222040	2023-10-12T13:27:02.040000	0.64	-0.334	1.267	-0.578
1697117222120	2023-10-12T13:27:02.120000	0.72	-0.254	0.86	-0.454
1697117222200	2023-10-12T13:27:02.200000	0.8	-0.269	0.789	-0.442
1697117222280	2023-10-12T13:27:02.280000	0.88	-0.292	0.804	-0.414
1697117222360	2023-10-12T13:27:02.360000	0.96	-0.282	0.811	-0.442
1697117222440	2023-10-12T13:27:02.440000	1.04	-0.277	0.833	-0.438
1697117222520	2023-10-12T13:27:02.520000	1.12	-0.262	0.862	-0.444
1697117222600	2023-10-12T13:27:02.600000	1.2	-0.256	0.882	-0.404

Data Preprocessing & Visualization

- All separated csv files merged together.
- Making label feature as exercise .
- Category feature - heavy/light (intensity)
- Participant feature- A/B/C/D/E
- Removing Time and time elapsed column, keeping only epoch value to track time using unix time.
- Set number.

Data Preprocessing & Visualization

epoch (ms)	time (01:00)	elapsed (s)	x-axis (g)	y-axis (g)	z-axis (g)
1697117221400	2023-10-12T13:27:01.400000	0.0	-0.181	0.796	-0.383
1697117221480	2023-10-12T13:27:01.480000	0.08	-0.192	0.823	-0.424
1697117221560	2023-10-12T13:27:01.560000	0.16	-0.203	0.808	-0.421
1697117221640	2023-10-12T13:27:01.640000	0.24	-0.217	0.833	-0.439
1697117221720	2023-10-12T13:27:01.720000	0.32	-0.206	0.845	-0.446
1697117221800	2023-10-12T13:27:01.800000	0.4	-0.209	0.867	-0.454
1697117221880	2023-10-12T13:27:01.880000	0.48	-0.247	1.047	-0.495
1697117221960	2023-10-12T13:27:01.960000	0.56	-0.359	1.422	-0.63
1697117222040	2023-10-12T13:27:02.040000	0.64	-0.334	1.267	-0.578
1697117222120	2023-10-12T13:27:02.120000	0.72	-0.254	0.86	-0.454
1697117222200	2023-10-12T13:27:02.200000	0.8	-0.269	0.789	-0.442
1697117222280	2023-10-12T13:27:02.280000	0.88	-0.292	0.804	-0.414
1697117222360	2023-10-12T13:27:02.360000	0.96	-0.282	0.811	-0.442
1697117222440	2023-10-12T13:27:02.440000	1.04	-0.277	0.833	-0.438
1697117222520	2023-10-12T13:27:02.520000	1.12	-0.262	0.862	-0.444
1697117222600	2023-10-12T13:27:02.600000	1.2	-0.256	0.882	-0.404



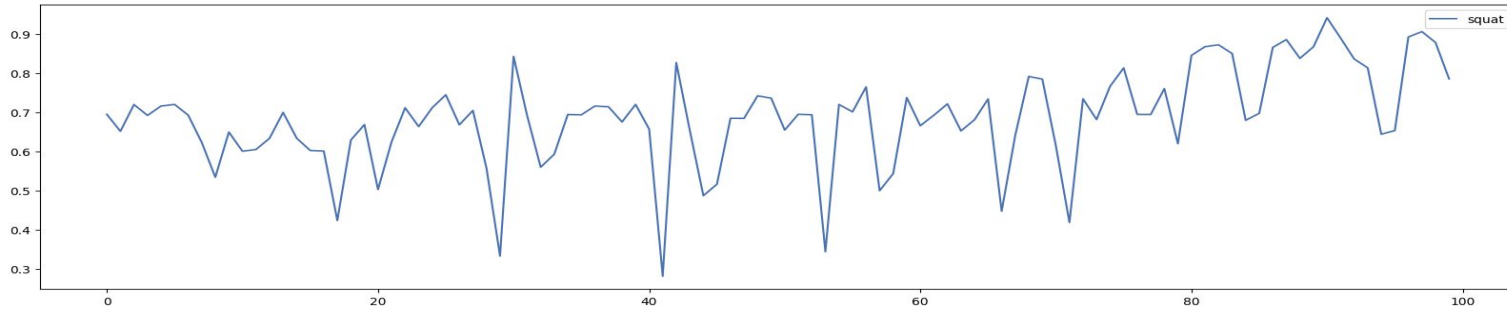
epoch (ms)	time (01:00)	elapsed (s)	x-axis (deg/s)	y-axis (deg/s)	z-axis (deg/s)
1697116969353	2023-10-12T13:22:49.353000	0.0	-2.866	-0.61	-5.366
1697116969393	2023-10-12T13:22:49.393000	0.04	-2.5	-0.61	-3.963
1697116969433	2023-10-12T13:22:49.433000	0.08	1.707	-3.902	1.524
1697116969473	2023-10-12T13:22:49.473000	0.12	2.439	-4.939	-4.207
1697116969513	2023-10-12T13:22:49.513000	0.16	7.5	-6.037	1.768
1697116969553	2023-10-12T13:22:49.553000	0.2	3.049	-4.207	-1.037
1697116969593	2023-10-12T13:22:49.593000	0.24	-2.561	0.976	-6.402
1697116969633	2023-10-12T13:22:49.633000	0.28	1.768	3.72	3.232
1697116969673	2023-10-12T13:22:49.673000	0.32	18.049	-2.256	8.902

epoch (ms)	acc_x	acc_y	acc_z	gyr_x	gyr_y	gyr_z	participant	label	category	set
2023-10-09 15:08:05.200	0.013500	0.977000	-0.071000	-1.890400	2.439200	0.938800	B	bench	heavy	29
2023-10-09 15:08:05.400	-0.001500	0.970500	-0.079500	-1.682600	-0.890400	2.170800	B	bench	heavy	29
2023-10-09 15:08:05.600	0.001333	0.971667	-0.064333	2.560800	-0.256000	-1.414600	B	bench	heavy	29
2023-10-09 15:08:05.800	-0.024000	0.957000	-0.073500	8.061000	-4.524400	-2.073000	B	bench	heavy	29
2023-10-09 15:08:06.000	-0.028000	0.957667	-0.115000	2.439000	-1.548600	-3.609800	B	bench	heavy	29

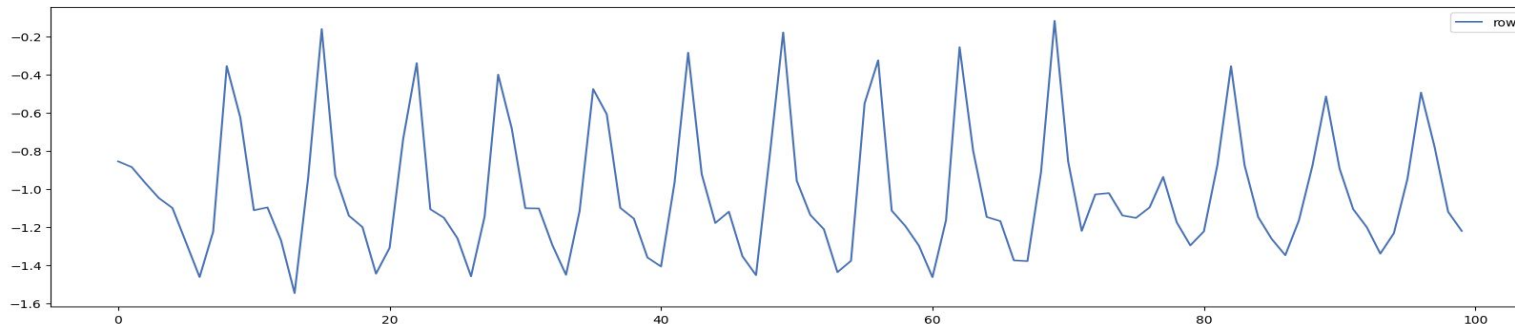
Data Preprocessing & Visualization

Acceleration Profile in Y-axis

❖ 1 Person - Different Exercises



Squat

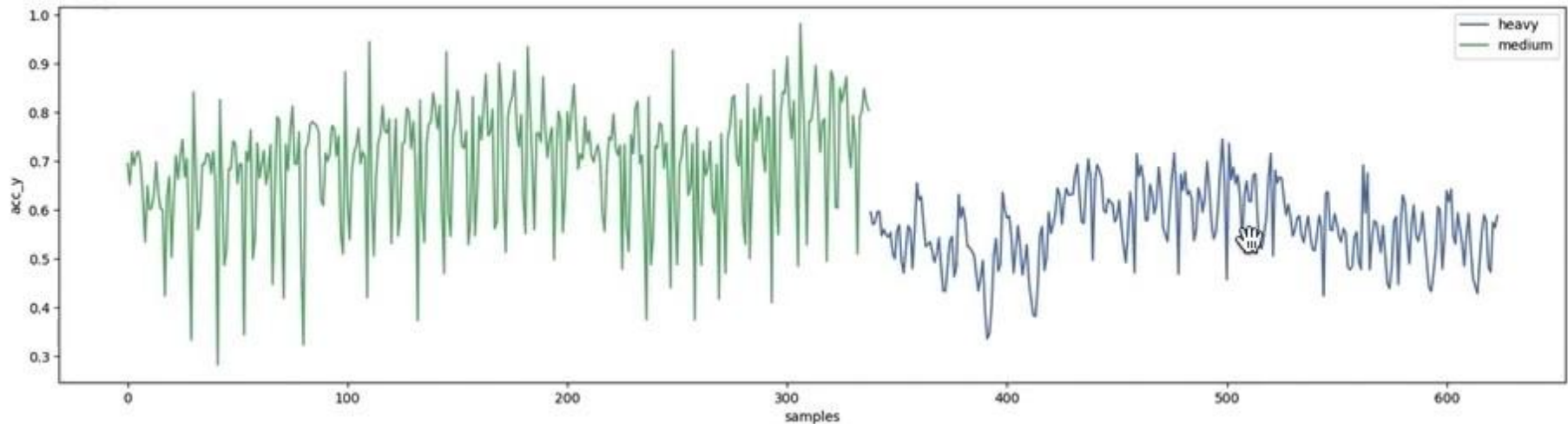


**Bent over
rows**

Data Preprocessing & Visualization

Acceleration Profile in Y-axis

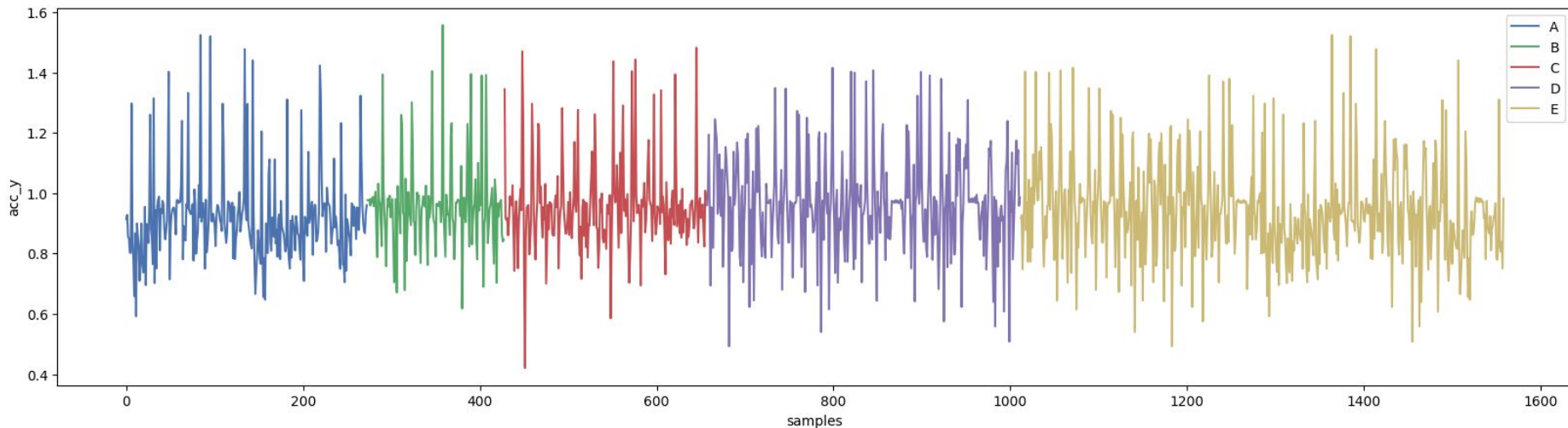
❖ 1 Person - 1 Exercise - Different Intensities/Weights (for squat visualisation)



Data Preprocessing & Visualization

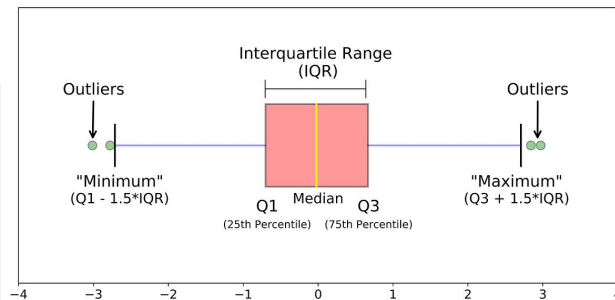
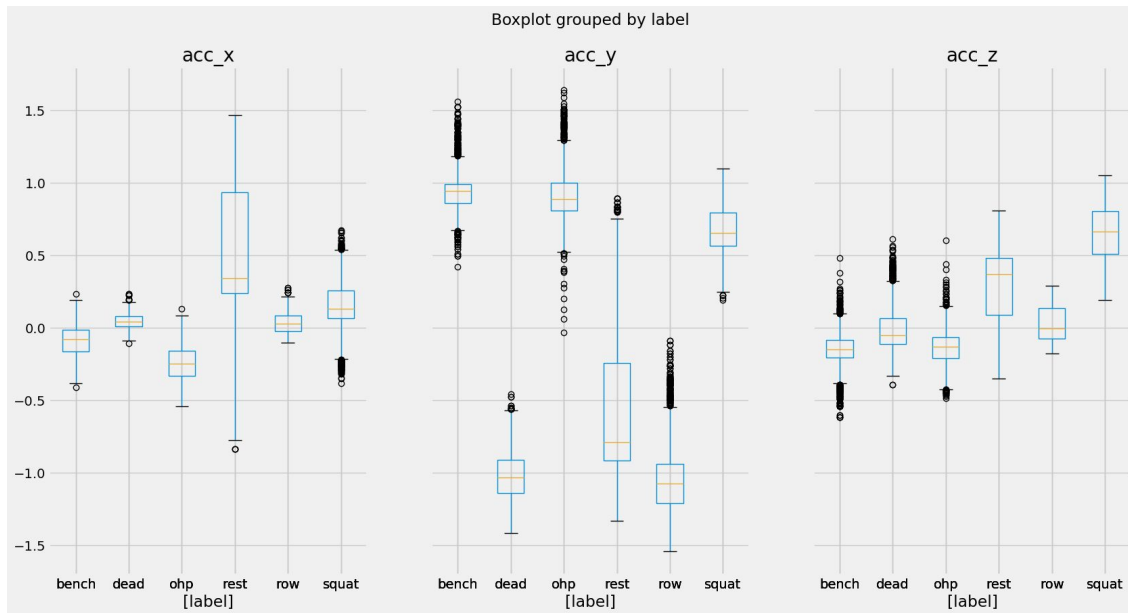
Acceleration Profile in Y-axis

❖ 5 Person - 1 Exercise (benchpress by all participants)



Data Preprocessing : Outliers

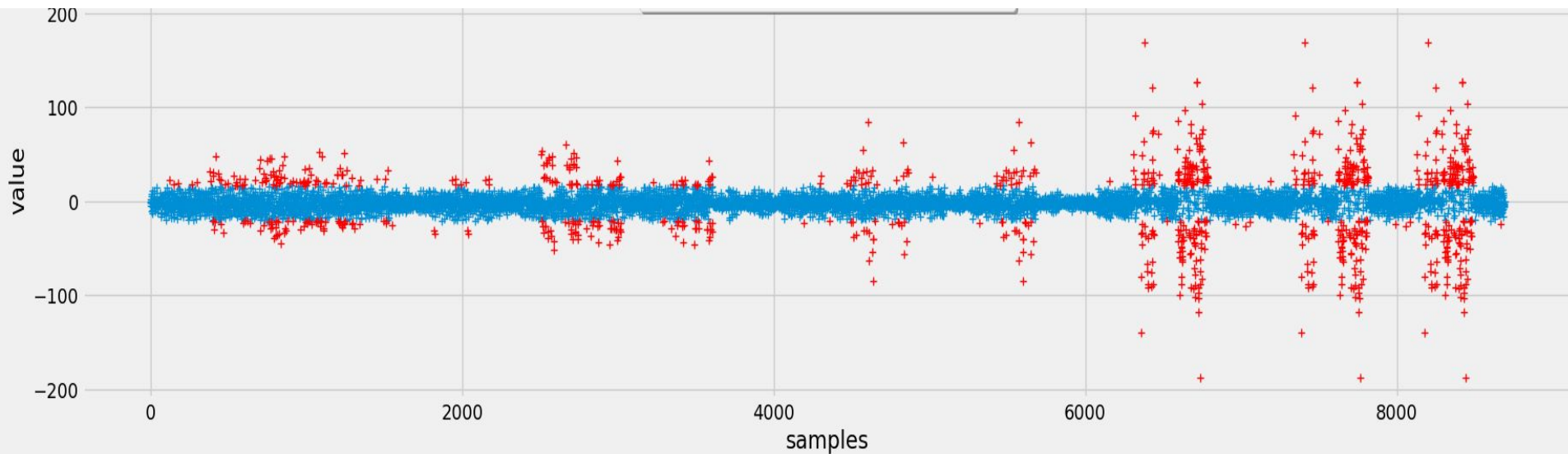
- First we applied **Boxplot** which uses interquartile range for outlier detection.



Data Preprocessing : Outliers

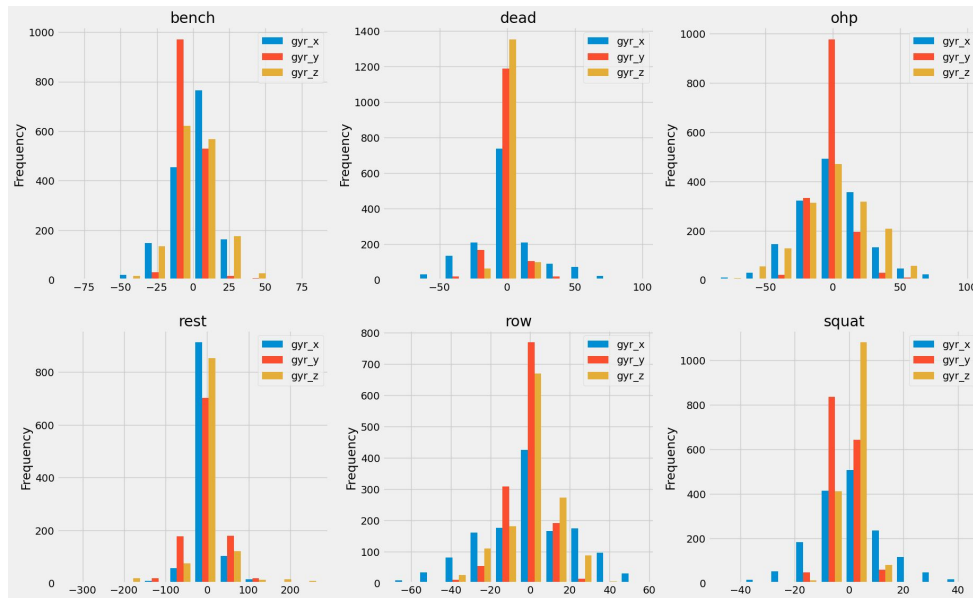
- Applying Boxplot on `gyr_y` column of bent row exercise.

- Red** - outliers
- Blue** - not outliers



Data Preprocessing : Outliers

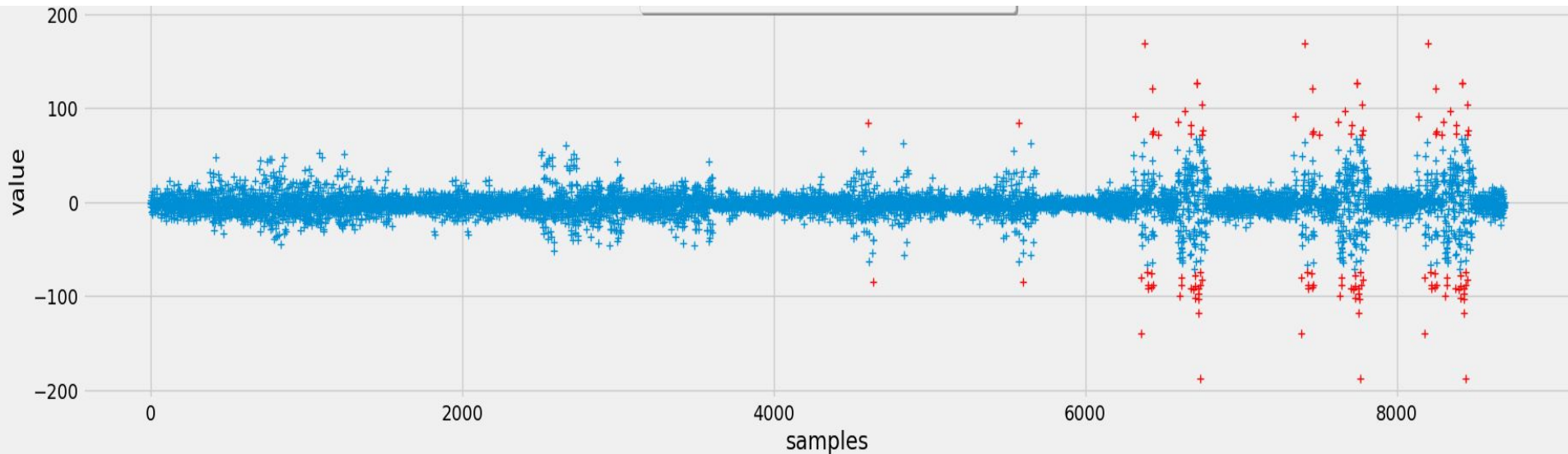
- **Chauvenet's Criterion** makes the assumption that the values in a dataset are normally distributed.
- Our dataset also follows the normal distribution.



Data Preprocessing : Outliers

- Applying Chauvenet's Criterion on `gyr_y` column of bent row exercise.

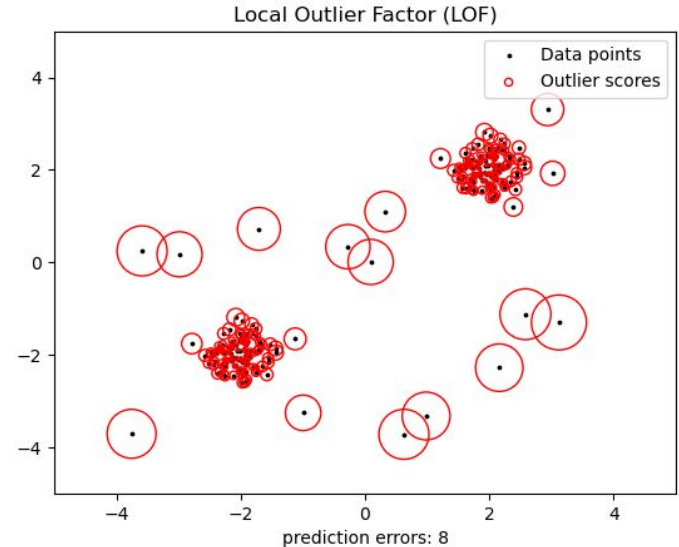
- Red** - outliers
- Blue** - not outliers



Data Preprocessing : Outliers

Local Outlier Factor:

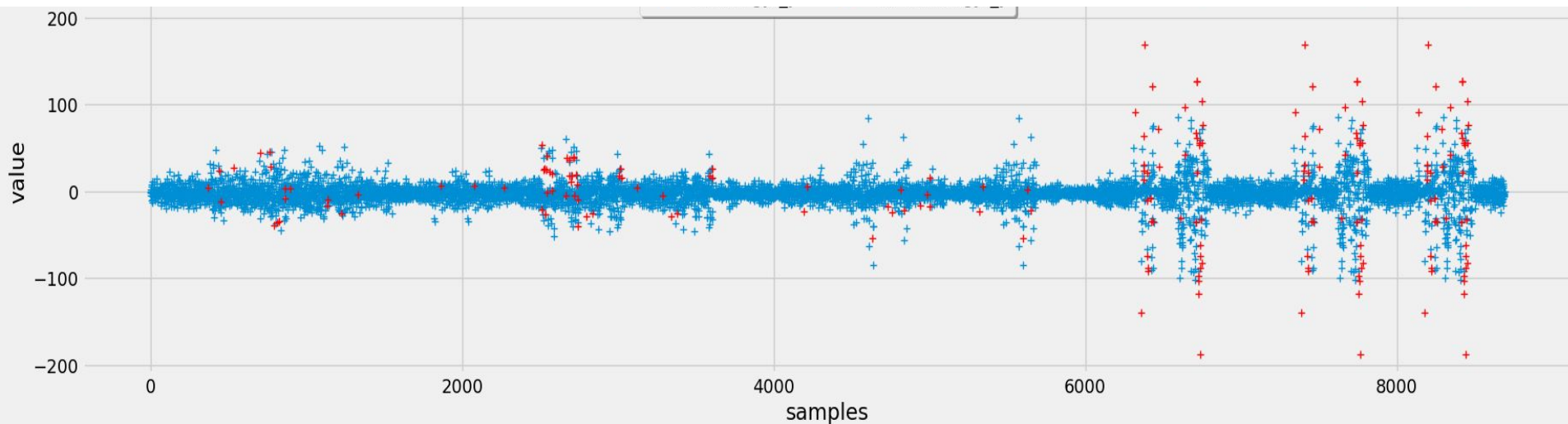
The LOF algorithm measures the local density deviation of a data point with respect to its neighbors. The basic idea is that outliers are data points that have significantly lower local density compared to their neighbors. LOF assigns a score to each data point, indicating its degree of "outlierness."



Data Preprocessing : Outliers

- Applying LOF on `gyr_y` column of bent row exercise.

- **Red** - outliers
- **Blue** - not outliers



Data Preprocessing : Outliers

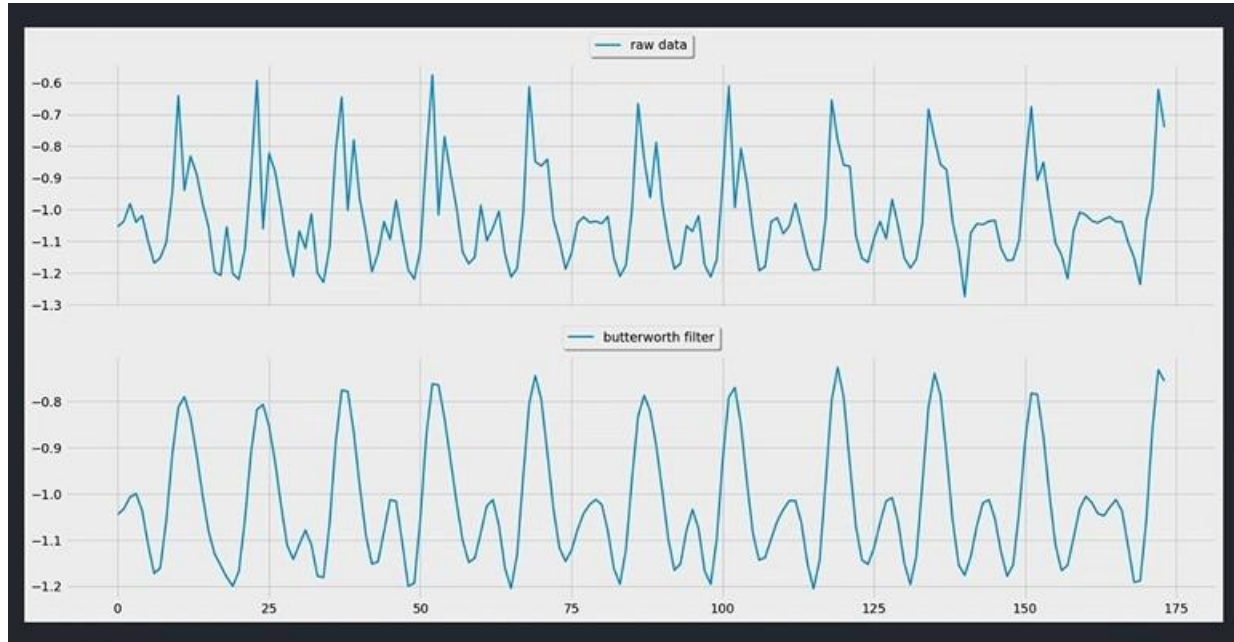
From above methods, Chauvenet's Criterion performed better than others and now from 8693 non null entries in each column, following non null entries are left.

```
✓ outliers_removed_df.info() ...  
  
<class 'pandas.core.frame.DataFrame'>  
DatetimeIndex: 8693 entries, 2023-10-09 15:08:05.200000  
Data columns (total 10 columns):  
#   Column      Non-Null Count  Dtype    
--  --      -  
0   acc_x       8691 non-null   float64  
1   acc_y       8682 non-null   float64  
2   acc_z       8685 non-null   float64  
3   gyr_x       8668 non-null   float64  
4   gyr_y       8625 non-null   float64  
5   gyr_z       8630 non-null   float64  
6   participant  8693 non-null   object  
7   label       8693 non-null   object  
8   category    8693 non-null   object  
9   set         8693 non-null   int32  
  
dtypes: float64(6), int32(1), object(3)  
memory usage: 713.1+ KB
```

After than we filled the missing values of outliers using **linear interpolation**.

Data Preprocessing : Low Pass Filter

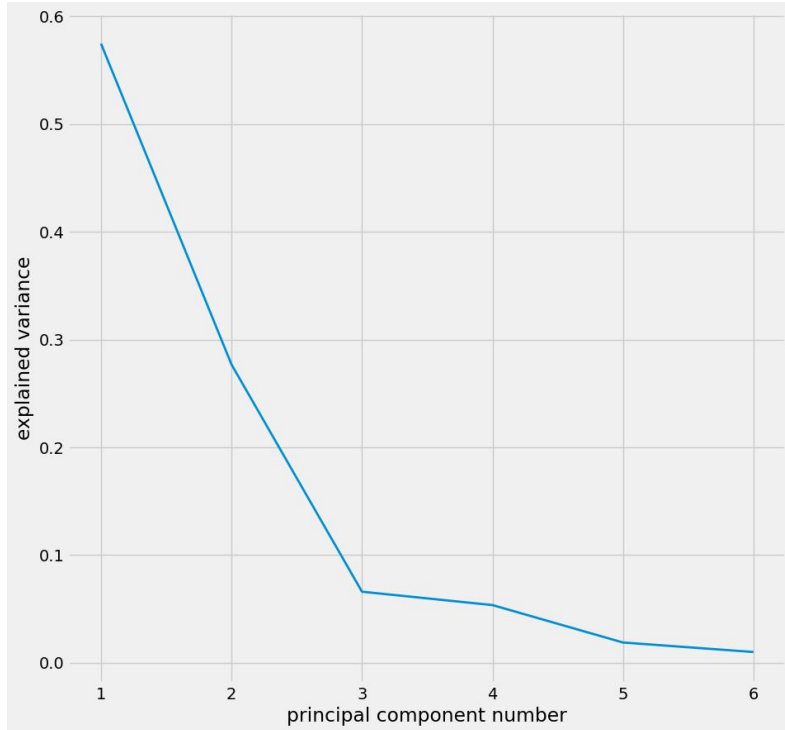
Used **Butterworth** Low pass filter on all columns to remove high frequency noise.



Data Preprocessing: Applying PCA

- We applied PCA to check if we can have good data variance with less number of features.
- We applied PCA on 6 features `acc_x`, `acc_y`, `acc_z`, `gyr_x`, `gyr_y`, `gyr_z`.
- After applying PCA on 6 features, we got 6 PCs so to decide relevant PCs we will plot PC vs Variance along that PC.

Data Preprocessing: Applying PCA



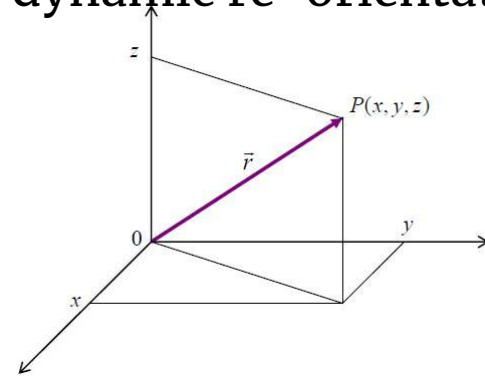
In this curve we can see that after 3rd PC, change in variance becomes very less, so we should choose 3 PCs for data representation.

Data Preprocessing: Sum of Squared Attribute

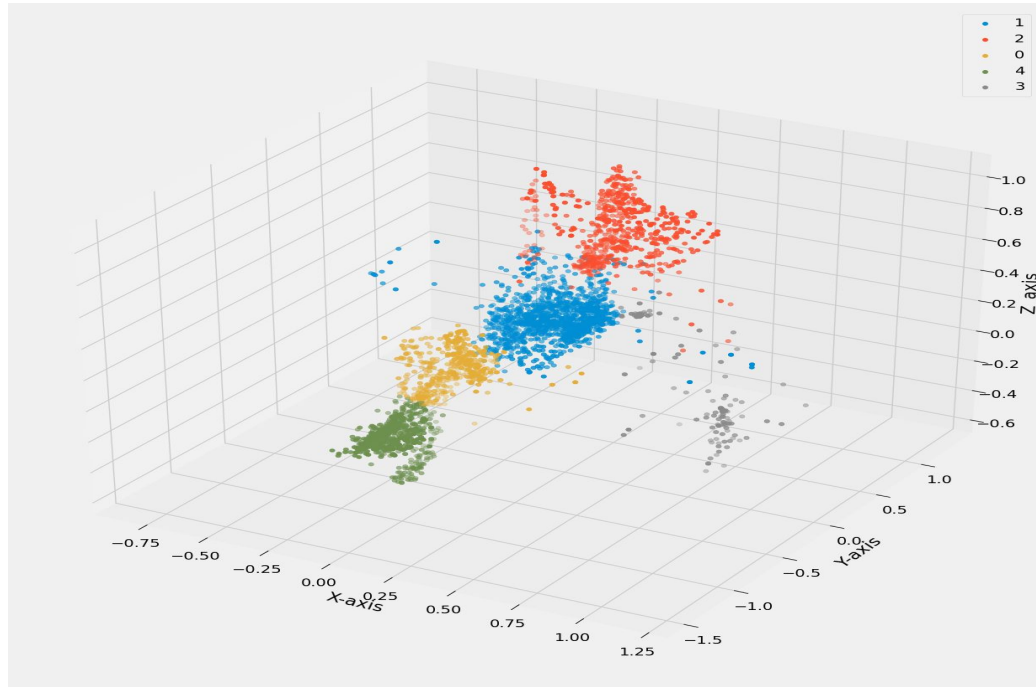
As we exercise the orientation of device can change which can severely affect the prediction. To avoid this we calculate r .

r is the scalar magnitude of the three combined data points: x , y , and z . The advantage of using r versus any particular data direction is that it is impartial to device orientation and can handle dynamic re-orientations.

$$r_{\text{magnitude}} = \sqrt{x^2 + y^2 + z^2}$$

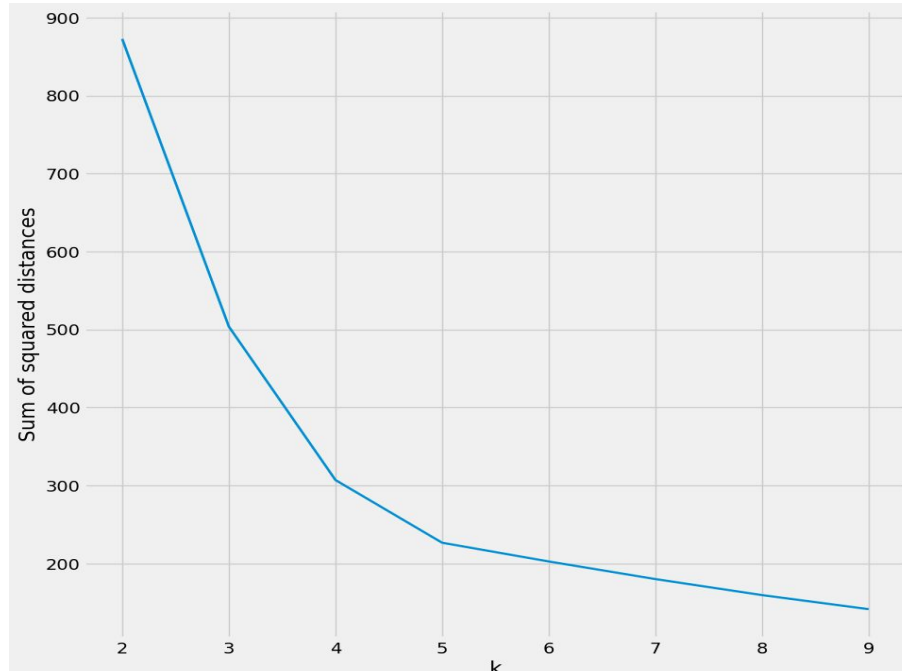


Data Preprocessing: K Means Clustering



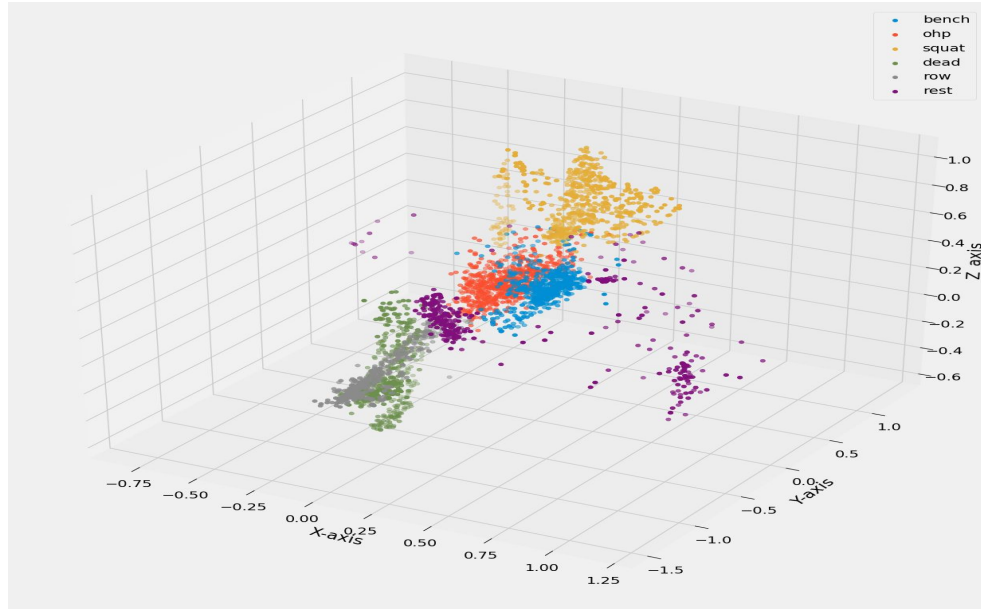
Data Preprocessing: K Means Clustering

How did we get the right number of clusters? -> Elbow method



Data Preprocessing: K Means Clustering

Number of clusters is nearly equal to the number of labels. So let label1=cluster1 , label2 = cluster2 and ... label6 =cluster6



Data Preprocessing: K Means Clustering

We can see that number of clusters also can be a strong predictor for identifying the exercise.

So add one more feature as Cluster

Note that we are taking clusters on the basis of acceleration only not on gyroscopic data as it can create different clusters which will not match the clusters formed by accelerometer and will cause decrease in accuracy.

Data Preprocessing: Final Data Outline

Basic features: acc_x, acc_y, acc_z , gyr_x, gyr_y, gyr_z

PCA features: PC1, PC2, PC3

Sum of squared attributes features: acc_r, gyr_r

Clustering feature: Cluster

Pass this dataset to model training

Methodology

```
feature_set_1 = list(set(basic_features))  
feature_set_2 = list(set(basic_features + square_features))  
feature_set_3 = list(set(feature_set_2 + pca_features))  
feature_set_4 = list(set(feature_set_3 + cluster_features))
```

We divide the feature sets as shown and train them with 4 algorithms–

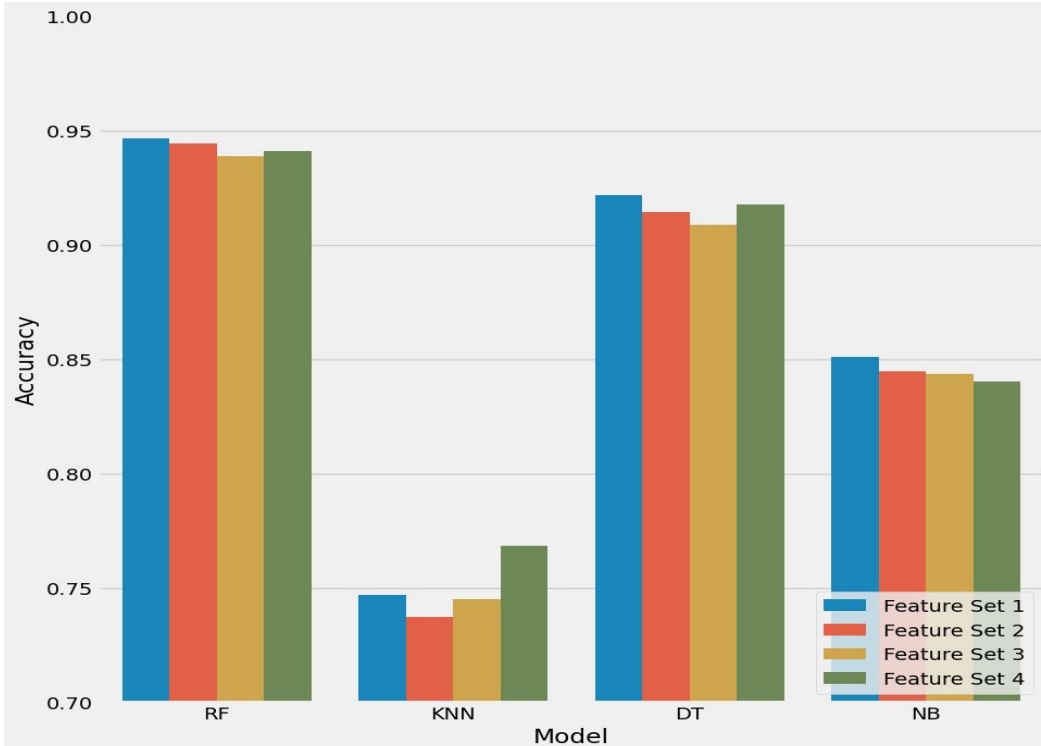
Random Forest

Decision Tree

KNN

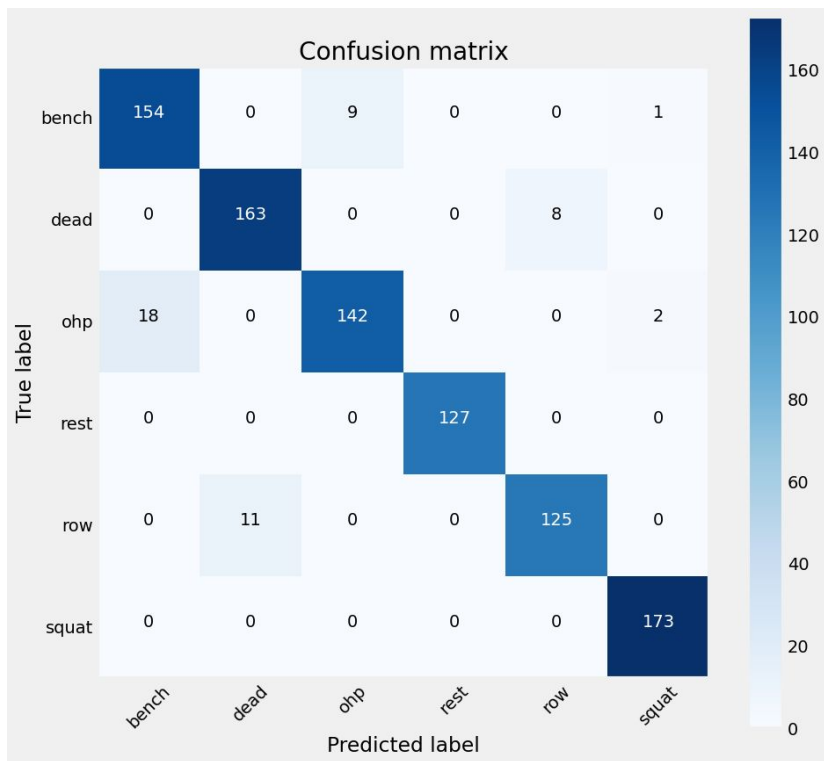
Naive Bayes

Methodology



Feature set 1 with Random Forest is giving highest accuracy

Methodology

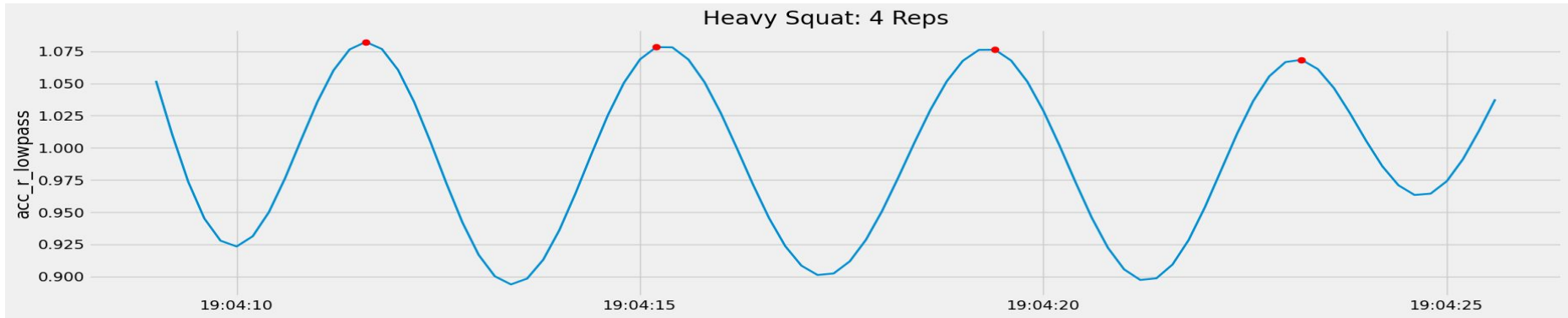


✓ accuracy ...

0.947481243301179

Counting Reps

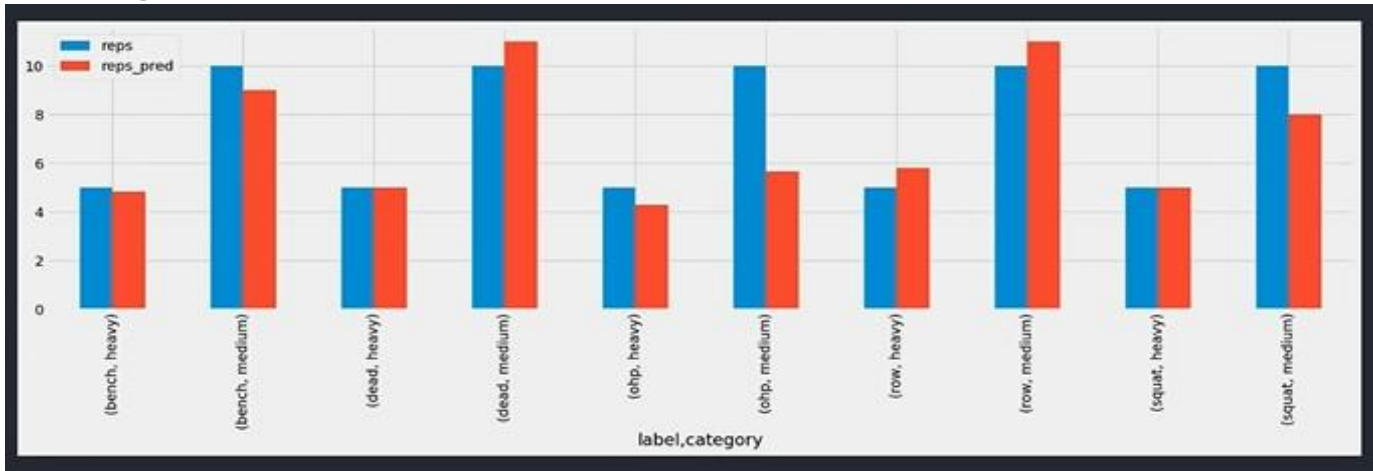
We plotted acc_r vs time graph after applying low pass butterworth filter on acc_r.



Here number of peaks will tell the number of repetitions.

Counting Reps

The original experiment was performed with every heavy set consisting 5 reps and light,medium set with 10 reps.



MAE-1.82 i.e. we are missing 1 or 2 reps on any set.

Conclusion

Success in Exercise Classification:

- The model demonstrated success in classifying strength training exercises using wearable sensor data and machine learning techniques. The Random Forest model achieved an impressive overall accuracy of 94.6%, showcasing the potential for accurate exercise recognition in real-world scenarios.

Conclusion

- The model was successful in implementing the effective peak counting algorithm for repetition counting, customized per exercise, achieving MAE=1.82.

Future Work

- To integrate the model in smart watch as an app
- Broaden the model by integrate more strength exercises
- We would like to work on a larger dataset so that it is better generalised for real life situations
- Providing tailored guidance and feedback to users based on their workout history.
- We are looking forward towards the possibility of the idea for a startup.

Thankyou