Machine learning predictions on workout acquired data

Department of Industrial Engineering
Course of Foundation of measurements for mechanical systems

Pierfrancesco Oselin - Mat. 202199
pierfrancesco.oselin@studenti.unitn.it

Contents

1 Introduction

With the smartphone revolution in the digital era and the cost reductions in the electronic manufacture, sensors have become widespread. Nowadays they can be found even in the most simple devices, such as lamps (sensitivity sensors for touchless switches), thermostats or smart assistants.

This persistent presence of sensors and the desperate need of data acquisitions is characterizing the so called Fourth Industrial Revolution. Smaller and smaller as well as well tuned devices that allow high frequency sampling are every single day required to be developed. In particular automated factories, robot-managed industrial productions, autonomous driving or the Internet of Things are some of the main agents that are boosting the development and research in this field.

However ones of the most common devices which include lots of different sensors are smartphones: from accelerometers to gyroscopes, from Hall to proximity sensors, from LiDAR to ultrasonic-based fingerprint readers these devices represent a huge possibility to acquire data. In particular, due to they are present in people's pockets everyday, they are a huge deal for acquiring behavioral and health data.

Moreover, in 2014 a radical revolution in the tech world occurred with the introduction of smartwatches: small electronic devices just as normal watches, but with the capability to communicate with smartphones and the outside world. Even if for most of the people they represent just a possibility to make calls or receive notification straight onto your wrist, for data analysts they are not.

Heart rate, blood oxygenation, blood pressure, glycemia level as well as activity monitoring are just few of the possible parameters that can be tracked. The collection of data associated to millions of people are allowing in the last few years the development of brand new smart assistant, from automating tasks to health-related help.

Due to the increasing spread of smartwatches among people, newer ICT companies are competing in the app development rush, especially in the fitness and sport world. From run tracking to swimming monitoring there are tons of possibilities to keep performances and improving under control.

Because of this, I decided to simulate the data acquisition of a typical smartwatch to demonstrate how these fitness companies treat and analyze

data, proving how is possible to recognize different activities or sport just by exploiting data extracted by these new devices.

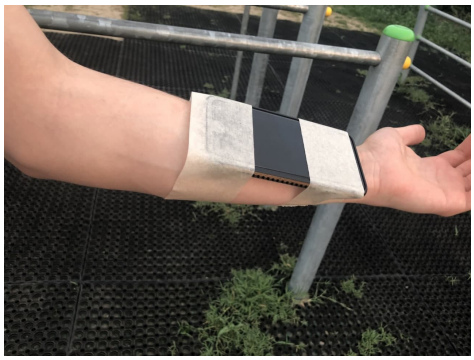
2 Tasks and data acquisition

For this experiment data acquisition has been made possible by exploiting the Matlab smartphone application developed by MathWork, which gives the user the possibility to read and store signals from all the sensors inside this device. In particular the ones available in this case could access and record

- Acceleration
- Magnetic Field
- Orientation
- Angular Velocity
- Position

Due to the choice to analyze sport activities, only dynamic-related sensors were considered, so geo-related data such as position and magnetic field intensity have been neglected.

The main idea, as briefly said in Section 1, is to simulate the data acquisition process likewise in typical smartwatches. To accurately achieve this goal the smartphone has been attached to the tester's left wrist.



Moreover, for this analysis 6 different activities have been considered

ID	Activity type
Activity 1	Jump rope
Activity 2	Sit-ups
Activity 3	Crunches
Activity 4	Chin-ups
Activity 5	Push-ups
Activity 6	Run

and in order to be as reliable as possible, different testers with different smartphones have been asked to record data. In particular

Tester	Smartphone
Subject 1	iPhone SE
Subject 2	Samsung A31
Subject 3	Xiaomi Mi A2
Subject 4	Samsung J5
Subject 5	Huawei P9
Subject 6	iPhone XS

A typical data acquisition session during this experiment looked like

Figure 1: In order: jump rope (1), sit-ups (2), crunches (3), chin-ups (4), push-ups (5) and run (6)

For each device and tracked activity the sampling frequency was set to $100\,Hz$ and various timing were recorded. Generally speaking, if more data are acquired, more detailed can be the analysis and therefore the final prediction.

3 Data analysis

In smartphones the common accelerometers for measuring *acceleration*, *angular velocity* and *orientation* are nowadays well engineered devices which can operate at relatively-low frequency: the bandwidth in the most of them is between 8 and $1000\,Hz$. In the most of the cases, they are MEMS-based devices.

Figure 2: Bosch BMI055, a 6-axis inertial measurement unit (IMU) consisting of a digital, tri-axial 12-bit acceleration sensor and a triaxial 16-bit gyroscope. It is one of the most popular sensor implemented in smartphones

A first approach to identify the different activities can be looking at the distribution in histograms for each activity and trying to find possible patterns

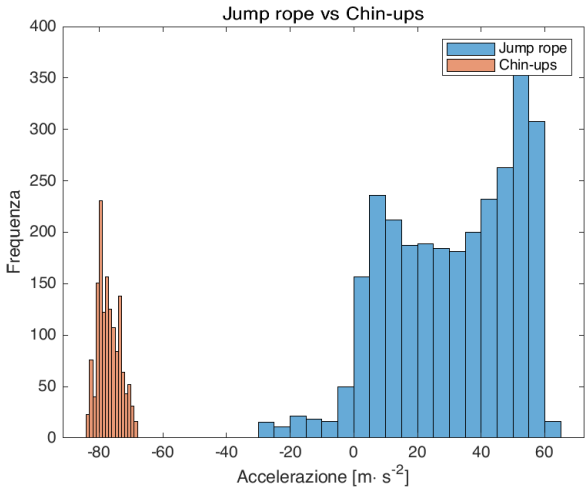


Figure 3: Comparison between histogram of two different activities

In Figure 3 an example is shown: the comparison between *jump rope* and

chin-ups looks pretty solid, however this is a fortunate combination, in fact analyzing other combinations histograms look like

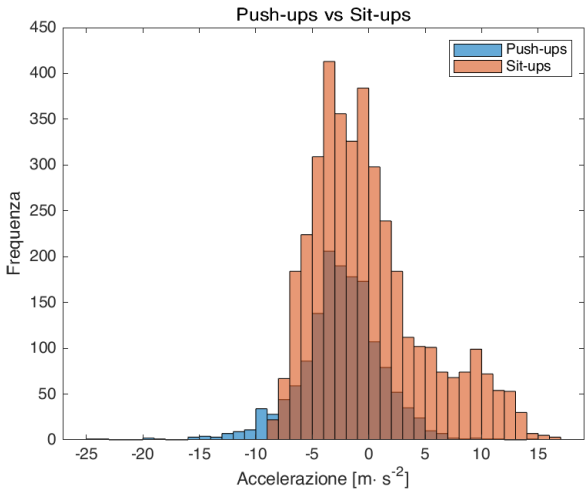


Figure 4: Comparison between histogram of two other activities

In this case the two activities overlap each others and the identification operation results to be very ambiguous. Because of this the need to analyze data in other ways raises up. A good solution can be switching to the frequency domain and repeat the analysis.

In order to do that a mathematical transformation must be applied to correctly convert data in frequency-domain signals: the Fourier Transformation has been therefore exploited to successfully apply the conversion.

In particular, due to discrete data acquisition (the *sampling frequency* was set to $100\,Hz$) the Fourier Transform is in actual fact called *Discrete Fourier Transform* (DFT). In Matlab, due to this conversion is applied very quickly, the transformation is called *Fast Fourier Transform* (FFT).

FFT is hence applied to the dataset and the comparison can be repeated

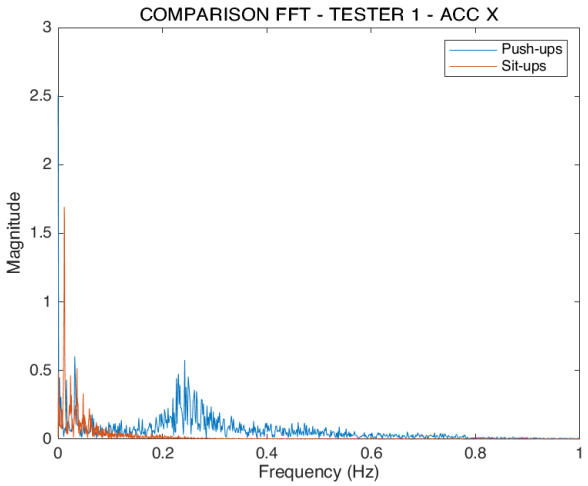


Figure 5: Comparison between frequency plots of two different activities

In Figure 5 the same activities shown in Figure 4 are plotted. It is possible to see how in the frequency domain *push-ups* and *sit-ups* are now distinguishable. Therefore basing the analysis on both time and frequency domains is a good strategy to obtain more reliable results.

Moreover, in the frequency domain a brand new information can be analyzed: noise overlap. Due to mass production cost constraints, smartphones sensors are fabricated with low accuracy, therefore during operations some noise affects them. Smartphone producers are aware of this downside and instead of introducing more reliable sensors (that would be economically disadvantageous) they compensate the problem with very well tuned digital solutions to extract desired cleaned signals.

In particular this can be achieved by introducing ad-hoc filters (low pass, high pass or band pass) to exclude noises just before signal processing and dynamic compensation.

Filters can be applied *online* or *offline*: online filters can be applied directly into the stream to obtain a real time filtering effect. However because of this unavoidable phase shiftings are introduced.

Offline filtering is instead a procedure applied just after data acquisition and storing: by this very precise filters can be designed, allowing almost unaltered modules and zero-phase shifting. Unfortunately these designs can be obtained with very in-deep analysis, definitely hard to achieve with just students' skills.

Looking at the acquired data plots, a constant offset modifies the curves: it is obvious how an high pass filter is required, in fact low noise behaves as a DC input which corresponds to a constant module shifting in the output signals.

Due to possible noise both in low and high frequencies a bandpass filter should be implemented. However, by looking at the frequencies spectrum (for example Figure 6) it is possible to see how high frequencies are totally irrelevant. On the contrary, because the information inside the signal are mostly in the low frequencies, an high pass filter must be applied. Both of them has been tested, realizing the following band filters

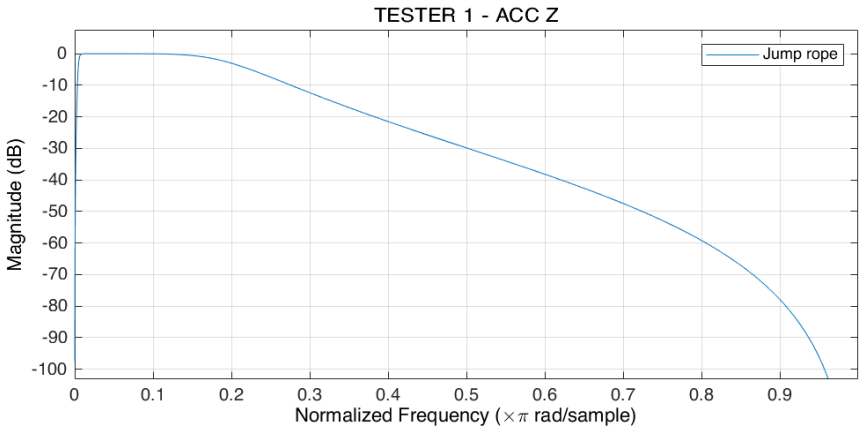


Figure 6: Band pass filter

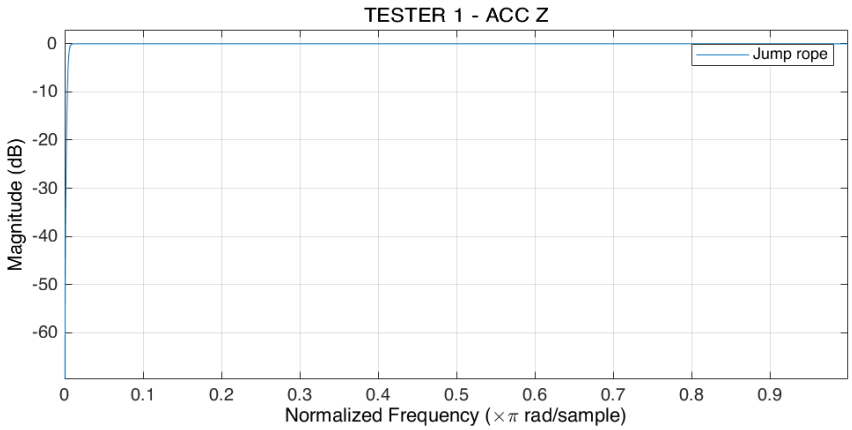


Figure 7: High pass filter

In particular third order Butterworth filters have been adopted. After few consideration on the last significant frequency data present, the cutoff frequencies have been set to $0.25\,Hz$ for the high pass filter and respectively 0.25 and $10\,Hz$ for the bandpass.

The differences were minimal, so the bandpass filter has been discarded.

Moreover, to be as reliable as possible the following choices should have been made

- One filter for each group of data (acceleration, angular velocity, orientation)
- One filter for each different device (iPhone, Samsung, etc.) due to different sensors
- Randomize data acquisition

Both of the above suggestions have not been implemented in this project due to lack of time. From now on I will assume the dataset is reliable enough.

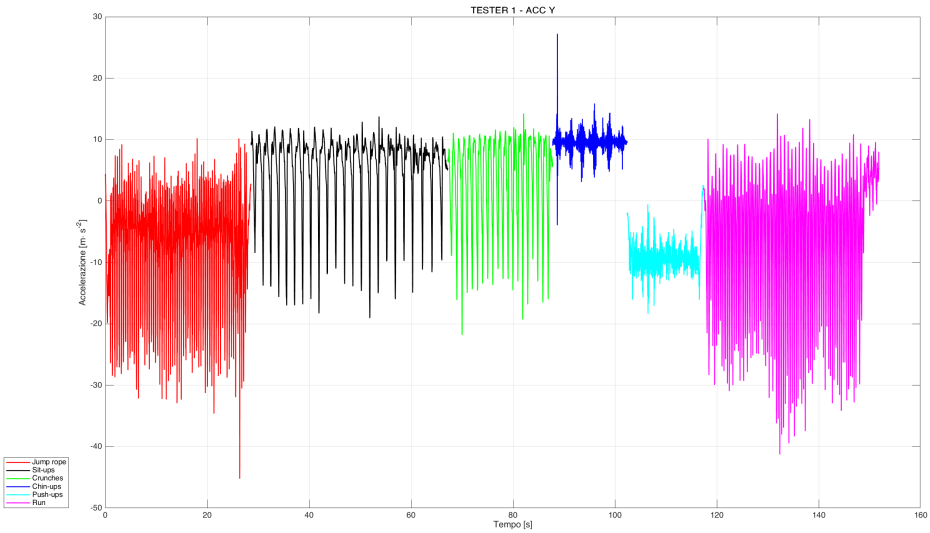


Figure 8: Unfiltered signal

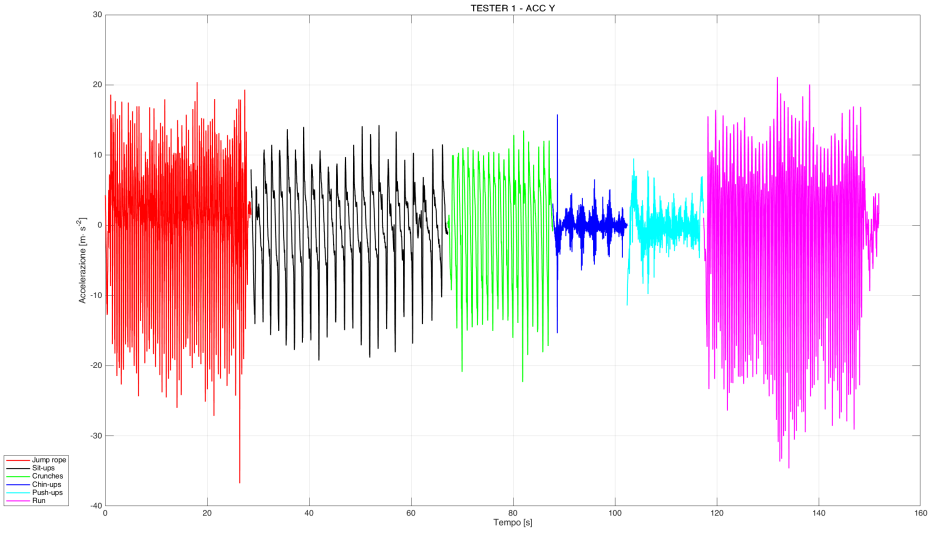


Figure 9: Filtered signal in the time domain, using a third order Butterworth filter

4 Matlab functions

During the online experimental session organized by the professor, a previously given dataset containing 6 different activities (walking. walking upstairs, walking downstairs, sitting, standing and laying down) was asked to be analyzed, eventually writing some functions to correctly manage data.

In particular the given data structure were

$$1 \times 30\, struct \;\Big|\; 1 \times 6\, cell\, array$$

where the vertical array was essentially made of one row for each tester and it showed two attributes: *actid* and *totalacc*. Actid was a collection of activities IDs, listed by sampled time whilst totalacc contained the acceleration data related to actid.

The cell array was instead a bigger data structure which presented 6 columns, one for each activity. In each cell 3 $n \times 150$ matrix were packed, one for each acquired coordinate. The main idea was to distribute the acquired data (merged across the testers) in a defined number of rows: for each row specific features were estimated in order to exploit unique parameters value on which the machine learning algorithm could lean on for making predictions.

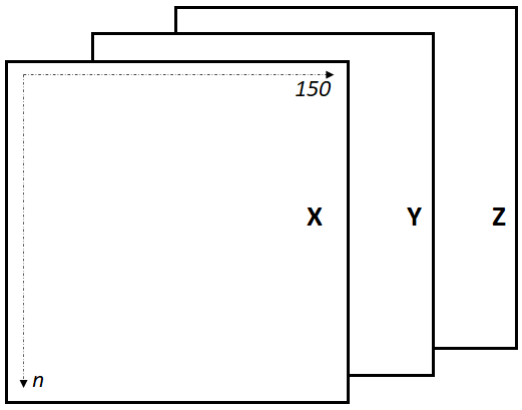


Figure 10: Second data structure organization

My idea was to create some functions which could manage and organized raw data from input and recreate structures similar to the ones listed above. In details, I wanted to write some could which could handle

- Total flexibility
- Data corruption repairing
- Automatic data merge, even from different devices
- Parameter tuning
- Possibility to be used for any data processing

Due to the fact the former given structure was named `Subjects` and the latter `Stat_c`, the functions I wrote have been called

```
convertToSub(dataset);
convertToStat(dataset);
```

where *dataset* was organized as

```
dataset = {tester_data1,
           tester_data2,
           tester_data3,
           tester_data4,
           tester_data5,
           tester_data6
           };
```

In this way the importing procedure has been kept as clean and easy as possible.

By creating data structures similar to the ones previously used in class, I was able to re-use the already written functions for the filtering, plotting and machine learning procedures.

The `subjects`-like structure has been used to analyze the acquired data in the time and frequencies domain in order to determine the *Butterworth coefficients* and the *cutoff frequency*.

The `stat_c`-like array has been used as starter ground to estimate the following statistics features

Mean	Std
Peak	Peak position
Median	Variance
Covariance	Range
Rms	Mode
Mean or median absolute deviation	

Table 1: Possible features

In order to follow the flexibility guideline previously introduced, the number of adopted features is decided each time by the user, as well as the dataset length and the column division.

The Matlab `classificationLearner` module has then been used to process the obtained matrix of features. In the case all statistical parameters are selected and considering 9 different data classes (three for each type of acceleration)

$$n_{features} \times m_{data\,group} + ID_{column} = 11 \times 9 + 1 = 100 \text{ columns}$$

and the data structure looked like

Acc X	**Acc Y**	**Acc Z**	**Ori X**	**Ori Y**	**Ori Z**	**Ang X**	**Ang Y**	**Ang Z**	**ID**
...	...	...	...	...	...	...	...	...	...

Table 2: Data structure sample

where each data type contained the chosen statistics

Acceleration X							
Mean	**Std**	**Peak**	**Peak Pos**	**Variance**	**Covariance**	**Rms**	...
...	...	...	...	...	...	...	...

Table 3: Data structure sample

The algorithm has been able to work with 99 parameters for each `stat_c`'s row, which collapses in one point in a scatter plot. A single row can be seen as located in a 99-dimensional-space.

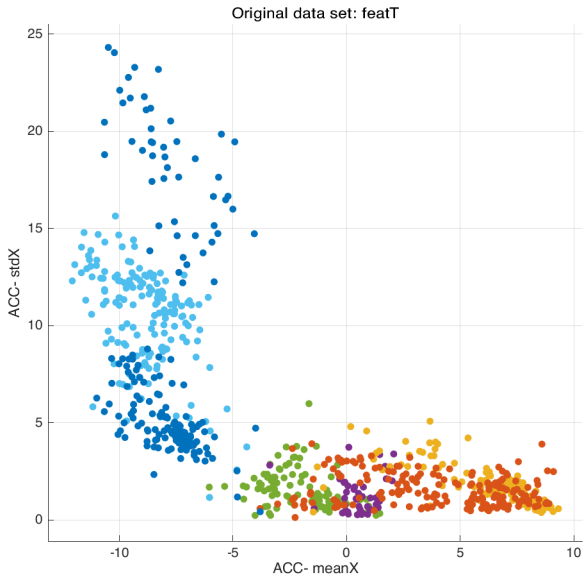


Figure 11: Classification Learner scatter plot

On both axis different features can be set in order to understand how they are related to each other. In particular

- If the points are casually widespread, the selected features have nothing in common. New information are added
- If the points are tidily distributed, a mathematical relation bonds the two features and no extra information are added.

In detail the second situation looks like

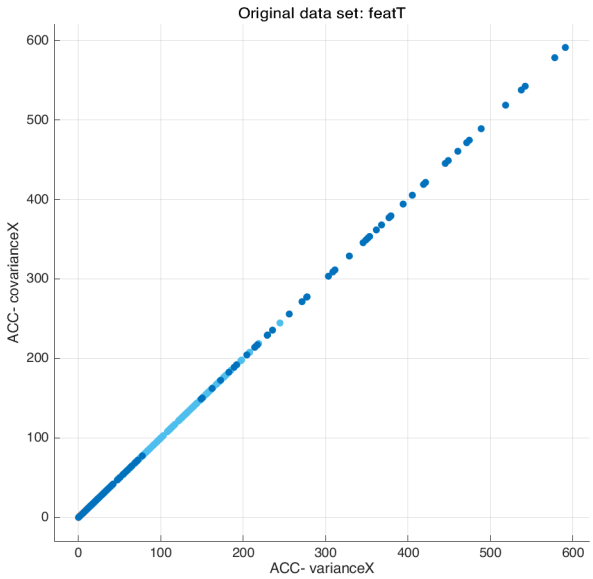


Figure 12: Classification Learner scatter plot but no extra information are added

In order to obtain the most reliable results without falling into super-dimensional spaces a principal component analysis (PCA) should be undertake, as useless features are discarded.

Finally with the classification learner interface a machine learning model can be trained using different algorithms. Among all the available ones the *Bagged Trees*, an ensemble classifier, has been chosen for its reliability and performances.

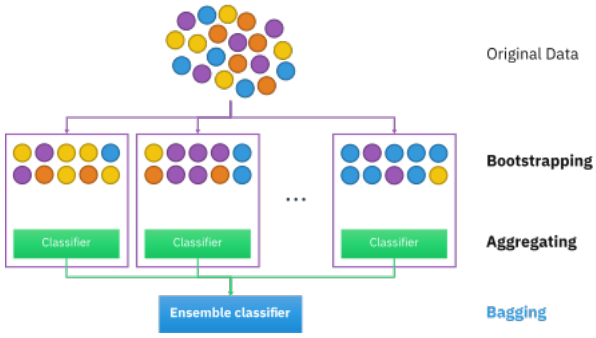


Figure 13: Bagged trees algorithm

A smart way to test if the model can correctly predict new data is to train it with a portion of the entire dataset and try to predict behaviors with the remaining part.

In this case

- 80% formed the training dataset
- 20% formed the testing dataset

The overall behavior can be easily represented using *confusion matrixes*, tables that show how many data has been correctly interpreted and how many misinterpreted. In particular

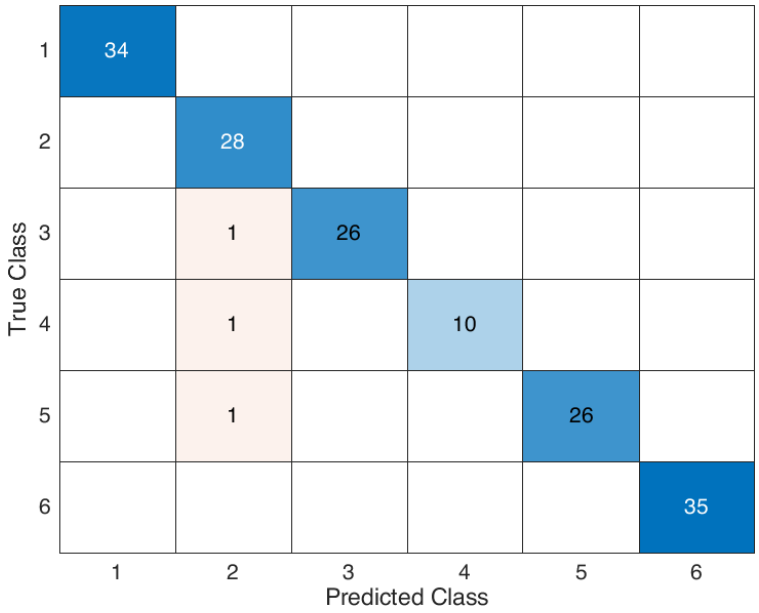


Figure 14: Confusion matrix using 80% of the dataset as training base

Another smart way to test how reliable is the trained model is to use entire testers' data as training base and the rest of them as testing. This is the most interesting way to investigate the machine learning algorithm, in fact smartwatches data analysis models are trained on specific datasets and made working on completely external sources. Following the first idea ($80\% - 20\%$) testing data belong to the same dataset as the training ones and results could be not so trustworthy.

Therefore in this experiment different combinations were tested

Training base	**Testing base**
2 testers	4 testers
3 testers	3 testers
4 testers	2 testers
5 testers	1 testers

Table 4: Adopted combinations

Due to different numbers of data in the training sets, different accuracies were hit

Training base	**Accuracy**
2 testers	96.4%
3 testers	94.6%
4 testers	91.1%
5 testers	92.7%

Table 5: Obtained accuracies after the training sessions

A decreasing-accuracy trend was expected due to the increasing amount of considered data. If poor dataset are analyzed, the algorithm can strongly predict behaviors and patterns. On larger data it instead predicts with less confidence, due to the increased possible relationships among them. However, even if accuracy decreases while feeding the testing base, reliability definitely improves, as the respective confusion matrixes show.

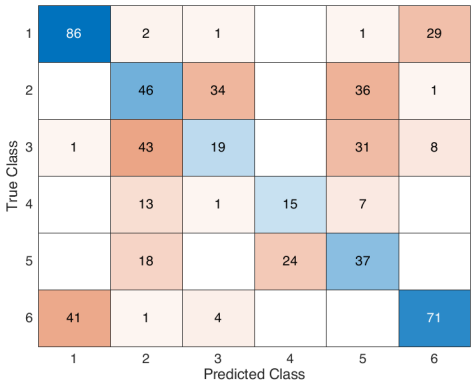


Figure 15: Confusion matrix using a 2-testers-dataset

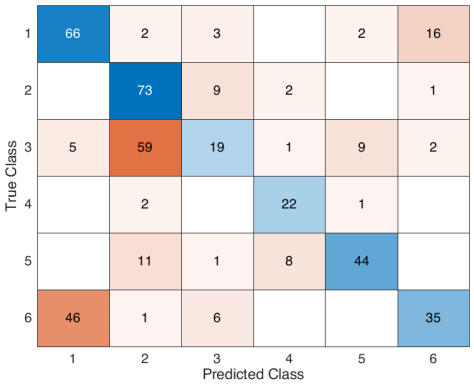


Figure 16: Confusion matrix using a 3-testers-dataset

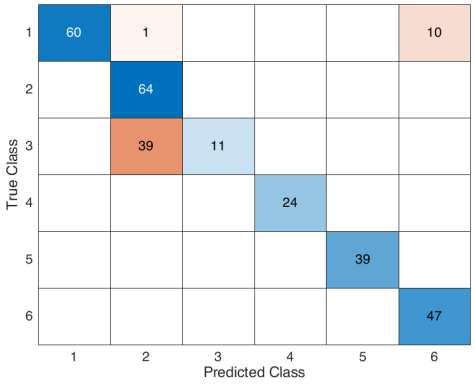


Figure 17: Confusion matrix using a 4-testers-dataset

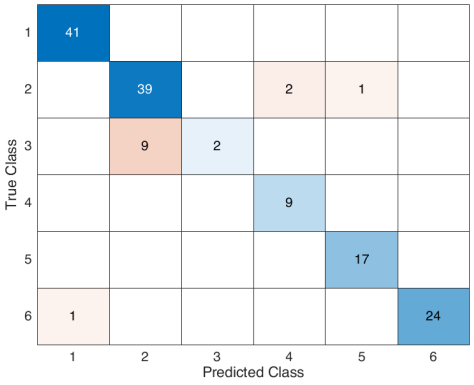


Figure 18: Confusion matrix using a 5-testers-dataset

5 Conclusion

This has been a very fun and interesting project to work on, especially for improving my Matlab knowledge and signal processing techniques. Moreover, taking the field by acquiring data personally made me feel like a working engineer.

Speaking about the experimental results, in order to get better performances a definitely larger dataset should have been used (at least 50-100 testers) and each activity should have been tracked for at least twice the time.

However the results are quite solid anyway, reaching an overall 93.9% in

the *Bagged trees* algorithm for the 6-testers-dataset. The related confusion matrix is good as well.

Signal processing procedures should have been more accurately by creating ad-hoc filters for both each device and data group. In particuar, a better filter design would have avoided phase shiftings.

Finally, it has been demonstrated how smartwatches can quite easily recognize different sport activities for health monitoring: next step, real-time recognition.