Sensor Based Activity Recognition

Deep Learning approach to classify human activity

Module:	Vertiefung Schreiben (VSR)
Author:	Navjot Zubler / Joël Grosjean
Instructor:	Marcel Messerli
Created on:	16/11/2022
FHNW Data Science/	HS22

Table of Contents

Introduction

Human Activity Recognition (HAR) is the identification of one or more human physical activities using sensor data. Wearables such as Apple/ Garmin smart watches use built-in sensors to count the number of steps or distance travelled. Smartphones have multiple build-in sensors, too, which can be used to detect events or gather data about human activities. There are a wide range of applications where these sensors come into play such as IoT topics, smart Home, Find your Phone etc.
Using classical Machine Learning (ML) algorithms such as decision trees, logistic regression and Deep Learning (DL) algorithms such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) it is possible to train models to classify a human activity.

Classification is one of the standard ML tasks, which is about mapping a class label to a given item. In this challenge, the task at hand is to assign an activity (*class label*) based on sensor input. It is a multi-class classification problem as the target class has four labels: Walking, running, cycling, and sitting.

The goal is to train different models (RNN, CNN, Decision Tree, and Logistic Regression) and compare their performance based on prediction accuracy.

- Which variables are key in classifying an activity? (feature importance)
- What effect does a specific data transformation have on the performance of ML algorithms?
- Does performance of ML algorithms vary when using different aggregation approaches on data points?
- Which aggregation on data points makes works best?

Methodology

Data collection

Logging device: Raw data was collected using iOS App 'SensorLog' [1] by user 1 & Android App 'AndroSensor' [2] by user 2.

Logging Rate: The logging rate of the used sensors was set to 100 Hz i.e. 100 data points per second. This option allows for further down-sampling, in case it is needed.

Activity logged: Each user logged four different activities with a 20 minute duration each: Walking, running, cycling and sitting (idle) with different device positions.

Device position: Each activity was logged outdoor with mobile device in the *left pant pocket* and with the following device orientations: *towards body*, *away from body*, *device upside down* or *device upright*.

Data format: Raw data was saved in the CSV format. In total, 31 data files were generated by recording each activity with device position in either of the four above by each user. The

meta-data on each of these raw CSV-files was documented for reference later. See Fig 1 in Appendix.

As two different apps were used to log data, different sensors are used (which are dependent on the physical device of the user) and the resulting raw data from different apps has different set of variables / features. An overview is available in Table 1:

	SensorLog App	**AndroSensor App**
User	User 1	User 2
Device	iPhone	Android
# of variables	~71 variables	~31 variables
Duration	20 mins	20 mins

Table 1: Overview - Apps used to log raw data

Data Exploration

A simple experiment was done in which one user performed a single activity and logged it using both smartphones (iPhone and Android device) at the same time while going for a short walk. The results from these two apps were then compared with help of plots and underlying characteristics per sensor such as units, magnitude, coordinates were compared. Additionally, explorative data analysis (EDA) was performed on the raw data for each activity and user by plotting violin plots, box plots & histograms.

Based on this experiment and plots, it was observed that certain sensor data from the Android device needs to be transformed to be aligned with same sensor data from the iPhone. There were no missing values observed in the logged sensor data.

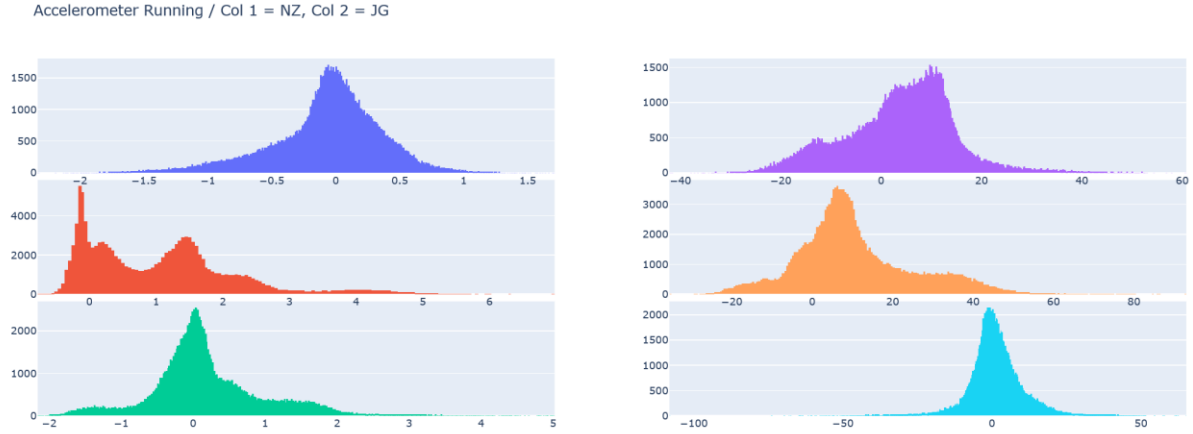


Fig 1: Histogram of Accelerometer sensor for user 1 in Col 1 and user 2 in Col 2

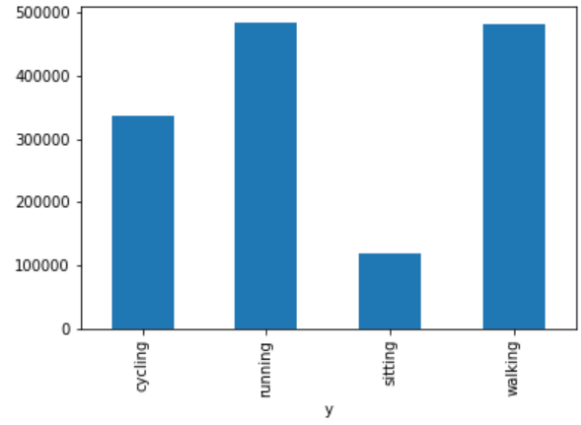


Fig 2a: Number of datapoints per activity – user 1

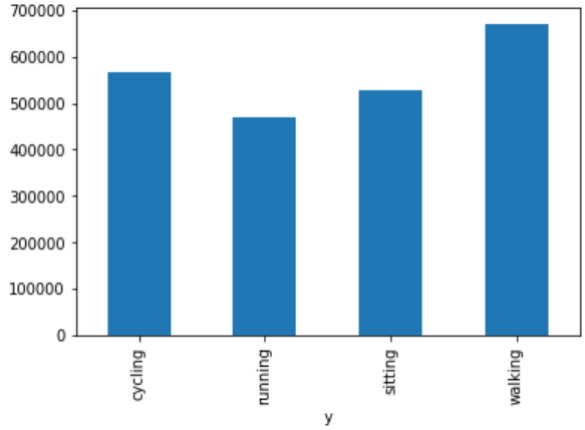


Fig 2b: Number of datapoints per activity – user 2

Data pre-processing

Each activity log using SensorLog/AndroSensor generated a Comma Separated value (CSV) file. Each row of this file represents one data point with a timestamp. Each file includes data from different sensors such as accelerometer, magnetometer, gyroscope, gravity and orientation sensors for each coordinate / direction (X, Y & Z).

1. Clean raw data:

- It was observed that a single file generated using SensorLog app was double the size as compared to a file generated by AndroSensor App. This was due to the fact that SensorLog data includes ~70 columns/variables in csv file and each data value had a precision of 12 decimal point. In order to keep the files to a size which can be uploaded to GitHub, columns which are not present in both files were removed, and all values were rounded to 6 decimal places.
- There are always few seconds in the beginning and at the end of each activity when the logging has already started but the activity has not started or has already stopped i.e. when user starts the App on device, putting the device in pant pocket, start the actual activity and when finished with the activity, take the device out and stop the logging. Therefore, a few seconds of data were ignored from the beginning and end of the logged data to avoid any unnecessary noise in the signal.

2. Data handling:

- Converting date times from string format to datetime.
- Renaming columns to align the names in all files generated using both Apps.
- Since all sensors record data for the X,Y and Z coordinates, vector magnitudes were created for every sensor. This has the benefit of making the data orientation independent. They were calculated using the following formula:

$$sensorA_{mag} = \sqrt{(sensorA_x^2 + sensorA_y^2 + sensorA_z^2)}$$

- Absolute values of minimum and maximum of the three directions ($sensor_x$, $sensor_y$, $sensor_z$) were created as well.
- GPS data such as latitude, longitude and altitude were not considered since Android App didn't always log them properly.

3. Data transformations:

- Additionally, certain sensor data from the AndroSensor app was transformed to align it with corresponding sensor data from SensorLog app. Details on these transformations are captured in table 2.

Sensor	SensorLog (iPhone)	AndroSensor (Android)	Transformation
Accelerometer	Measures in G i.e. value of 1 G corresponds to earth's gravity of 9.8 $\frac{m}{s^2}$	Measures in unit $\frac{m}{s^2}$	Accelerometer & Gravity sensor data for AndroSensor transformed to G by dividing by -9.80665
Gravity	Measures in *G*	Measures in unit $\frac{m}{s^2}$	
Orientation	Measured in unit *rad*	Measures in unit *degree*	AndroSensor data transformed to unit rad & flipped by dividing by 60

Table 2: Overview of data transformations

Data Aggregation and Train-test split

Each activity of approx. 20 minutes duration was down sampled to either 20 Hz, 5 Hz or 2 Hz i.e. every 5th, 20th or 50th row was included. The data was split into training and test files such that 15 minutes of the activity were used in the training set and 5 mins of the activity were used in the test set, followed by aggregating specific columns using a rolling window function of a specific length.
In the experiments, rolling window of either 20 seconds or 5 seconds were used i.e. columns were aggregated over 20 or 5 rows. Fig 3 shows an overview on aggregation and train-test split used before training the models using the classical ML approach.

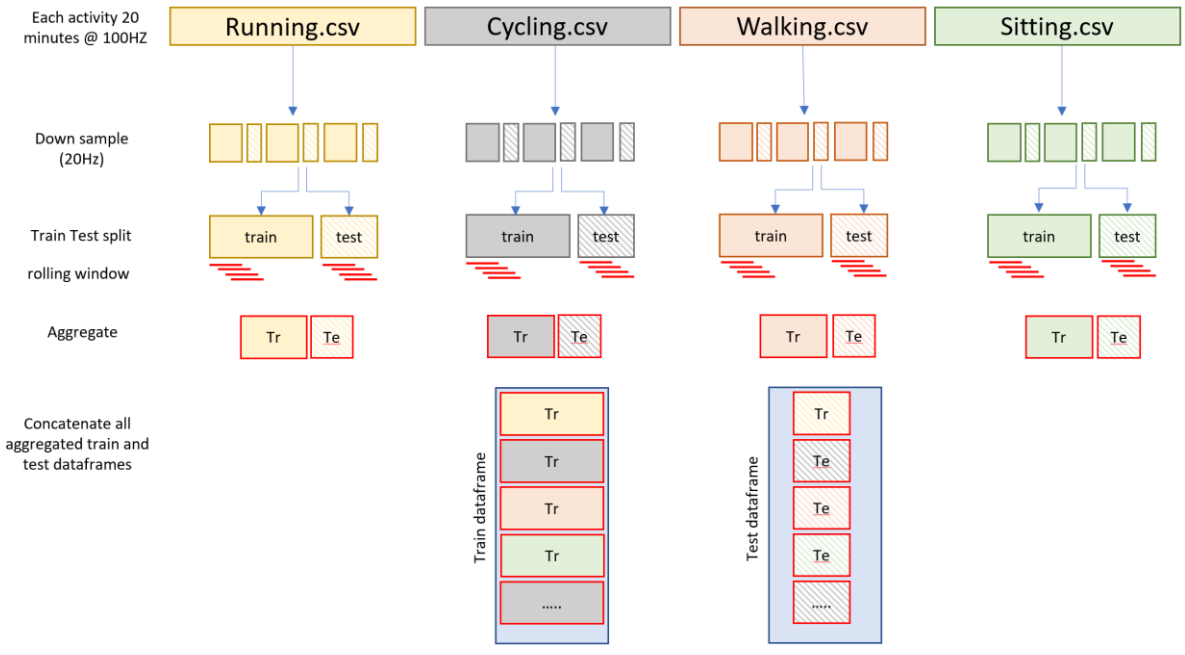


Fig 3: Overview of train – test split

Data was down sampled from 100 Hz to either 20 Hz, 5 Hz or 2 Hz, followed by aggregating 20 seconds or 5 seconds of data :

- **Aggregation 1**: mean and standard deviation on $sensor_x$, $sensor_y$ and $sensor_z$. Aggregation 1 is coordinate dependent i.e. it results in new variables such as $mean - sensorA_{axis}$ and $std - sensorA_{axis}$
- **Aggregation 2**: mean and standard deviation on absolute value of minimum, maximum and vector magnitude of $sensorA$. Aggregation 2 is coordinate independent.

Model training and results

Classical ML Models: An overview on different model runs using classical ML is available here. The training was done on aggregated data (aggregation 1 or 2) and different model runs were logged using weights and biases (wandb.ai). Fig 4 and 5 below show an overview of different runs with data aggregation 1 and 2 respectively.

Reading from left to right, it shows the data frequency used (20Hz / 5Hz / 2Hz), followed by aggregation applied on moving window size (20 seconds or 5 seconds), proportion of the data used as test/validation set (20% or 50%) , criteria used to measure the quality of fit, maximum depth of the tree, resulting accuracy on training data and accuracy on validation/test set.

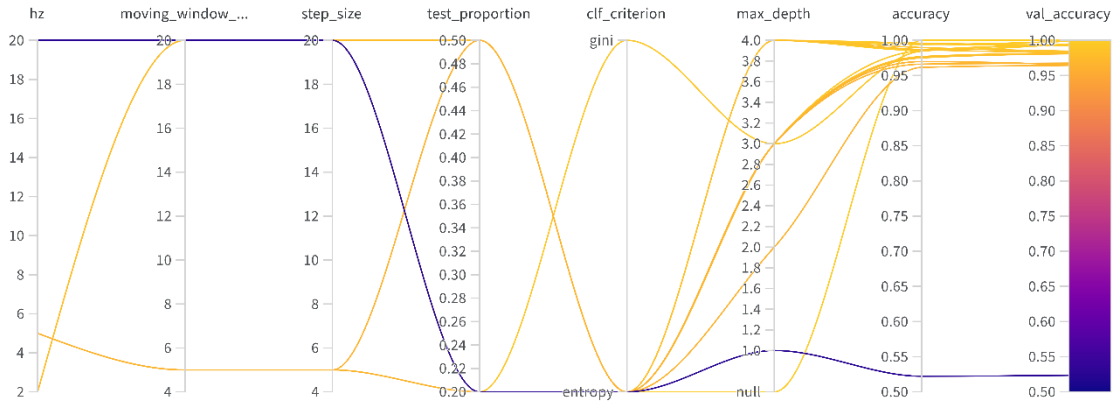


Fig 4: Decision tree model summary with aggregation 1

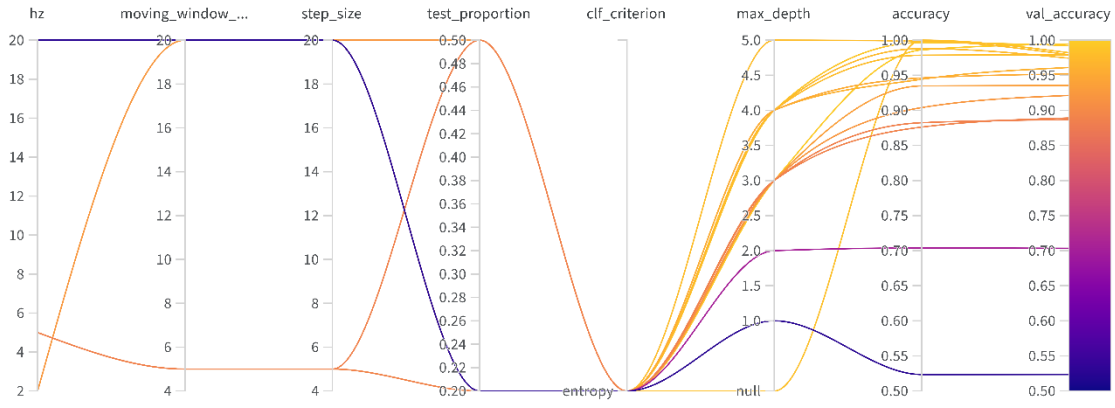


Fig 5: decision tree model summary with aggregation 2

It was observed that models trained with decision tree model using aggregation 1 resulted in high validation accuracy (over 95% on test data with max_depth greater than or equal to two). This was also the case by using 50% of the data as test data. One possible reason could be that the model is overfitting on data when sensor coordinates are present as variables in dataset.
On the other hand, decision tree with data aggregation 2 (coordinate independent), resulted in a validation accuracy of 90% with maximum depth of three.
Grid Search for best parameters using 4 fold CV with decision tree for different combination of moving window and HZ is added in appendix.
Detailed report is available here: Link to wandb report

Deep learning Model:

Recurrent Neural networks (RNN) are often used for sequential data such as text data, time-series and weather forecasting. A model with a Gated Recurrent Unit (GRU) followed by one or two dense layers was used. The output is a vector of length 4, each with a probability for 4 classes. The deep learning framework used was TensorFlow.

Fig 6 shows the model architecture using GRU as the first layer. The model complexity was chosen to be low so that model can be trained on FHNW Juypter Hub or Google Colab without the need of special infrastructure or cloud computing.

On average, it took around 18 to 20 minutes to train such a model.

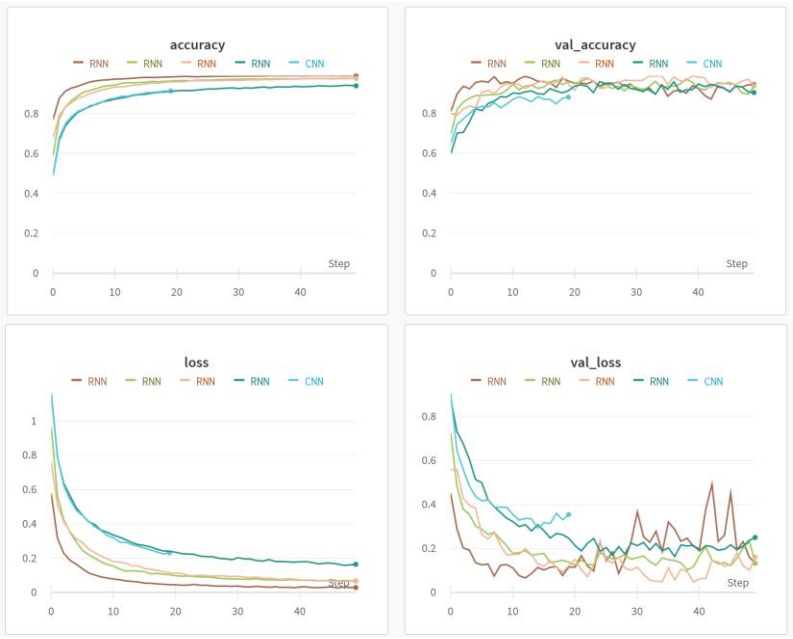


Fig 7: Model performance metrics

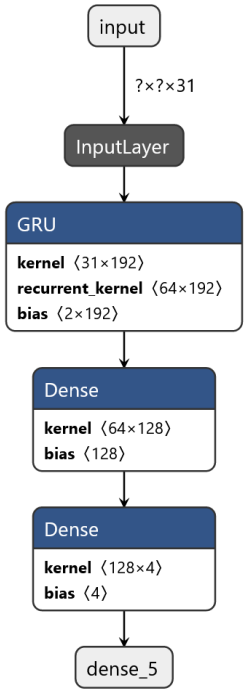


Fig 6: GRU Model architecture

Fig 7 shows the performance metrics 'accuracy' and 'loss' on the training and test set for different model complexities run for up to 50 epochs. Bottom-right graph shows the loss on the validation set. The spikes seen in this graph points to the fact that the models start

to overfit on test / validation dataset even though the loss continues to decrease on training set.
More details on the results and interpretation is available in report here

Conclusion:

An advantage of using decision tree is the ability to graphically visualize how such an algorithm makes decision to classify in one of the four classes. Such a graph is easy to interpret and shows how nodes are split resulting in most amount of information gain out of key features. Fig 8 shows an example of such a graph which considers depth of three.

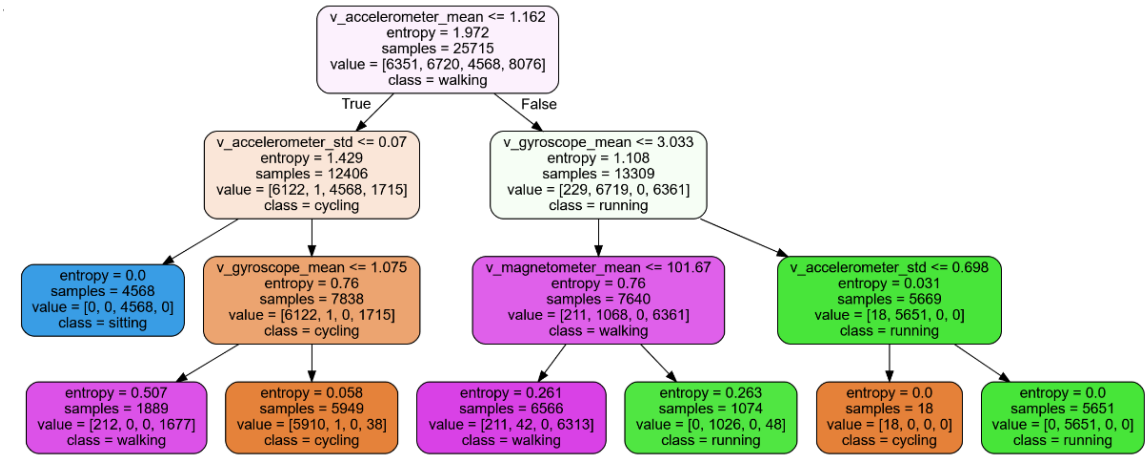


Fig 8: Graph of decision tree with aggregation 2

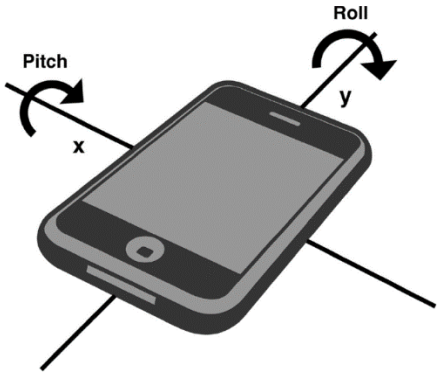
In practice, it would be desired for an algorithm to output a label within few seconds of starting an activity such as within 10 or 20 seconds. Such decisions are key towards the data aggregation approach. Deep learning models based on GRU, simpleRNN or LSTM work well for sequence data from sensors. Such models have an advantage to learn the aggregation to be applied based on model.

Appendix

1. About Sensors

The terminology about the main device sensors is introduced below:

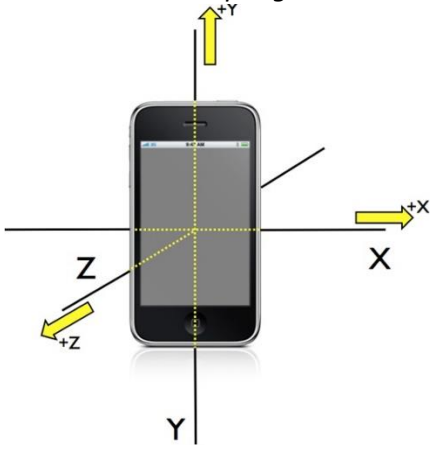
Yaw , **pitch** and **roll** refer to the rotation of the device in three axes. [6]



Source: Springer

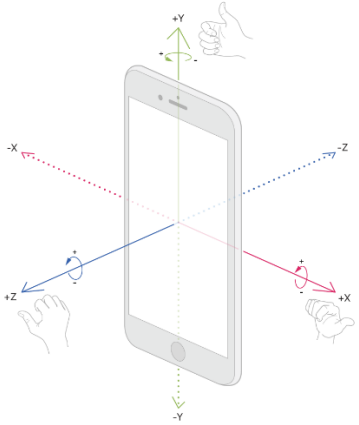
Accelerometer: measures the linear acceleration of the device on X, Y & Z axis as G-force values. On iPhone, a value of 1.0 represents a load of approximately 1-gravity (Earth's gravity = 9.8 m per seconds). While in motion, the figures represent the acceleration due to gravity, plus the acceleration of the device itself relative to its rest frame.

X corresponds to roll, Y to pitch and Z to whether the device is front side up or front side down.



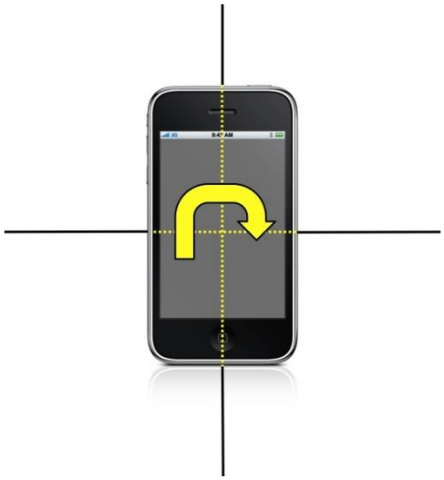
Accelerometer axes
Source: O'Reilly

Gyroscope: measures the rate at which a device rotates around a spatial axis in radians per second. Rotation value may be positive or negative depending on the direction of the rotation.



Source: Apple developer doc

Magnetometer: measures the earth's magnetic field relative to the device in micro Tesla. In the absence of any strong local fields, these measurements will be of the ambient magnetic field of the Earth, allowing the device to determine its "heading" with respect to the geomagnetic North pole. Combing the heading (yaw) information returned by the device with the roll and pitch information returned by the accelerometer helps determine the true orientation of the device in real time.



Source: O'Reilly

2. Meta-Data for raw data (csv files):

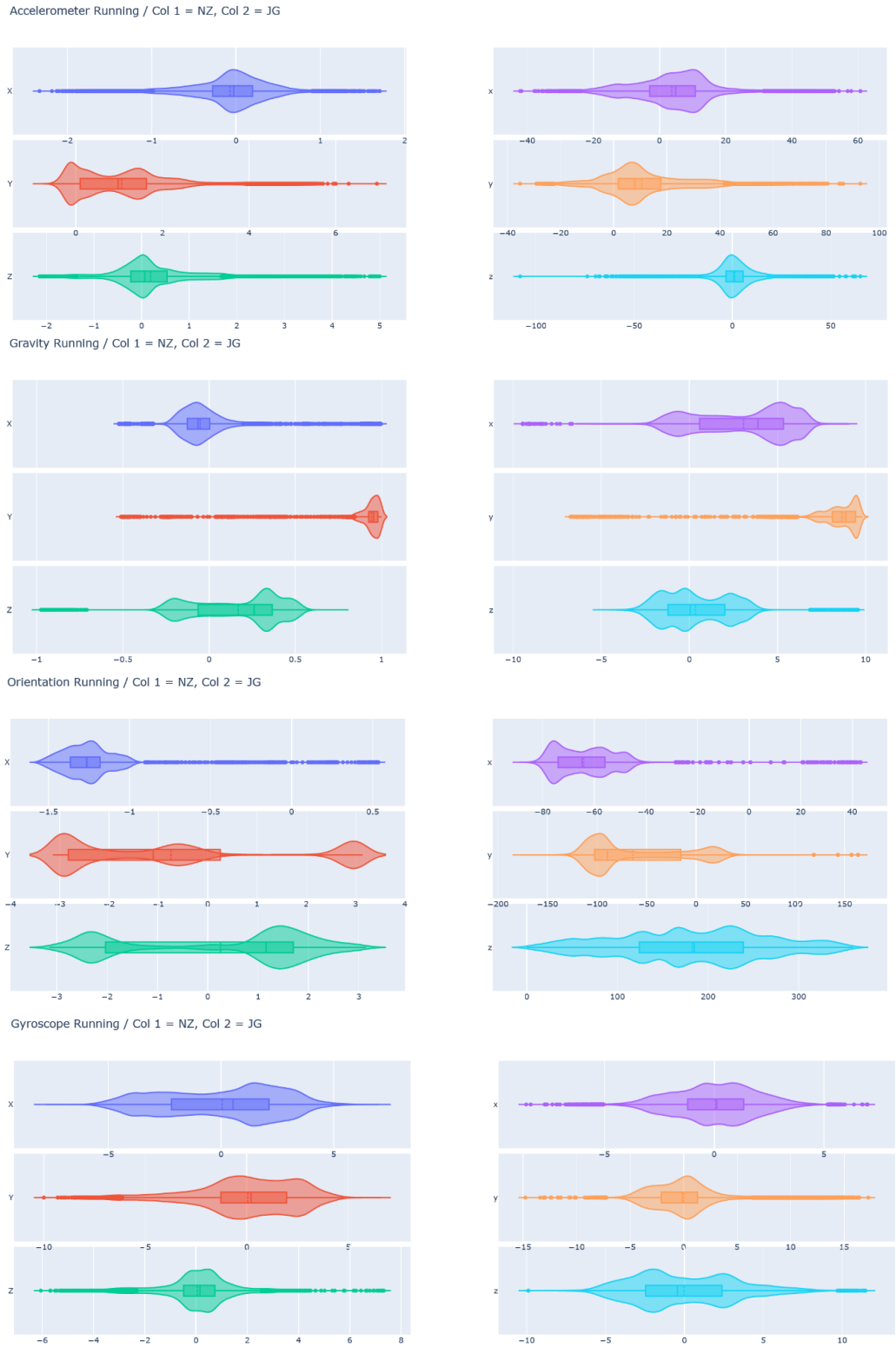
	file	user	activity	pocket	position_x	position_y
0	cycling_jg_1.csv	jg	cycling	left pant pocket	screen towards body	upside down
1	cycling_jg_2.csv	jg	cycling	left pant pocket	screen towards body	upright
2	cycling_jg_3.csv	jg	cycling	left pant pocket	screen towards body	upright
3	cycling_jg_4.csv	jg	cycling	left pant pocket	screen not towards body	upside down
4	cycling_jg_5.csv	jg	cycling	left pant pocket	screen not towards body	upright
5	running_jg_1.csv	jg	running	left pant pocket	screen towards body	upside down
6	running_jg_2.csv	jg	running	left pant pocket	screen towards body	upright
7	running_jg_3.csv	jg	running	left pant pocket	screen not towards body	upside down
8	running_jg_4.csv	jg	running	left pant pocket	screen not towards body	upright
9	running_jg_5.csv	jg	running	left pant pocket	screen not towards body	upright
10	sitting_jg_1.csv	jg	sitting	left pant pocket	screen towards body	upside down
11	sitting_jg_2.csv	jg	sitting	left pant pocket	screen not towards body	upside down
12	sitting_jg_3.csv	jg	sitting	left pant pocket	screen towards body	upright
13	sitting_jg_4.csv	jg	sitting	left pant pocket	screen not towards body	upright
14	walking_jg_1.csv	jg	walking	left pant pocket	screen towards body	upright
15	walking_jg_2.csv	jg	walking	left pant pocket	screen towards body	upside down
16	walking_jg_3.csv	jg	walking	left pant pocket	screen towards body	upside down
17	walking_jg_4.csv	jg	walking	left pant pocket	screen not towards body	upside down
18	walking_jg_5.csv	jg	walking	left pant pocket	screen not towards body	upright
19	cycling_nz_3.csv	nz	cycling	left pant pocket	screen not towards body	upright
20	cycling_nz_8.csv	nz	cycling	left pant pocket	screen towards body	upside down
21	cycling_nz_9.csv	nz	cycling	left pant pocket	screen towards body	upright
22	running_nz_1.csv	nz	running	left jacket pocket	screen not towards body	upright
23	running_nz_3.csv	nz	running	left pant pocket	screen not towards body	upright
24	running_nz_6.csv	nz	running	left pant pocket	screen towards body	upside down
25	running_nz_7.csv	nz	running	left pant pocket	screen towards body	upright
26	sitting_nz_3.csv	nz	sitting	left pant pocket	screen not towards body	upright
27	walking_nz_1.csv	nz	walking	left jacket pocket	screen not towards body	upright
28	walking_nz_2.csv	nz	walking	left jacket pocket	screen towards body	upright
29	walking_nz_4.csv	nz	walking	left pant pocket	screen towards body	upside down
30	walking_nz_5.csv	nz	walking	left pant pocket	screen towards body	upside down

Figure 1: Meta-data on raw data

3. Raw sensor data

Running

Below sensor data for activity ‘running’ is plotted and compared for both users



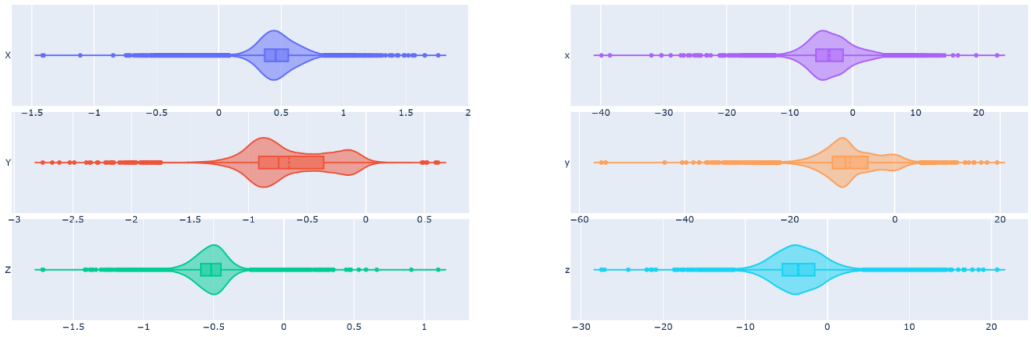
Walking

Below sensor data for activity ‘walking is plotted and compared for both users :

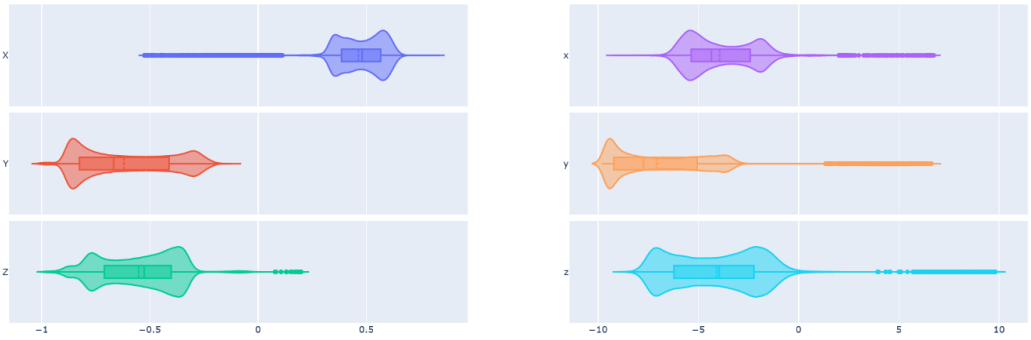
Cycling

Below sensor data for activity 'cycling' is plotted and compared for both users:

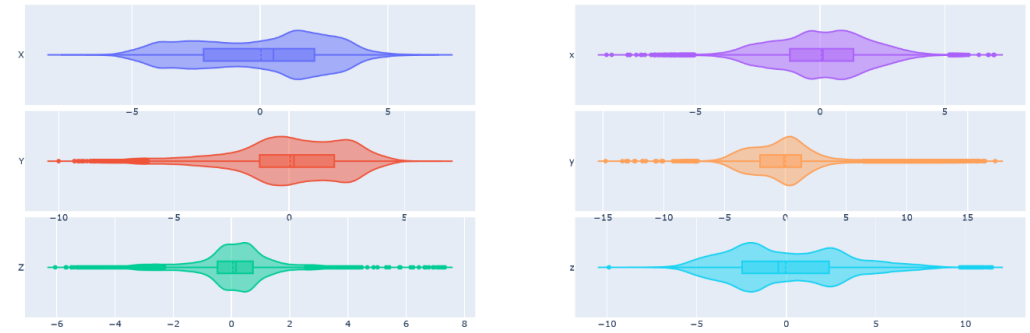
Accelerometer cycling / Col 1 = NZ, Col 2 = JG



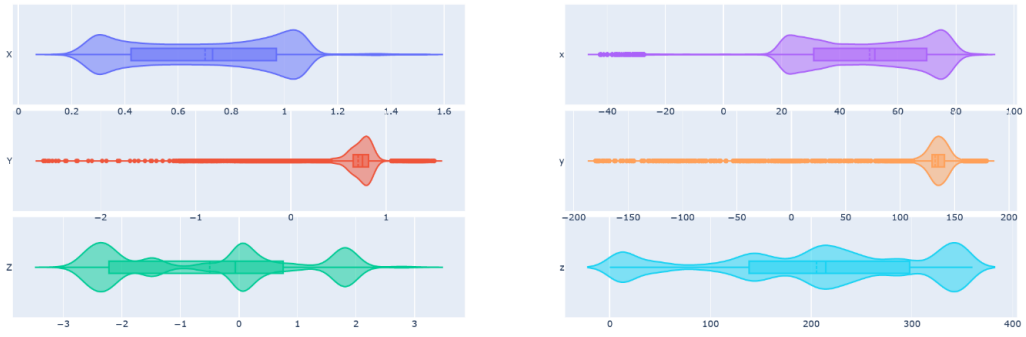
Gravity Cycling / Col 1 = NZ, Col 2 = JG



Gyroscope running / Col 1 = NZ, Col 2 = JG

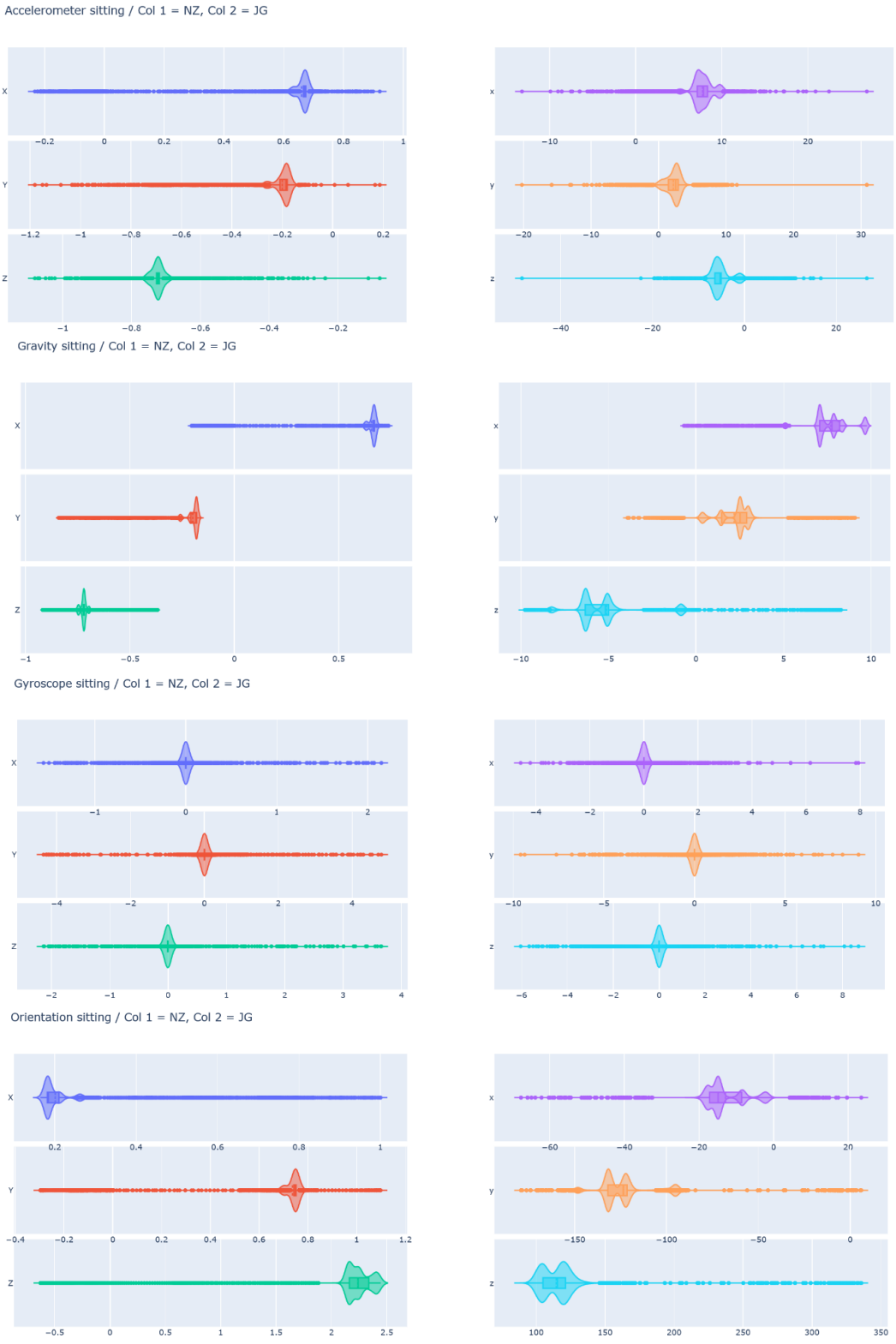


Orientation cycling / Col 1 = NZ, Col 2 = JG



Sitting

Below sensor data for activity ‘sitting’ is plotted and compared for both users :



4. Correlation map

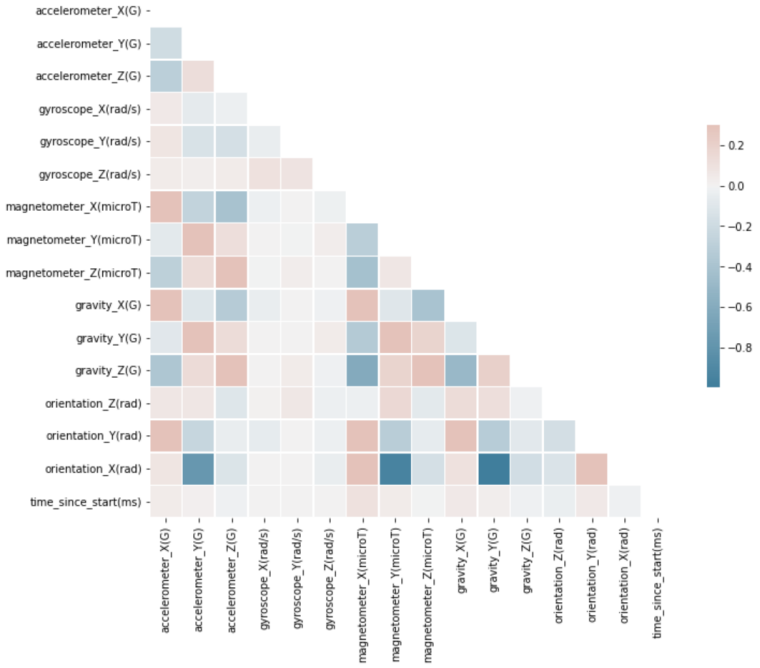


Fig: Correlation heatmap raw data user 1

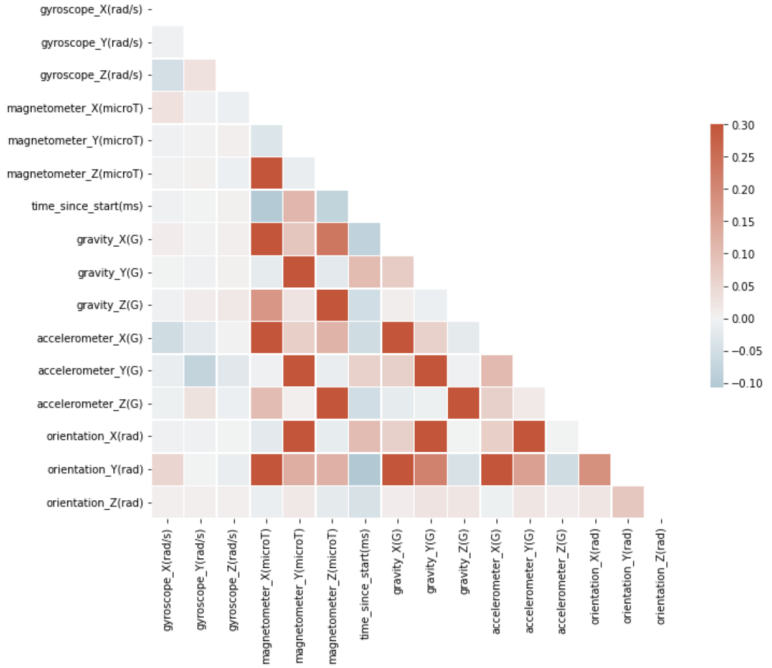


Fig: Correlation heatmap raw data user 2

5. Missing value analysis

Missing value analysis was done using library missingno. The height of the bar in the bar-chart below indicates the presence of valid values.

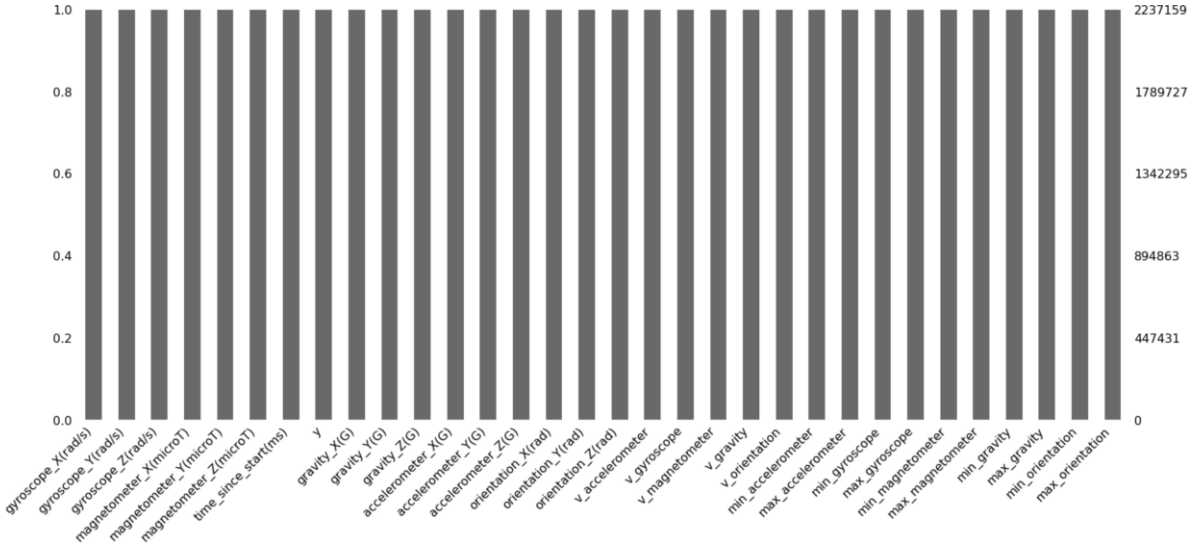


Fig: Missing value analysis – raw data from user 2

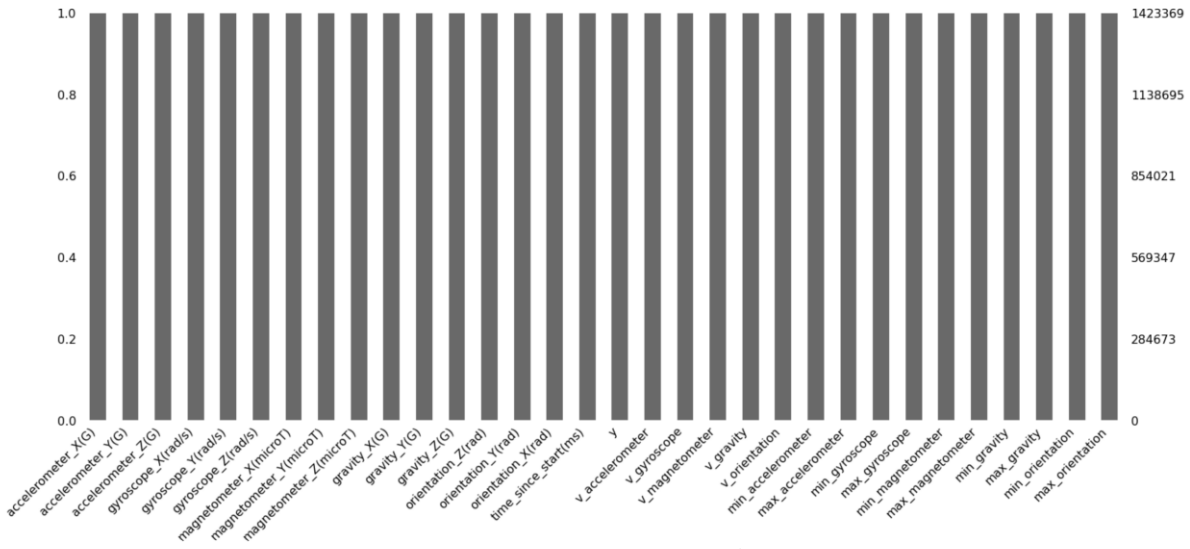
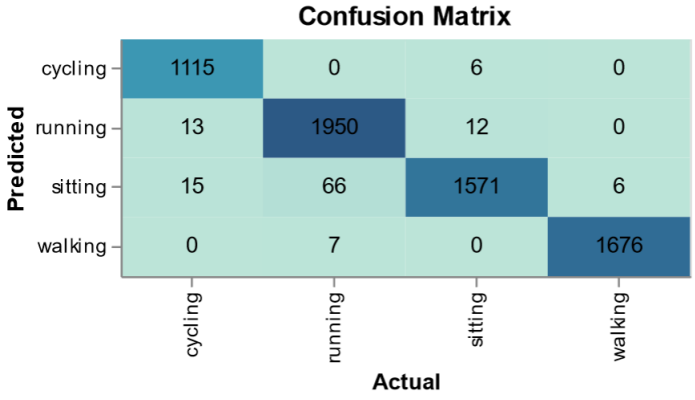
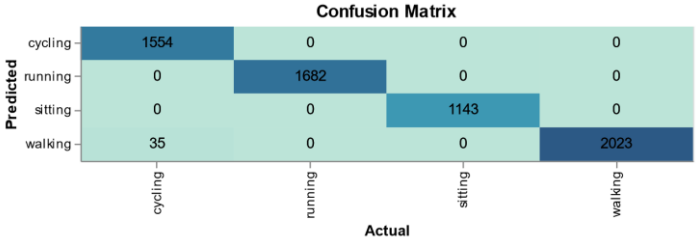


Fig: Missing value analysis – raw data from user 1

6. Overview- Classical ML Model run:

Decision Tree : model with maximum depth 3 or 4
Random Forest: model with number of trees: 1000 or 500
Logistic regression: Multi-Layer Perceptron (MLP) with 1 neuron

Classical ML model performance on test /validation set using confusion matrix:

Model	Aggregation	Confusion matrix	Train/Test accuracy
Logistic regression	Aggregation 1 – 20Hz and 20Seconds window		Train accuracy: 96.5% Test accuracy: 98%
Decision Tree	Aggregation 2 - 20Hz & 20 seconds window		Train accuracy: 98.6% Test accuracy: 99.4%

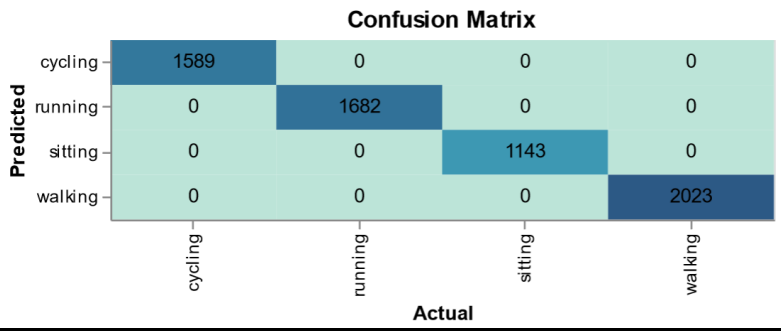
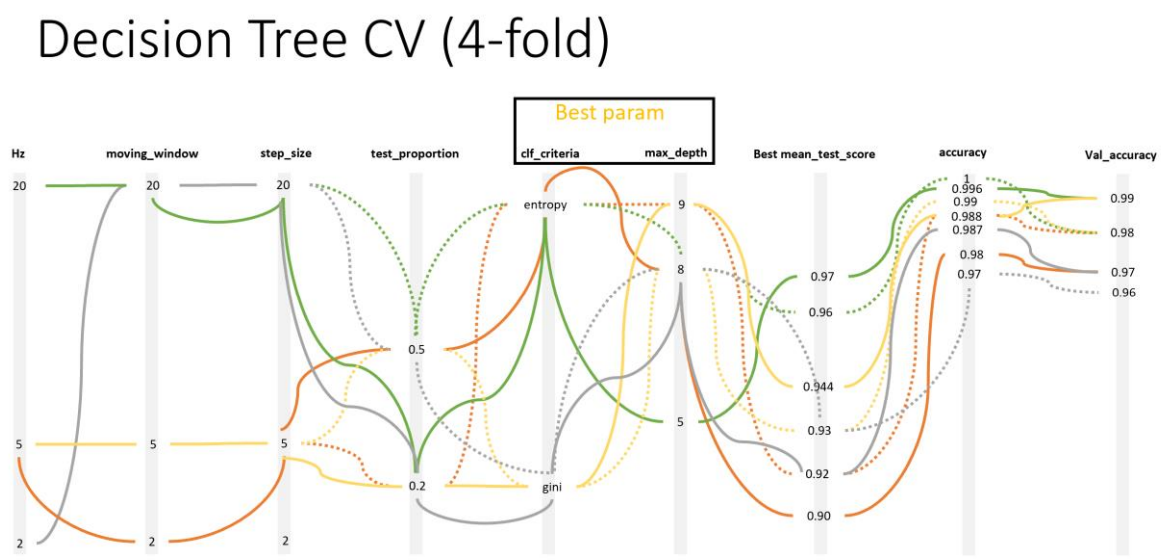
Random Forest	Aggregation 2 – 20Hz & 20 seconds window		Train and test accuracy: 100%

Table 3: Overview classical ML model run

7. Decision tree CV Grid Search



8. Decision tree graphs

Aggregation 1:

Reading it from top to bottom, there are 25715 samples at root node (node 0). The first decision boundary is based on variable 'orientation_Y(rad)_std' <= 0.261. This decision boundary is selected by testing all the possible decision boundaries splitting the dataset and choosing the one that minimizes the entropy of the splits. Class 'walking' is picked at this node as it has maximum number of data points (8076). The process continues iteratively until maximum depth is reached (here defined as 3) or until all data points are separated.

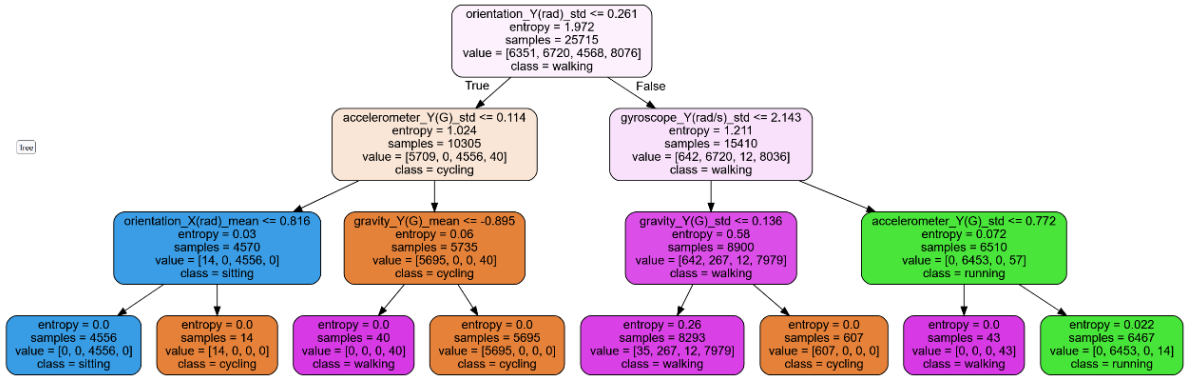


Fig 8: Graph of decision tree with aggregation 1

Aggregation 2:

Decision tree on data aggregation 2 has same number of samples at Node 0 but the decision boundary is based on variable 'v_accelerometer_mean'.

9. References

[1] SensorLog iOS App
[2] AndroSensor Android App
[3] Research paper : Amari Vaughn, Paul Biocco, Yang Liz, Mohd Anwar, 2018 Activity Detection and Analysis Using Smartphone Sensor
[4] Time-series resampling and moving windows: Link
[5] iPhone accelerometer, Gyroscope, Magnetometer
[6] Yaw, pitch and roll : Link & Springer

10. List of Abbreviations

ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
HAR	Human Activity Recognition
LSTM	Long Short-Term Memory
CSV	Comma separated value
EDA	Explorative data analysis
GRU	Gated Recurrent Unit

11. Terminology

Log	Log is referred as an activity over 20 minutes => 1 csv file
Variable	A variable is a specific characteristic of a data point which is recorded. For example, gravity_x, accelerometer_x etc
Raw data	Data which has not been pre-processed
Class label	In context of this challenge, label here is either one of the following: walking, running, cycling, sitting.