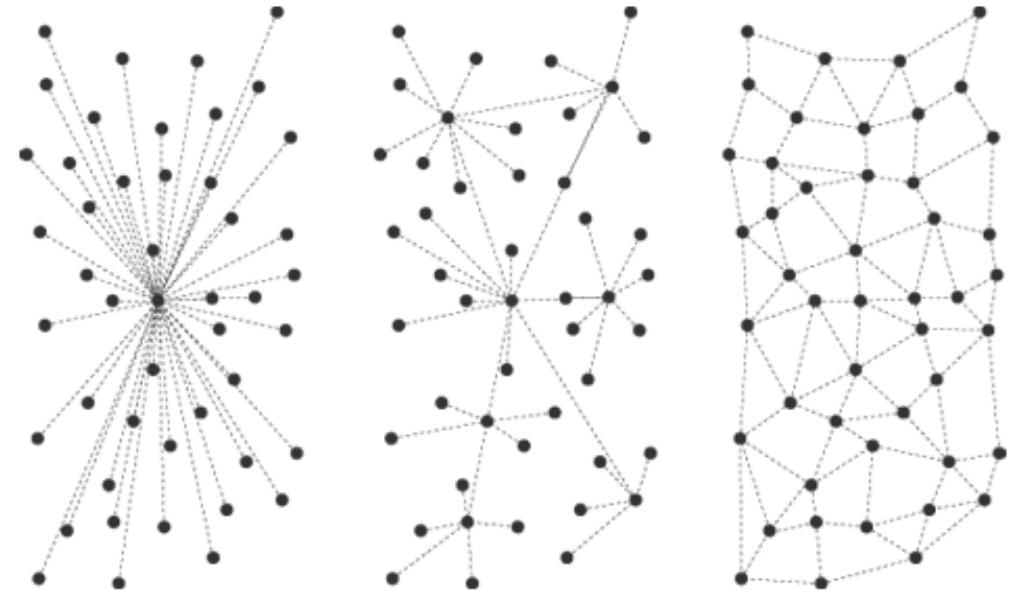


Activity detection

Introduction

In this project we were asked to utilize the Spark + MLIB package to produce classification for activities on 9 different subjects. The dataset is fairly large with about ~400 thousands entries per subject

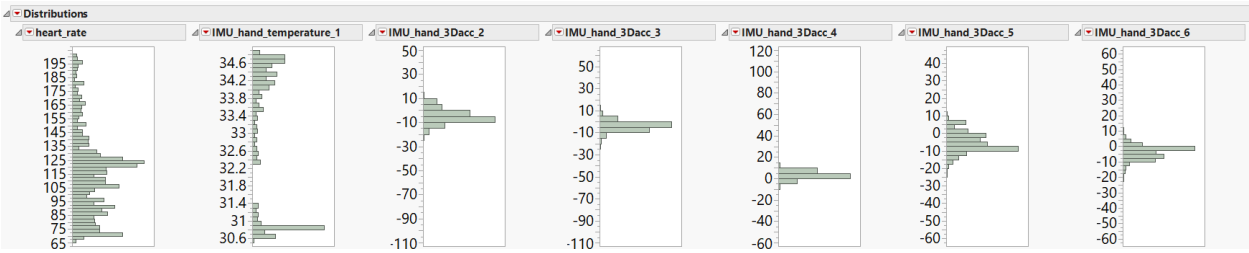
In this project we will present the exploration of the data,
the cleaning of the dataset, preprocessing that was done and finally the tuning of the model and it results

We have chosen to use the random forest algorithm

Note: in the code we have used Pandas, but only in the process of collecting and presenting the summary data of the results.

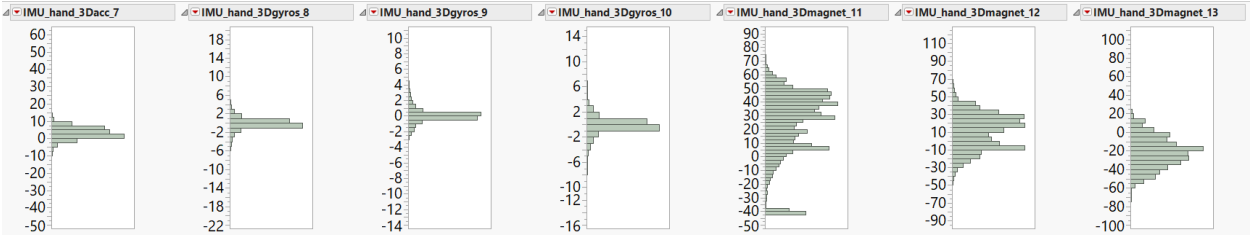
DATA EXPLORATION

In this section we have used one of the subjects and examined the feature values



the heart rate sensors seems tuned and calibrated and values are diverse

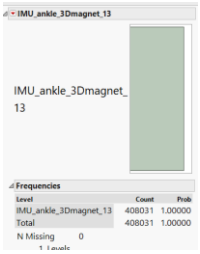
rest of the parameters in the group seems of normal distribution and not skewed



As the group above, information seems of normal distribution and not skewed

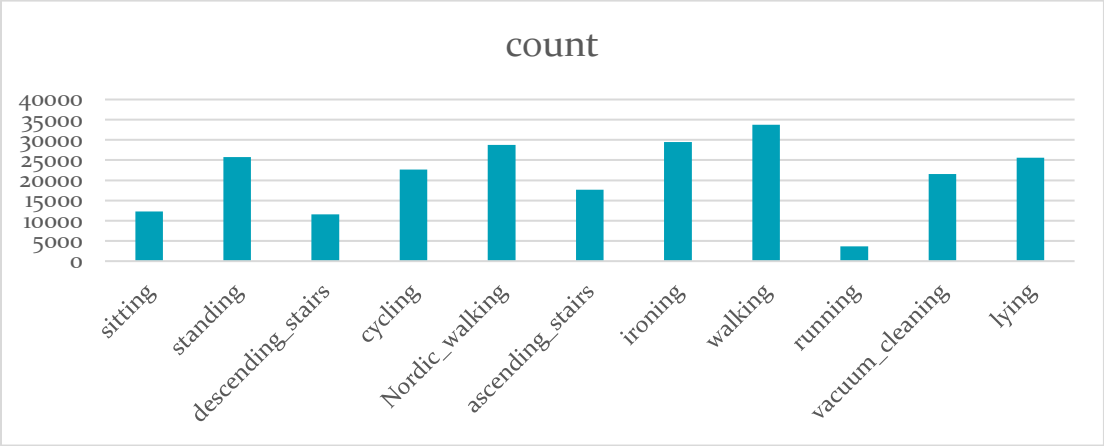
.. the rest of the features seems as both groups above,

Only execution is IMU_ANKLE_3dMAG, which doesn't carry any real information



Since we are using random forest, we didn't remove this feature since the chances of it being used is minimal

As for the labels in this subject, the labels are well distributed with multiple labels in place and not a single very large group label



DATA CLEANING

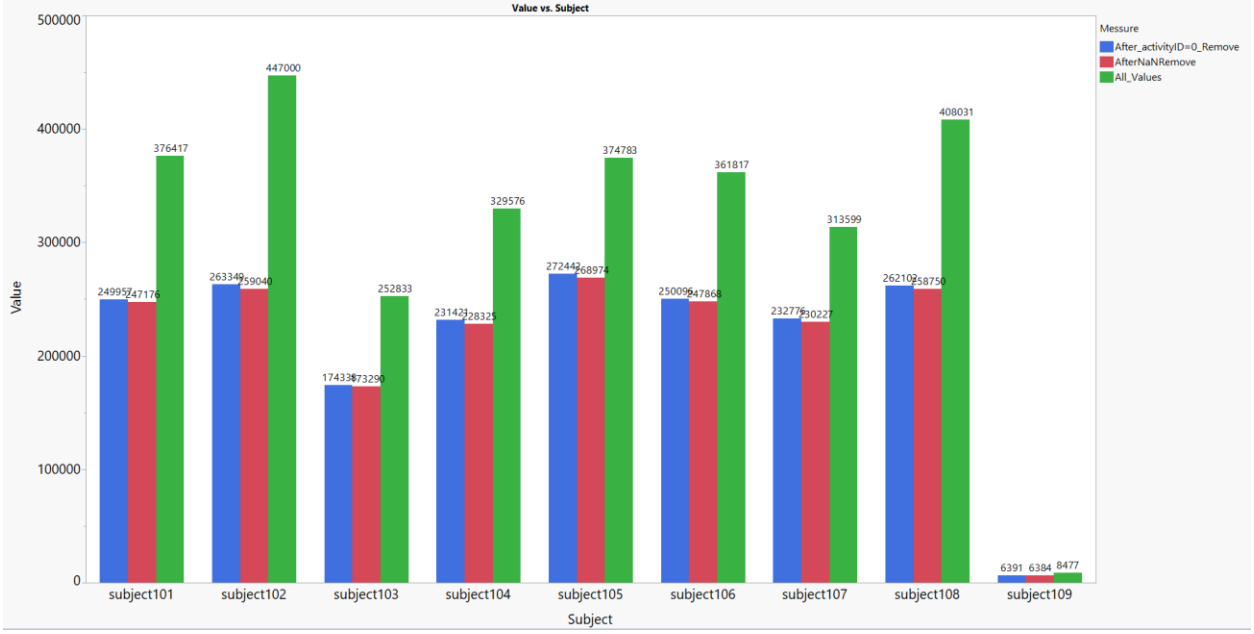
Columns remove

As the ReadMe instructed the “orientation” columns cannot be counted for thus removed from the dataset,

Overhaul 12 columns removed

Data Cleaning

In the data cleaning process the following was taken place:
remove of activityID=0
remove of Nan/Null values



As can be seen from the graph, subject 9 have signifyingly low count of values, which may results in overfitting when preforming the training of the mode,
for the rest of the subjects; removal of “activity=0” resulted in ~20% reduction in the dataset and removal of “NaN” Values had very little influence on the dataset size

DATA PREPROCESSING

Note: as raised in the class, there was a initial request to split the data by time, this will results in "Unknown" activities in some cases, to deal with that, we have performed also the preprocessing on all the data combined and then preformed the split

The split itself was done randomly meaning with no regard to the time-stamp as suggested in the class

The following preprocessing took place

1. Filling the heartrate values
    a. The heart rate monitor works in different frequency from the other given sensors, therefore filling of the values was needed
    b. For this task we used the fill forward method, filling the following missing values with the latest given value

timestamp	activityID	heart_rate
84.28	1	NaN
84.29	1	NaN
84.3	1	NaN
84.31	1	NaN
84.32	1	NaN
84.33	1	80.0
84.34	1	NaN
84.35	1	NaN
84.36	1	NaN
84.37	1	NaN
84.38	1	NaN
84.39	1	NaN
84.4	1	NaN
84.41	1	NaN
84.42	1	NaN
84.43	1	NaN
84.44	1	80.0
84.45	1	NaN
84.46	1	NaN
84.47	1	NaN

only showing top 20 rows

Figure 2 before NA fill

timestamp	activityID	heart_rate
84.28	1	null
84.29	1	null
84.3	1	null
84.31	1	null
84.32	1	null
84.33	1	80.0
84.34	1	80.0
84.35	1	80.0
84.36	1	80.0
84.37	1	80.0
84.38	1	80.0
84.39	1	80.0
84.4	1	80.0
84.41	1	80.0
84.42	1	80.0
84.43	1	80.0
84.44	1	80.0
84.45	1	80.0
84.46	1	80.0
84.47	1	80.0

only showing top 20 rows

Figure 1 After NA fill

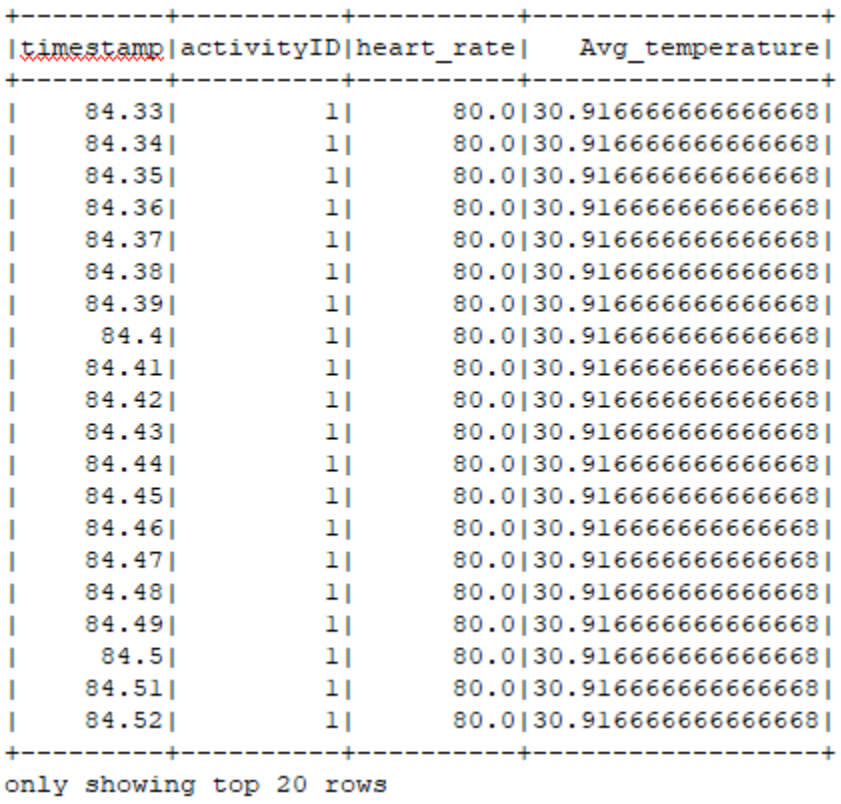
2. Reducing the heart rate in rest
    a. In the given PDF it was given for each subject what its "resting" heart rate, we have created a new column with the heart rate menus the "resting" heart
    b. This will not have big impact in model per subject as we do here, but in combined model it will help the model

timestamp	activityID	heart_rate	heart_rate_menus_rest
84.28	1	null	null
84.29	1	null	null
84.3	1	null	null
84.31	1	null	null
84.32	1	null	null
84.33	1	80.0	22.0
84.34	1	80.0	22.0
84.35	1	80.0	22.0
84.36	1	80.0	22.0
84.37	1	80.0	22.0
84.38	1	80.0	22.0
84.39	1	80.0	22.0
84.4	1	80.0	22.0
84.41	1	80.0	22.0
84.42	1	80.0	22.0
84.43	1	80.0	22.0
84.44	1	80.0	22.0
84.45	1	80.0	22.0
84.46	1	80.0	22.0
84.47	1	80.0	22.0

only showing top 20 rows

Figure 3 create new column HR menus rest

3. Averaging the temperate
    a. 2 sensors in "ankle" "chest" and "leg" was giving the same reading of "temperature"
    b. As we saw these features has high importance when running the model, therefor we have used the 3 and average them into a single column "Avg_temperature" which had high importance when running the model



```
+---------+----------+----------+------------------+
|timestamp|activityID|heart_rate|   Avg_temperature|
+---------+----------+----------+------------------+
|    84.33|         1|      80.0|30.916666666666668|
|    84.34|         1|      80.0|30.916666666666668|
|    84.35|         1|      80.0|30.916666666666668|
|    84.36|         1|      80.0|30.916666666666668|
|    84.37|         1|      80.0|30.916666666666668|
|    84.38|         1|      80.0|30.916666666666668|
|    84.39|         1|      80.0|30.916666666666668|
|     84.4|         1|      80.0|30.916666666666668|
|    84.41|         1|      80.0|30.916666666666668|
|    84.42|         1|      80.0|30.916666666666668|
|    84.43|         1|      80.0|30.916666666666668|
|    84.44|         1|      80.0|30.916666666666668|
|    84.45|         1|      80.0|30.916666666666668|
|    84.46|         1|      80.0|30.916666666666668|
|    84.47|         1|      80.0|30.916666666666668|
|    84.48|         1|      80.0|30.916666666666668|
|    84.49|         1|      80.0|30.916666666666668|
|     84.5|         1|      80.0|30.916666666666668|
|    84.51|         1|      80.0|30.916666666666668|
|    84.52|         1|      80.0|30.916666666666668|
+---------+----------+----------+------------------+
only showing top 20 rows
```

Figure 4 averaging the temperature into 'Avg_temperature'

4. Replace the labels
    a. For ease of use we have replaces the "activityID" which is INT into the actual named label

```
+---------+----------+----------+-----+
|timestamp|activityID|heart_rate|label|
+---------+----------+----------+-----+
|    84.28|         1|      null|lying|
|    84.29|         1|      null|lying|
|     84.3|         1|      null|lying|
|    84.31|         1|      null|lying|
|    84.32|         1|      null|lying|
|    84.33|         1|      80.0|lying|
|    84.34|         1|      80.0|lying|
|    84.35|         1|      80.0|lying|
|    84.36|         1|      80.0|lying|
|    84.37|         1|      80.0|lying|
|    84.38|         1|      80.0|lying|
|    84.39|         1|      80.0|lying|
|     84.4|         1|      80.0|lying|
|    84.41|         1|      80.0|lying|
|    84.42|         1|      80.0|lying|
|    84.43|         1|      80.0|lying|
|    84.44|         1|      80.0|lying|
|    84.45|         1|      80.0|lying|
|    84.46|         1|      80.0|lying|
|    84.47|         1|      80.0|lying|
+---------+----------+----------+-----+
only showing top 20 rows
```

Figure 5 replacing the activiyID with label

HYPERPARAMETER TUNNING

Although running the model with almost default parameters yield very good results, we went forward and preformed a Hyper-Parameter tuning for the Random Forest classifier

For this task we have used the CrossValidation method using K=5 and used one of the subjects to tune the model

The following parameters were tuned

.addGrid(rf.maxDepth, [0, 1, 5])

.addGrid(rf.minInfoGain, [0.01, 0.001])

.addGrid(rf.numTrees, [5, 10, 30,100])

The MaxDepth which control the depth of each Decition Tree

The InfoGain which controls the minum required information gain to split a node

And the number of trees which controls the number of tress in the “forest”

This stage took the longest time to execute, running with K=5 with multiply parameters

Results in 33-minute run

```
Command took 33.65 minutes --  at 7/9/2018, 11:09:08 AM on My Cluster
```

Results

The results of the hyper parameter tuning yield the following best model:

numTrees	100
minInfoGain	0.01
maxDepth	5

These parameters were used applied to all subjects and models

Note: the chosen model was the one with the biggest number of parameters (trees =100) (depth = 5)

There is a actual concern of overfitting the model, but since we have used cross validation and the dataset is “quite big”.

Of course, this will be examined when preforming the model training and testing on all subjects

PREFORMANCE

Upon running each tuned model on the test set, we have used a simple split for Training and Testing the model

Using a ratio of 20% given to the test, as the dataset is large, we assumed this is safe to perform a simple split rather than a cross validation.

In general, the results were very good

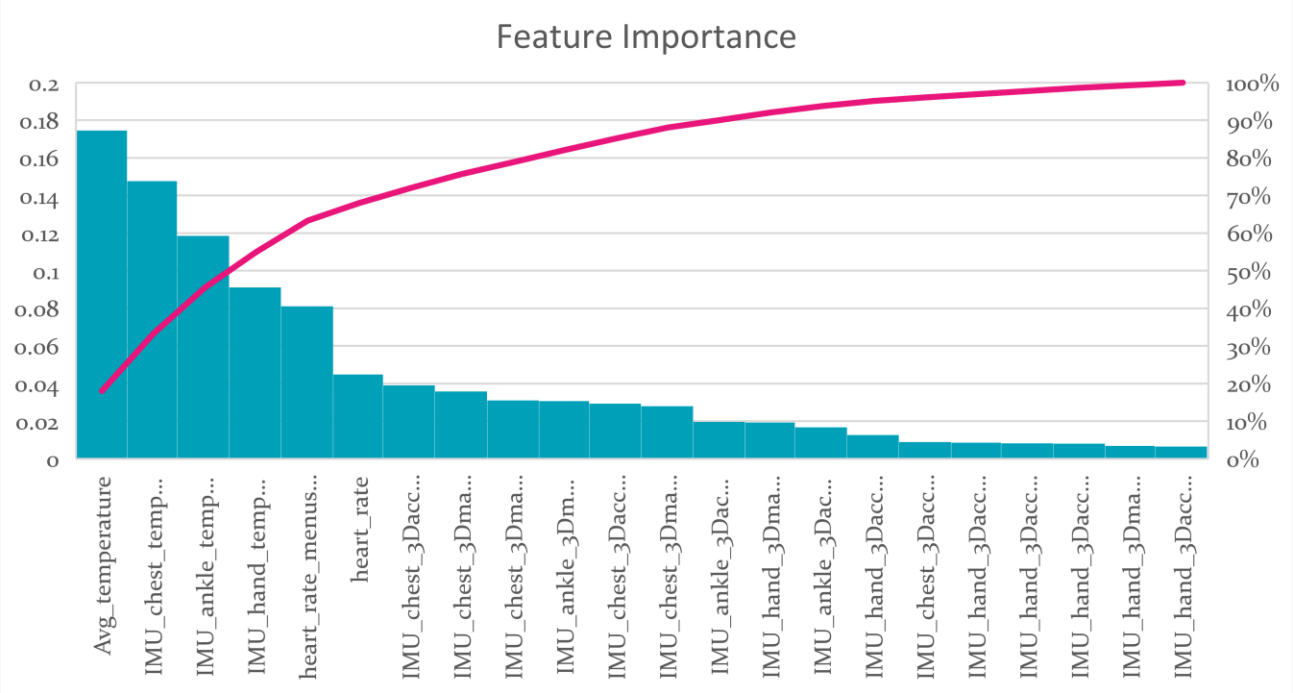
Execution time

Fit the model per subject and testing the results took ~25 minutes

Command took 25.87 minutes at 7/9/2018, 1:49:49 PM on My Cluster

Feature importance

As a result of using random forest we can easily extract the feature importance, we have extracted the feature importance of subject 8



The Bars represent the value of the feature importance, and the line percent the accumulated value.

As the line reaches 100% means that the rest of the feature had very little importance, and the presented one had the most influence on the results

The crafted featured of AVG temp had the biggest importance, while still the separated values of temperature also had high value. After which the “heart_rate_menus_rest” followed by “heart_rate” as well.

Results

For evaluating the models, we have used the following metrics

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

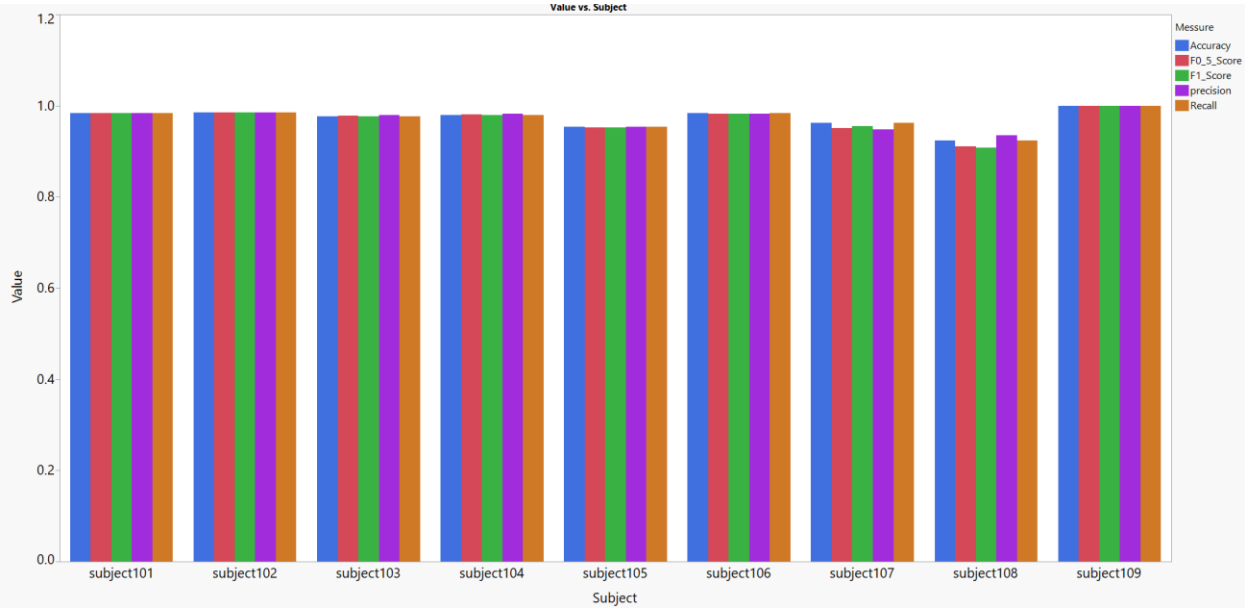
$$Accuracy\ = \frac{True\ Positive + True\ Negative}{ALL}$$

$$F1 = \frac{Precision * Recall}{1 * Precision + Recall}$$

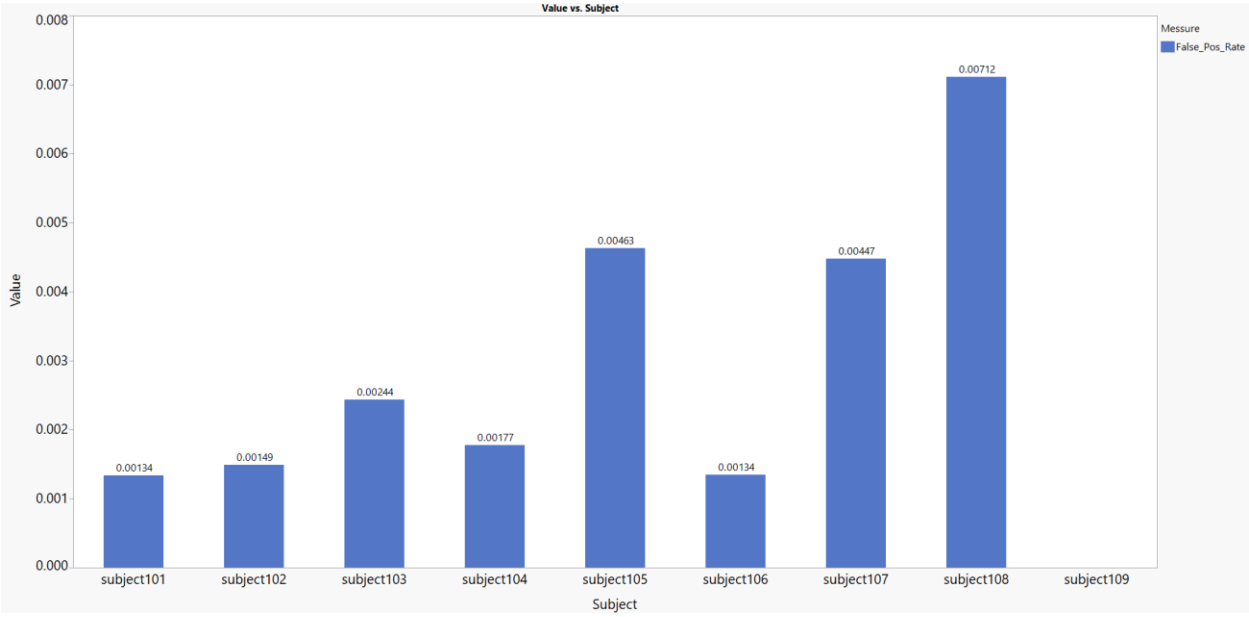
$$F0.5 = (1 + 0.5^2)\frac{Precision * Recall}{0.5^2 * Precision + Recall}$$

As these tables are given for binary classification, for the multiclass case we have used a weighted measure, which means every measure is calculated on each class multiplied by the precent of the class out of all the classes

Overhaul results

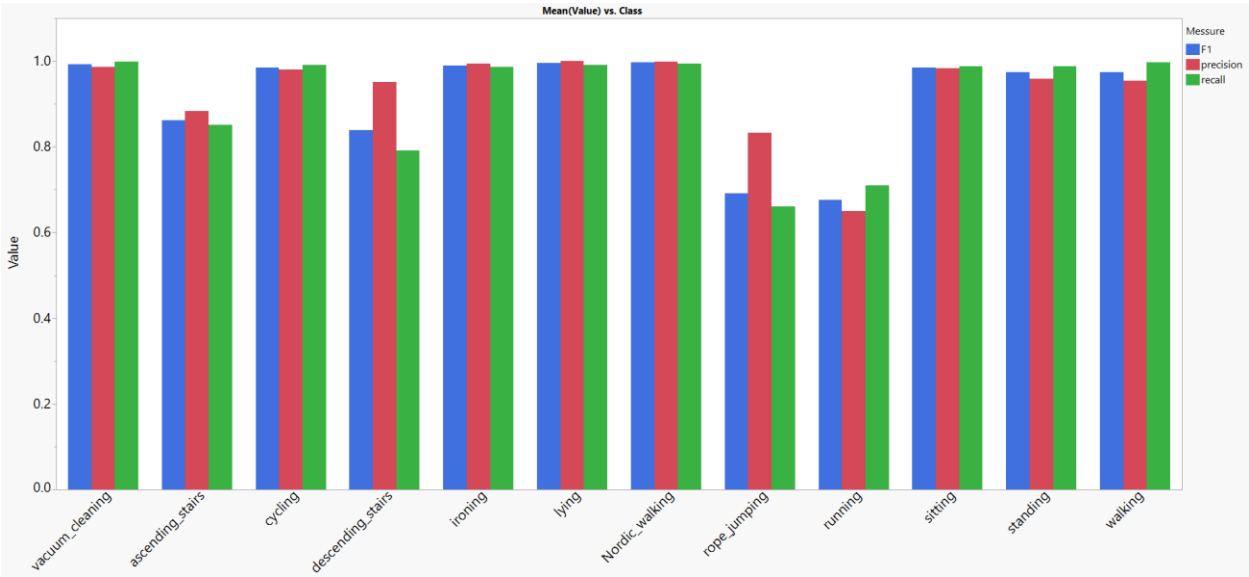


As can be seen in graph, all metrices are at 100% or nearly 100%, having precision, recall and accuracy at these very high values can be interpreted as successful execution of the model classification also in the predicted class and also in low false alarm rate



Examining the false positive results (weighted per class) show very good results in low amount of miss classification

Results per class



In the graph above, which shows the average performance of each activity throughout all subjects show very good results, all besides activity “Running” and “rope jumping”

Drill down to this case we observe that action of “rope jumping” is not present in all subjects and in subject08 is yielded a very high precision, but low recall where most of the activity “rope jumping” was classified as “running”