POLITECNICO DI TORINO

FACOLTÀ DI INGEGNERIA

Master degree course in Biomedical Engeneering

Master Degree Thesis

Human Activity Recognition in Apple developing environment

A preliminary study for WatchKit app developement

Supervisors:
prof. Giuseppe Calafiore

Fabiano Finocchio
matr. 240195

Anno accademico 2019-2020

A Vito e Teresa

Acknowledgements

Ho sempre provato sentimenti contrastanti alla fine di un percorso: sin dalla più tenera età mi sono sempre proiettato al passo successivo, guardando con distacco, noia e superficialità il momento presente. Per fortuna, crescendo, si impara dai propri errori o quantomeno ci si prova: in questo caso, trovo giusto e doveroso proiettarsi al futuro sì per trovare lo slancio necessario, ma sacrosanto voltarsi indietro a ringraziare una ad una le persone che mi hanno accompagnato fin qui.

Grazie in primis, ovviamente e doverosamente, a mamma e papà. Spesso ci diciamo che sono il sunto dei vostri difetti e probabilmente è vero; quel che non ci diciamo altrettanto spesso è come siate riusciti a trasmettermi la tenacia incrollabile che da sempre vi contraddistingue e non è facile, per chi sa quanto indolente io possa essere. Anche quando la speranza ha vacillato ed ero ad un passo dal cadere, il vostro esempio è stato la bussola che ha permesso ai miei passi di continuare ad incolonnarsi l'uno dietro l'altro.

Grazie a Maria Eugenia, l'altra metà del mio cuore. Così diversi, così lontani eppure mai per un secondo divisi. Un filo invisibile lega la mia anima alla tua ed è talmente forte da resistere a qualsiasi distanza. Un giorno, spero, prenderemo l'ago e lo useremo per fare un punto, così da poterci ricongiungere sotto il cielo della stessa città.

Grazie a Emiliano, per i discorsi interminabilie soprattutto per le parole non dette che nel nostro caso forse sono quelle che più descriverebbero al meglio il rapporto che abbiamo. Vent'anni che ci conosciamo e forse solo ultimamente abbiamo smesso di essere i bambini zucconi in grado di azzuffarsi in ogni competizione. Come ulteriore step di maturazione forse tre di quelle parole si possono scrivere: ti voglio bene.

Summary

This master thesis work is about the Human Activity Recognition field, which is a part of the Body Area Network. The main purpose of this thesis work is to be able to collect and analyze signals and parameters useful for recognizing human activities performed by users.

To do this, a WatchKit App was developed using an Apple Watch Series 2 combined with an iPhone 6; this app allows, by pressing a button, to collect accelerometer data (already deprived of the gravitational component), gyroscope, orientation of the device through the three Euler angles, at a fixed frequency of 50 Hz.

The app also allows, through the authorizations provided by the user, to collect data relating to the user's current speed through the device's GPS and the user's heartbeat obtained from the HealthKit package present in the iOS system. This data thus obtained are recorded through JSON format files and stored within the user's iPhone within the iOS file system app.

Data was collected on 2 male subjects in environments and uncontrolled modes, choosing a total of 8 activities such as: walking, running, indoor cycling, outdoor cycling, driving, elliptical and rowing machine, smoking.

Data is then transferred and processed on an external server and processed using Python v.2.7 and his data wrangling, visualization and machine learning packages such as Pandas, Scikit and Matplotlib.

In the processing phase, data sequences have been divided into windows of 5 seconds each, associating for each window the related heart rate and speed values (which are not possible to sample in a fixed way) through the relative timestamp obtained during the recording phase.

Subsequently, features in the time and frequency domain were extracted from the windows, so as to obtain a dataset composed of 1054 instances, each containing 214 features.

By applying a filter feature selection, in which highly correlated and low variance feature are removed from the aforementioned dataset, the number of features is reduced to 21. It was also explored in this work HAR excluding heart rate and current speed: in that case, the features considered were 18. Having thus obtained the definitive dataset and split into training and test set, some supervised model were tuned and crossvalidated with stratified k-fold, with k=5. Model selected were : K Nearest Neighbor, Support Vector Machine, Decision Tree and Random Forest. The tuned model were also employed to perform HAR on the test set and the performance of the different classifiers were compared, with the best accuracy (98%) obtained by Random Forest. The result obtained sugggest the use of Apple Watch as a quality-recording device, with high recognition rate even employing very simple classifying models.

Contents

List of Tables

List of Figures

Chapter 1

Human Activity Recognition: introduction and overview

1.1 Problem introduction

In the society in which we live,the cost of health care is often difficult to be afforded for all sections of the population. Thus, it becomes fundamental to promote and encourage people to follow a healthy lifestyle that improves them by aiming at a work of prevention, saving on the cost of care .

The percentage in elderly people is growing and there's more and more the necessity of support in daily acitivities which allows even in subjects with reduced autonomy (with motor and psycho-cognitive disabilities) to continue living safely in their home, reducing psycological distress and costs for hospitals and care structures.

Furthermore, using Human Hactivity Recognition (HAR) to detect physical activity is a possible way to encourage a more active lifestyle in the population, recognizing and encouraging physical activities, acting as a useful tool in the prevention of obesity and diseases of the cardiovascular system.

In this perspective, the study of the Recognition of Human Activities, combined with the ease of obtaining increasingly less intrusive sensors, can allow can allow to easily monitor the actions performed by the user, encouraging positive habits and, where possible, discouraging negative ones.

In this thesis work, we decided to investigate the ability to discern human activities by collecting data through Apple devices, in particular the Apple Watch which, placed on the wrist, occupies a strategic position for the detection of signals useful for HAR.

The choice of the Apple branded device is not accidental: from data coming from Strategy Analytics [1] , Apple Watch occupies 50% of the smartwatch market, having sold over 22 million devices in 2018, four times compared to the second force on the market , FitBit.

Smartwatch Brands	Market Share(%)
Apple	50,0
Fitbit	12,2
Samsung	11,8
Garmin	7,1
Others	18,9

Table 1.1. Market share in 2018 to Strategy Analytics report

Furthermore, the fact that the Food and Drugs Administration has recognized the Apple Watch Series 4 as a class 2 medical device due to its ability to recognize cardiac rhythms and detect falls is a valuable reason for investigating the possibility to perform HAR using Apple Watch as a sensing device. [2]

1.2 Human Activity Recognition: problem description and design issues

1.2.1 Preliminary definitions

Here are provided some essential definitions in order to better understand the problem domain.

Time series: x $:< x_1, x_2, ..., x_n >$ is a set of n ordered real values.

Subsequence: with refence to a Time series x of length n, a subsequence s_m^k, with $k \leq m \leq n$ is the set of $m - k + 1$ ordered real values obtained by sampling the elements of x from the k_{th} to the m_{th} such as:
$s_m^k :< x_k, x_{k+1}, ..., x_m >$.

Class: a class is a set of time series which share one or more characteristics.
Label: a label is an identificative value for a given class.

Lara et al. provides to give a formal definition of the HAR problem [3].

1.2.2 Definition : HAR Problem

Given a set $S = S_0 ... S_{k-1}$ of k time series each one from a particular measured attribute, and all defined within time interval $I = [t_\alpha, t_\omega]$, the goal is to find a temporal partition $< I_0, I_{r-1} >$ of I, based on the data S, and a set of labels representing the activity performed during each interval I_j. This implies that time intervals I_j are consecutive, non-empty, non-overlapping, and such that

$$\bigcup_{j=0}^{r-1} I_j = I$$

The main limit of this definition is that the subject in theory performs a single activity in a given time interval, implying that for each time interval I_j there is only one corresponding activity, which often does not deal with real conditions (for example it is possible to walk while smoking). Anyway, in order to simplify the HAR task, this assumption is often considered to hold.

1.2.3 Definition: relaxed HAR Problem

A new definition of relaxed HAR problem is given, introducing fixed time windows.

Given a set $W = \{W_0 ... W_{k-1}\}$ of m equally sized time windows, totally or partially labeled, and such that each W_i contains a set of time series $Si = S_i,0, ..., S_{i,k-1}$ from each of the k measured attributes, and a set $A = a_0, ..., a_{n-1}$ of activity labels, the goal is to find a mapping function $f : Si \Rightarrow A$ such that the returned label corresponds as much as possible to the activity in W_i.

1.3 General structure for a HAR application

Usually, the design of a HAR model follows the same stream of most of machine learning applications : the dataset is split into *training* and *test* set. The training phase is composed by the following phases:

1. data collection from sensors/wereable device
2. data communication with server
3. feature extraction from raw data
4. learning algorithm
5. model building

Then the recongnition model is subsequenlty employed to associate, for each time window of a data stream, where possible, an acitivity in the dataset. This kind of approach presents two main issues:

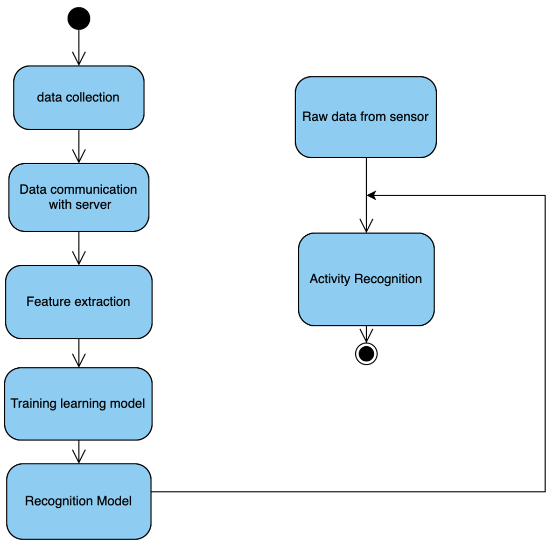


Figure 1.1. Workflow relative to learning model generation and its employ in HAR in case of remote processing

- *limited set of activities* : considering the enormous amount of the human activities which can be considered, it is fundamental to consider the right project parameters in order to achieve properly HAR on the activities that we aim to distinguish ;

- *window length*: it must be selected taking into account the parameters of *completeness of information* (each time window should contain the necessary information to distinguish activities properly) and lightness (each time window should not be too long because, in case system requires remote communication between different device, it couldn't work properly or it could require too much time).

In addition, a sliding window can be used to increase sensitivity in sudden changes of activity, as happens in everyday life. Hu et al. [4] provide a definition of sliding window in their work as a time series T of length m, and a user-defined subsequence length of n, all possible subsequences can be extracted by sliding a window of size n across T and extracting each subsequence, S_k. For a time series T with length m, the number of all possible subsequences of length n is m-n+1.

1.4 Approaches for data analysis

Lara et al. also distinguish seven main issues relative to HAR design:

1. *selection of attributes and sensors*: the most common sensor are accelerometers, due to their cheapness, low power consumption and utility in recognition. Parameters of interest are sampling frequency (usually from 10 to 100 Hz) and sensor placement. Magnetometer and gyroscope are sensor widely used as much. Also physiological and GPS signals are useful for recognition: first ones (f.e. heart rate, respiratory rate) can be discriminant between standing and sport activity, second might be used to understand user's transportation mode or used to distinguish between indoor or outdoor activity. In HAR can be also taken into account environmental parameters, such as light intensity, temperature, audio level etc.

2. *intrusiveness*: the user cannot wear a large number of sensors, otherwise, in addition to feeling less natural in performing the most common tasks, these could hinder him, altering usual movement patterns;

3. *data collection protocol*: collecting data in an uncontrolled environment would be preferable, but it usually causes a significant drop in the accuracy performance of the system; furthermore, the possible bias given given by physical characteristics, age, gender and the behavioral singularity of each individual may be taken into account;

4. *recognition performance*: it depends on the activity set, quality of training data, feature extraction method and learning algorithm. Of course including activity with similar patterns may reflect in a drop of accuracy of the HAR system; also simple activities such as walking or running are easier to recognize compared to complex activity such as practicing sports, watching TV and so on;

5. *energy consumption*: Data transmission and communication between devices or device and server is often wasteful: using short range communication protocols (f.e. Bluetooth or WiFi) allows the system to consume less energy. Data communication and data aggregation may be good strategies to reduce energy consumption. Analyzing environmental factors in HAR may be very expensive in these terms, especially in mobile application: in fact accessing to microphone, GPS may cause a remarkable increase in energy expenditure;

6. *processing*: the recognition task can be done on the device or data can be moved to an external server. Elaborating information on the device gives more scalability and robustness to the system because it doesn't rely in communication, but there are a lot of issues in terms of storage and energy

consumption. Computational power may be another aspect to take into account: in mobile devices it often presents limitations which may reflect on more elaboration time and energy consumption;

7. *flexibility*: although for many of the activities considered in the HAR there may be quite definite patterns, it is logical to think that each person performs an action in his own way. Undoubtedly an elderly person will walk differently than a young person, for example people with disabilities will present patterns that will not be easily recognizable. There are therefore two main approaches, a subject-dependent approach, in which for each user the accuracy percentages of the system are calculated and their average is considered the accuracy of the classifier, and a subject-indipendent approach, in which leave-one-out or cross-validation are used to build a valid model for all individuals.

1.5 Learning approaches for HAR

As it has been possible to perceive, the enormity of variables present in the problem suggests resolution not in a deterministic approach, but turning to learning algorithms that, once trained, are able to find solutions to the problem.

There are two main approaches to the recognition of activities: the supervised and the non-supervised approach. To overcome the limitations of this type of learning, a third method, semi-supervised learning, was introduced in the HAR case studies.

Abdallah et al. [5], in their review, provide meaningful descriptions, presenting advantages and disadvantages of these approaches. In supervised learning, an impressive amount of data is collected and assigned a label, associating a class instance with a sample, in this case an activity.

- Models generated by *supervised* learning can be distinguished between discriminative models, based on the modeling of boundaries between the classes present in the dataset (for example SVM, decision tree, kNN), generative models, which are based onprobabilistic approaches, of which the most famous example is the Naive Bayes Classifier and a hybrid approach, which combines the characteristics of the two to select the best features to use in the classification. The main limitation of the supervised approach is obviously the difficulty in finding a large number of labeled data: in a system that provides a continuous flow and in a context sensitive to sudden changes it is difficult to think how each sample can be labeled with related activity without taking into account loss of time and errors in labeling.

- to overcome the limits of the supervised approach, the applicability of unsupervised learning to this type of problem was studied; unsupervised learning consists of finding a similarity between data based on criteria such as distance. A famous example of unsupervised algorithm is the k-means clustering, in which on the basis of the distance of the k centroids relative to the k clusters selected, the sample is assigned to the closest of them.

- to take advantage of the robustness of supervised learning models but trying to reduce the disadvantages of creating a large labeled dataset, semi-supervised learning offers the possibility of combining a small amount of labeled data with a large amount of untagged data, having significant improvements in system accuracy. The assumption of indipendently identical distribution is also valid in semi-supervised learning.

1.6 Literature review

De Leonardis et al. [6] acquired 3-axis accelerometer, gyroscope and magnetometer signal using a Xsens technology sensor, MTx miniature magnetic and inertial measurement unit on 15 healthy subjects, 8 males and 7 females, considering 8 activities : sitting, standing, lying down, level walking, ascending and descending stairs, uphill and downhill walking. With a 5-s no overlapping windows, they extracted 38 features from all the 9 signal and after using a Genetic Algorithm for feature selection, they tested 5 classifiers: KNN, FeedForward Neural Network, SVM , Naive Bayes, (NB) and Decision Tree (DT) with more than 90% of accuracy.

Rosati et al.[7] used the same sensor with a 5 s window with 3 second overlap to build and compare two different datasets A and B, one obtained from features found in literature, the other considering zero crossings and derivate features. A GA was used for feature selection and KNN, FNN, SVM and DT were used for classfication with peaks of accuracy, with SVM, of 97.1 % for dataset A and 96.7 % for dataset B.

Using WHARF dataset, which consist of 12 activities, Jordan et al. [10] presented an architecture called ConvNet; segmenting signals with a time window of 5 seconds and generating a Convolutional Neural Network, made by 3 layers with 24, 48 and 32 convolutional filters. They investigated different Kernel dimension designing a ConvNet with 12x2 kernels, achieving an activity recognition rate of 79.3 %.

Xu et al. [9]in their work proposed, taking into account limited resources in battery capacity and computing power of mobile devices, developed using a deep learning tool called Caffe a SVM, obtaining different accuracy varying the number of features putting different threshold on the Pearson's correlation coefficient, obtaining accuracy values in recognition from 96,4% with 561 features , the whole feature set to 91.0% with 130 features.

Amezzane et al. [8]In their work, proposed some guide lines for mobile application in HAR: they reviewed the present literature exposing influence of parameters such as sampling frequency, sensor selection, feature set. Working on a public dataset from UCI repository which includes 6 activities , they tested Linear Discriminant Analysis (LDA), Radial Basis Function Support Vettor Machine (RBF-SVM), K Nearest Neighbor (KNN) and Random Forest(RF), trying different feature selection techniques such as Consistency-Based Filter (CBF), Recursive Feature Elimination (RFE) and Medium Decrease in Accuracy (MDA). They compared results in terms of accuracy, macro-precision, macro-recall, macro- F1 measure and training time.

In [11], Suto et al. investigated performance of feature selection methods utilizing WARD database with a 32-samples window with 50% overlap. Most common features in literature were extracted in time, frequency and correlation domain and Artificial Neural Network(ANN), KNN and DT were tested trying different feature set configurations for each subject, achieving the goal of HAR recognition.

In the next chapter, the iOS app developed to record data from Apple Watch and iPhone will be described.

Chapter 2

Motion Data Logging WatchKit App

In order to be able to obtain data from a closed environment such as iOS, it was necessary to develop an app that would allow the developer to record data from sensors equipped on smartwatches.We developed such an App, to which we refer to as Motion Data Logging, with the purpose of allowing the creation of a database for Human Activity Recognition: the choice of developing on iOS ambient is mainly the large diffusion of iOS devices on the market especially in that of smartwatches which, despite the selling numbers at the moment are not comparable with other devices such as smartphones, now present in everyone's daily habits, have the potential to be one of the strongly impacting new technologies.

The application, called Motion Data Logging, has a double interface, one relating to the smartphone and one to the smartwatch associated with it.

The data logging con be performed through the developed app in a simple way: by pressing the "Start" button it is possible to start recording the data flow from the accelerometer, gyroscope, magnetometer, GPS and from the optical detection sensor placed on the body of the smartwatch, useful for measuring the heart beat; when the "Stop" button is pressed, the recorded data is saved to a JSON file and sent to the iPhone in background, where it is accessible via the File system app.

Moreover, the interface on the iPhone side allows, through a text box, to insert a label in order to be able to associate each JSON file with the recorded activity, thus allowing to proceed with the training of models derived from supervised and semi-supervised algorithms.

It is also possible, without recording data relative to the heart rate because of the absence of the photoplethysmograph, to log the iPhone sensor data and use them for HAR .

More details about the design of the app will be given in the following sections.

2.1 Hardware

2.1.1 Accelerometer, gyroscope and magnetometer

Unfortunately, due to the limited information available on Apple's website, it was not possible to obtain the technical specifications regarding the sensors equipped on these devices.

However, from websites dealing with the subject, it is reported that from the iPhone 6 onwards Apple mounts two accelerometers, a Bosch BMA280 and MPU-6700 six-axis accelerometer from InvenSense, on the device with maximum sampling rates of 2000 *Hz* and 4000 *Hz*, respectively[12].

Still, on specific forums related to the subject, the sampling frequency of 100 *Hz* seems to be the maximum frequency for which the data obtained are recorded with a certain stability, but unfortunately no more reliable sources have been found.[13]. However, considering the small dimensions of this type of sensors, it can be assumed that for a device like the one used in this thesis work the specifications will not differ much. In any case, it was decided to use project specifications that would allow a stable collection of data. At the beginning of this master thesis work, the possibility to record data only from the iPhone was explored : then the effort was focused on creating a product that would have allowed the collection of data from it. Apple Watch, considered as a more reliable sensing device because of the number of movement patterns that can be detected which can be much greater than using a cell phone; moreover, its low intrusiveness allows the user to carry out the activities in a natural way, which may not happen by placing the mobile phone on an arm, for example, through an armband.

Regarding gyroscope and magnetometer, no specific technical information was found. The same for sensors equipped on Apple Watch

2.1.2 Heart rate measure system

Figure 2.1 shows the optical sensor on the back of the Apple Watch series 4, which is the latest version available on the market, as follows:

- four green light LEDs diods;
- two infrared LEDs;
- an electrode placed on the digital crown, which can be used to measure health parameters by putting your finger on it. This procedure allows to retrieve heart rate and other parameters in a more direct way;

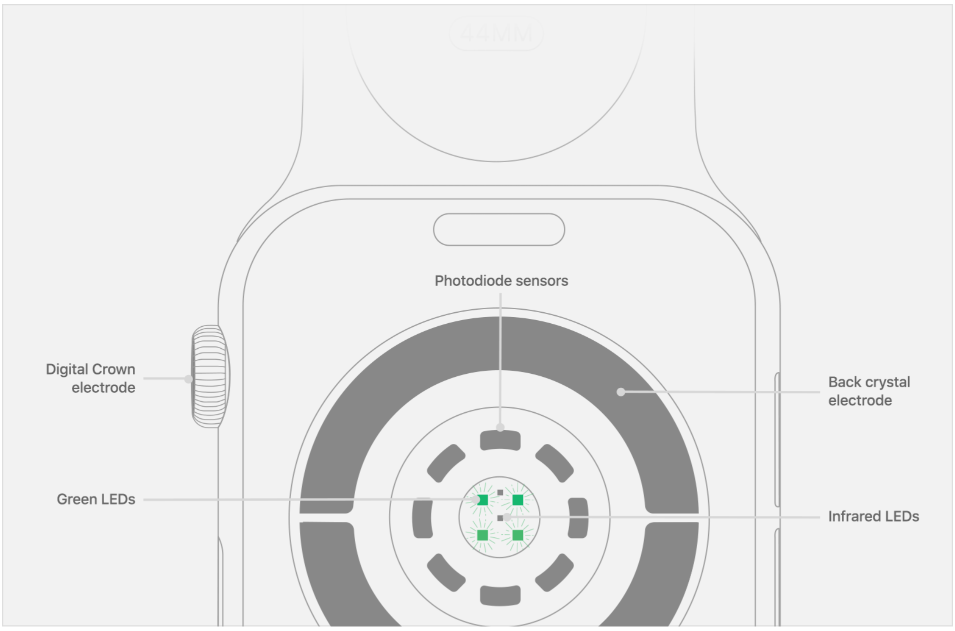


Figure 2.1. Image is relative to the diods and leds equipped on Apple Watch series 4, available on Apple website

- two semicircular electrodes, which carry out the measurements with the procedure described in the previous point, allow the closure of the finger-wrist electrical circuit, guaranteeing the timeliness and accuracy mentioned above.

The Apple Watch series 2, which has been used in this master thesis work, doesn't use electrodes, relying for heart rate measurement exclusively on photoplethysmography; it consists in emitting pulsations in the spectrum of the green. This light wavelength is absorbed by blood because of its strong composition of hemoglobine, which absorbs wavelenghts in the green spectrum as showed in figure 2.4.[14]

With each beat the flow that circulates increases and consequently the greater the absorption in the green spectrum of the tissues underlying the Apple Watch and vice versa, as heart rate decreases, absorption will be lower.

These measurements are carried out in the background by the app, thus not allowing a fixed sampling of the heartbeat, which can make it difficult to associate a sampling rate to the cardiac measurement.

Obviously a similar argument can be made regarding the reading of data from GPS, also carried out in the background and therefore not subject to a well-defined sampling frequency.

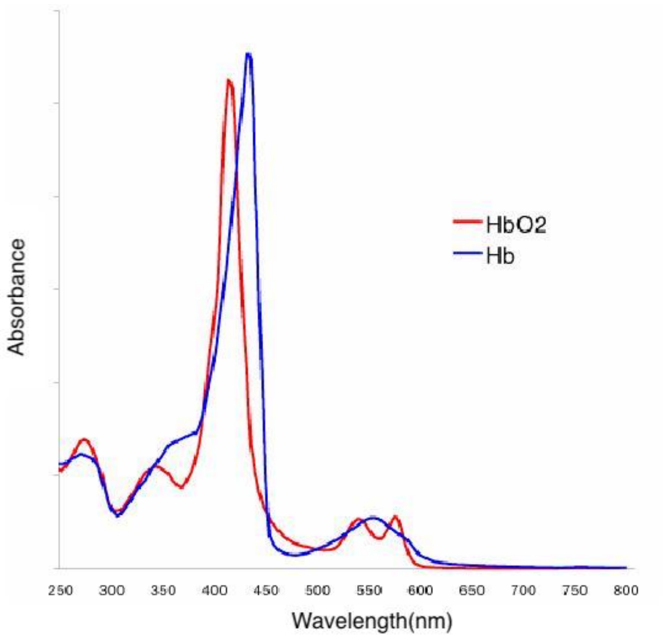


Figure 2.2. Absorption spectrum of hemoglobin in the oxygenated and deoxygenated form

2.2 App design

2.2.1 Swift programming language

The app was developed by writing the code in Swift, which is Apple's native language, used in the construction of all its products. The version of the language used is 5.0.

Apple describes the Swift language as a powerful, easy to learn and safe programming language; his major strength is the API which provides an interoperability with objective-C, which is a programming language best known and most widely used, from which Swift directly derives.

2.2.2 Xcode : the environment for iOS app developing

Apple provides developers with the Xcode development environment, available for free on the Apple Store. The great strength of this development environment is the ease in the top-down approach in the implementation of an application: Xcode easily allows to create a graphical interface for the device for which it is designed, associating the code to the various elements present in it.

Another strength of XCode environment is the possibility of obtaining an aesthetically pleasing interface for any version and model through the imposition of adaptive constraints on the margins of the secure area; regardless of the device for which it was designed, it allows a remarkable scalability within the products (iPhone, iPad and Apple Watch of all generations) of the parent company.

Furthermore, Xcode contains a powerful emulator called Simulator which allows the developer to carry out the debugging and testing phase directly from the graphic interface of the emulated device on the PC.

On the contrary, one of the major flaws of the apple development environment is undoubtedly the poor open-sourcing of the system. To be able to freely test devices and make the application accessible on the App store, the developer must be in possession of the Apple developer license called the Apple Developer Program, available for 99$ a year.

Without it, you can still develop and test your application on the Apple device, but there are limitations:

- there is a limitation on the number of applications developed by the individual developer. If he exceeds the number, it will not be possible to create a new project before 7 days;
- the application remains available on the device on which testing takes place for a few days: afterwards, it will no longer be possible to test it or use it directly on the device unless it is reinstalled directly by Xcode.

2.3 Software design

MotionDataLogging app is designed in order to create an environment in which, using Apple Watch, is possible to collect data to perform HAR. The class diagram explains how both controllers are linked by a WCSession object, instantiated in the iPhone interface, which is responsable of communication between devices. In the following paragraphs it will be explained in detail how the app works.

2.4 Graphical User interface

2.4.1 watchOS App

The graphical interface is designed to be as intuitive and usable as possible: in fact, it has two buttons, Start and Stop, which with an appropriate color coding (green for the Start, red for the Stop) immediately give the idea of their functions.

When the first is pressed, the methods for recording the values of the parameters of interest are recalled, while at the pressure of the second the recording is

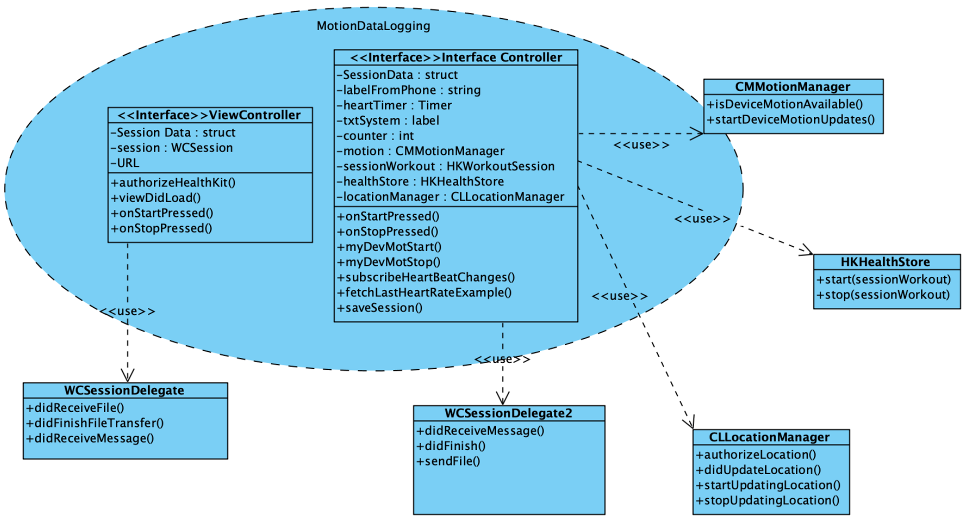


Figure 2.3. Class Diagram: the ViewController is the controller relative to the iPhone view; the InterfaceController is the controller relative to the AppleWatch

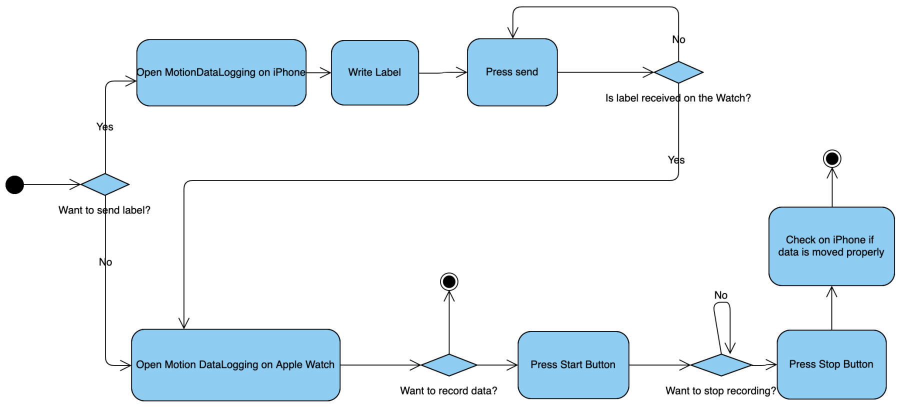

Open MotionDataLogging on iPhone
Write Label
Press send
No
Is label received on the Watch?
Yes
Yes
Want to send label?
No
Open Motion DataLogging on Apple Watch
Want to record data?
Press Start Button
No
Want to stop recording?
Press Stop Button
Check on iPhone if data is moved properly

Figure 2.4. The diagram follows the steps that the user may pursuit in order to obtain data from the AppleWatch

interrupted and the procedure for transferring the data collected on the iPhone is started and from there the storage procedure is started .

The interface also presents a timer, which allows the user to monitor if the system is recording and the duration of the log session.

The label at the bottom gives an indication of the operation of the system: if the label from phone is received, if stopped properly to record, and if the file is moved to iPhone with success.

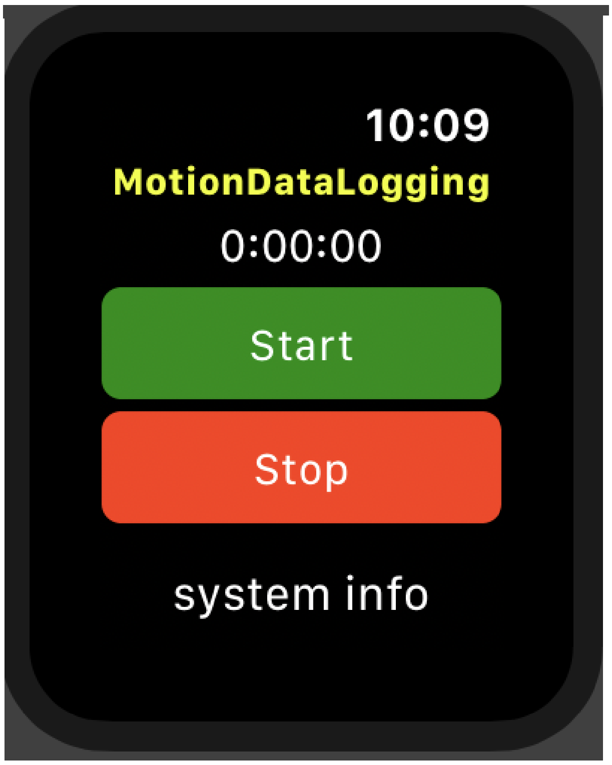


Figure 2.5. User interface for MotionDataLogging Watch App

2.4.2 iOS App

A text box has also been added, which allows the user to optionally insert a label for the following session: this label will remain active until the user decides to insert a new one. This choice is justified by the fact that often in the operation data logging you find yourself recording more consecutive times, the same task: to avoid loss of time, that choice has been made. In the same way, the user interface on the iPhone side tries to reproduce the Apple Watch visual aspect with the same two buttons that allow you to start and stop data recording on the smartwatch remotely; in addition, there are two text fields describing the operations performed on the iPhone side and the Apple Watch side:

- start recording on watch side;
- stop recording on watch side;
- transfering file;
- file moved to documents URL;
- impossibility of sending label.

The following paragraphs explain in detail the functions, methods and structures used within the application.

2.5 Software developement

This section will describe the packages used by the app to perform the following functions:

- retrieve data from accelerometer, gyroscope and magnetometer using the CMMotion Manager class and his functions and attributes;
- retrieve data relative to longitude, latitude, current speed from CLLocation manager class and his functions and attributes;
- retrieve biometric data relative to the Health package in Apple, such as heart rate, respiration rate and so on;
- provide a struct called SessionData which provides the structures to store data;
- store the SessionData into the app utility folder, accessible through the system iOS app File, with the class Storage, his method and functions;
- provide communication between devices with the WCSession class and his methods;

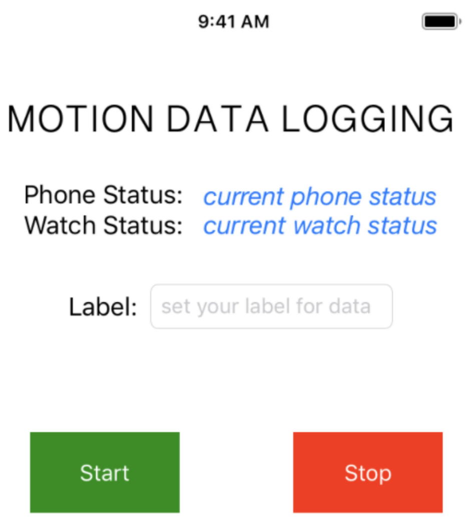

9:41 AM
MOTION DATA LOGGING
Phone Status: current phone status
Watch Status: current watch status
Label: set your label for data
Start
Stop

Figure 2.6. User interface for MotionDataLogging app for iPhone

2.5.1 CMMotionManager

According to Apple documentation[15], the CMMotion manager is declared in the controllers. The class provides methods to record:

- the istantaneous acceleration of the device in three-dimensional space with the methods relative to the **accelerometer**;
- the istantaneous rotation speed relative to the three axes with the methods relative to the **gyroscope**;
- the device orientation relative to Earth's magnetic field with **magnetometer**;

- device-motion data, which consists of CoreMotion **processed attributes**: it allows to obtain processed data such as device's attitude, rotation rate, calibrated magnetic fields, the direction of gravity, and the acceleration the user is applying to the device. Those data are obtained by CoreMotion algorithms which combine the three sensors mentioned above.

To take full advantage of the sensors and processing capabilities of Apple devices, we chose to gather data among those attributes. 9 time series were collected from this data :

1. User's acceleration relative to x axis in $\frac{m}{s^2}$;
2. User's acceleration relative to y axis in $\frac{m}{s^2}$;
3. User's acceleration relative to z axis in $\frac{m}{s^2}$;
4. Angular speed relative to x axis in $\frac{rad}{s}$;
5. Angular speed relative to y axis in $\frac{rad}{s}$;
6. Angular speed relative to z axis in $\frac{rad}{s}$;
7. *Pitch*, device's orientation relative to x axis in *rad*;
8. *Roll*, device's orientation relative to y axis *rad*;
9. *Yaw*, device's orientation relative to z axis *rad*;

It was decided to take into account the acceleration without its gravitational component, automatically removed by the provided Apple API. The choice to exclude the gravitational component in the data related to acceleration is justified by the fact that it does not provide useful information for the discernment of movement activities, while on the contrary it can be useful in detecting static activities such as sitting or lying. Micucci et al.[16] in their work choose to exclude gravity from raw accelerometer signal with a low-pass Butterwoth filter because gravity is assumed to have only low frequency component.

According with Apple documentation [17] with the function `myDevMotStart`:

1. the availability of Device Motion is checked;
2. the reference frame is fixed among:
    - `XArbitraryZVertical` describes a reference frame in which the Z axis is vertical and the X axis points in an arbitrary direction in the horizontal plane.

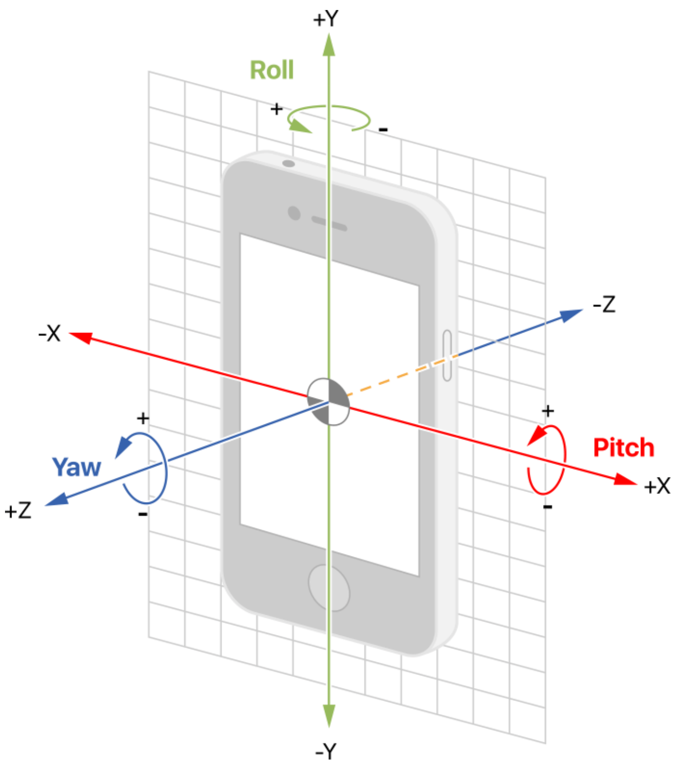


Figure 2.7. Pitch, Roll and Yaw in device reference frame

- `XArbitraryCorrectedZVertical` describes the same reference frame as xArbitraryZVertical except for the one of the magnetometer, when available and calibrated, is used to improve long-term yaw accuracy. Using this constant instead of xArbitraryZVertical results in increased CPU usage.
- `XMagneticNorthZVertical` describes a reference frame in which the Z axis is vertical and the X axis points toward magnetic north. Note that using this reference frame may require device movement to calibrate the magnetometer.
- `xTrueNorthzVertical` dscribes a reference frame in which the Z axis is vertical and the X axis points toward true north. Note that using this reference frame may require device movement to calibrate the magnetometer. It also requires the location to be available in order to calculate the difference between magnetic and true north.

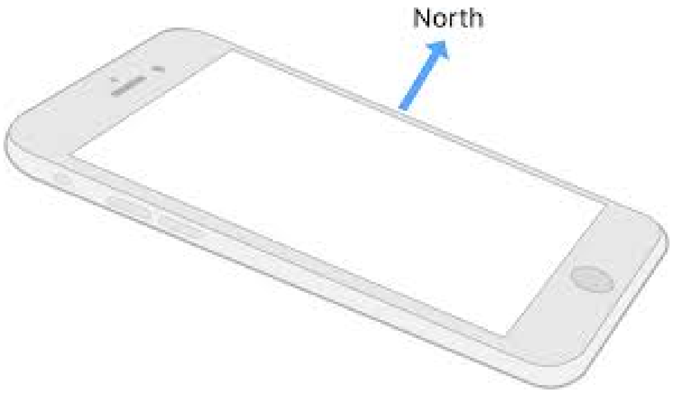


Figure 2.8. Example of reference frame with North set

To avoid further CPU usage and battery consuming, `XArbitraryZVertical` was selected.

3. sampling frequency is set to 50 *Hz*

4. a check on DeviceMotion activity status is performed ;

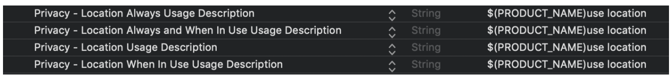
5. data are stored in a `SessionData` struct, ready to be exported.

With the funcion `MyDevMotStop` the MotionManager stops the acquisition.

2.5.2 CLLocationManager

To retrieve data relative to GPS, an instance of CLLocationManager, the Swift class deputed to gather data from GPS, is created in the Apple Watch controller interface.

Due to the regulations imposed by the respect of users' privacy, in order to allow the use of position data, it is necessary to modify the permissions specifying the use of the position in the `Info.plist` file; so the App will ask the user the permission of using his personal GPS data.



Privacy - Location Always Usage Description	String	$(PRODUCT_NAME)use location
Privacy - Location Always and When In Use Usage Description	String	$(PRODUCT_NAME)use location
Privacy - Location Usage Description	String	$(PRODUCT_NAME)use location
Privacy - Location When In Use Usage Description	String	$(PRODUCT_NAME)use location

Figure 2.9. permission required to authorize GPS data retrieving on an iOS App

At the start of the application through the delegation to the controller and through the methods `requestAlwaysAuthorization` and `requestWhenInUseAuthorization` The AuthorizeLocation function allows data to be recorded within the Struct `SessionData`.

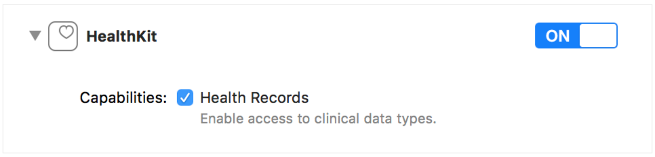
The `CLLocationManager` [18]Class allows to register data relative to :

1. **coordinate**, longitude and latitude;
2. **altitude**, and derivates;
3. **speed and course**, the speed of the user and the direction measured in degrees and relative to due north.

To try to save space to make communication easier between Apple Watch, only current speed and his relative timestamp have been selected to be stored in `SessionData` struct.

2.5.3 Health Store and HWorkoutSession packages

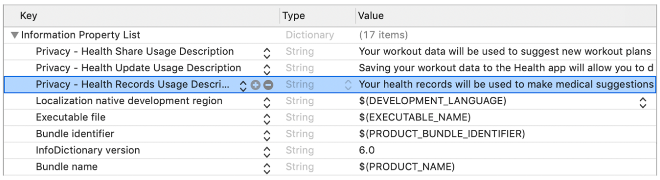
The Health app serves as a central repository for health and fitness data in iOS. With the user permission, apps built with HealthKit can communicate with the Health app to access and share information. To build an app with HealthKit, first of all Health Records capablities must be enabled [19].


HealthKit ON

Capabilities: ✓ Health Records
Enable access to clinical data types.

Figure 2.10. Enabling health records capabilities in iOS developing according to Apple documentation

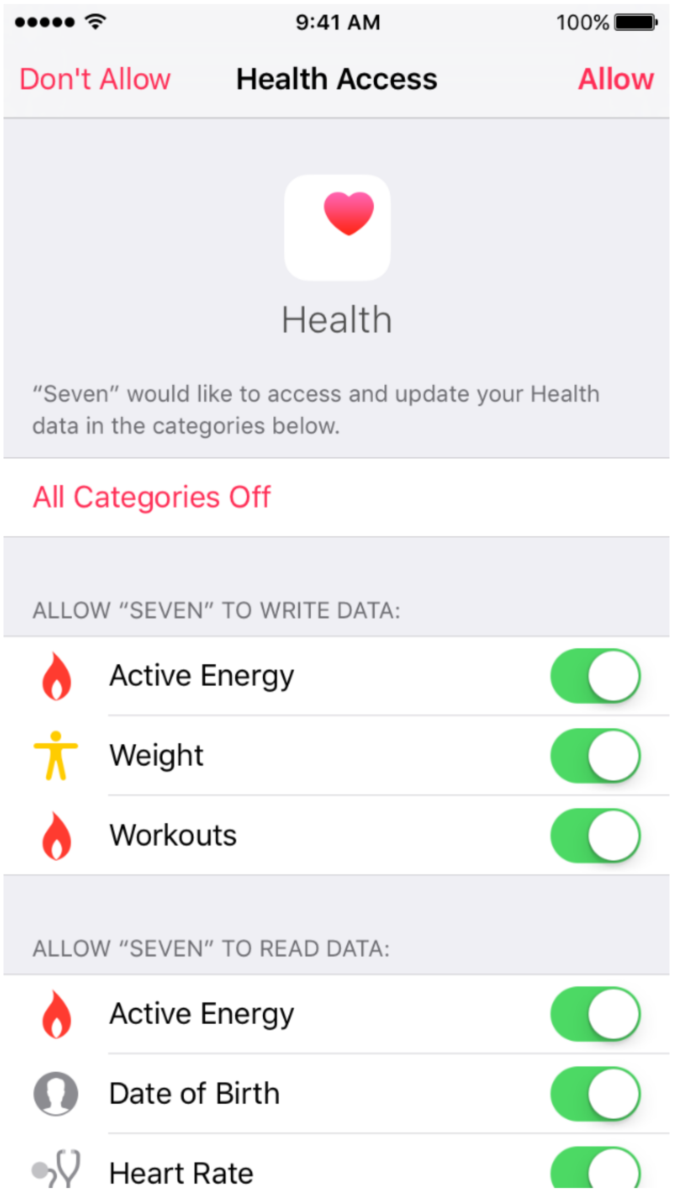
Then the permission to manage such sensitive data must be enabled into the `Info.plist` file



Key	Type	Value
Information Property List	Dictionary	(17 items)
Privacy - Health Share Usage Description	String	Your workout data will be used to suggest new workout plans
Privacy - Health Update Usage Description	String	Saving your workout data to the Health app will allow you to d
Privacy - Health Records Usage Descri...	String	Your health records will be used to make medical suggestions
Localization native development region	String	$(DEVELOPMENT_LANGUAGE)
Executable file	String	$(EXECUTABLE_NAME)
Bundle identifier	String	$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_NAME)

Figure 2.11. Example of authorization required for managing health data

So, to gather and store information about any health data, an instance of `HKHealthStore` class has been created in Apple Watch Interface Controller.


9:41 AM 100%

Don't Allow Health Access Allow

Health

"Seven" would like to access and update your Health data in the categories below.

All Categories Off

ALLOW "SEVEN" TO WRITE DATA:

Active Energy

Weight

Workouts

ALLOW "SEVEN" TO READ DATA:

Active Energy

Date of Birth

Heart Rate

Figure 2.12. example of user authorization interface required to an app which integrate HealthKit(source: www.apple.com)

Through the Controller function `authorizeHealthKit` the `HKSampleType` which the iOS app is allowed to write and read through the `HKHealthStore` method `requestAuthorization` were defined. For MotionDataLogging, only heart rate has been selected.

HealthKit uses `HKObjectType` subclasses to divide the different types of data stored in HealthKit. According to Apple developer documentation [20]:

- `HKCharacteristicType` represents data that doesn't change over time (for example, date of birth);
- `HKQuantityType` represents biological samples that contain a numeric value (for example heart or respiratory rate);
- `HKCategoryType` represents samples that contain an option from a short list of possible values (for example sleep analysis);
- `HKCorrelationType` represents complex samples that contain different quantity or category samples (for example, a food sample that includes a number of nutrition samples);
- `HKWorkoutType` represents a workout and its associated data;
- `HKDocumentType`, and `HKSeriesType` represent specialized data types.

There are three ways to access data from HealthKit Store, according to Apple documentation[21]:

- **Direct method calls** : the HealthKit store provides methods to directly access characteristic data, such as blood type, or birthday date;
- **Queries** : they return the current snapshot of the requested data from the HealthKit store;
- **Long-running queries**: these queries continue to run in the background and update your app whenever changes are made to the HealthKit store;

Since it is of our interest to continuously obtain the data relating to the heartbeat of the user during the acquisition, it was necessary to implement a long running query inside the controller which:

- Through the `subscribeToHeartBeatChanges` function prepares the query to be performed in such a way to obtain the last heartbeat sample recorded by the device;
- The `fetchLatestHeartRateSample` function executes the query;

In order to make the query run in the background in a stable way, it was necessary to add a Workout session, which by default allows the recording of data related to the HKStore, during which, through the aid of a timer, the functions are periodically recalled. In absence of the Workout session, during the standby the recording of data related to inertial sensors was also interrupted: it was therefore essential to implement it to guarantee a correct data recording process.

Through the `startWorkOut` function, recording GPS and health data functions are called and data retrieved into the `SessionData` struct.

2.5.4 Storage class

To allow easily the storage in the document folder, the Storage class was imported [22].

Through the `getURL` method, the app documents folder in the iPhone side is selected as the location where data are saved.

Through the `store` method, called from the `SessionData` struct, data are saved into the desired URL.

2.5.5 JSON format

JSON(Javascript Serialized Object Notation) is a markup language, completely independent of the programming language, but uses conventions known to C language family programmers. This feature makes JSON an ideal language for data exchange[23].

JSON is based on two structures:

- A set of **name / value pairs**. This could be realized as an object, a record, a struct, a dictionary, a hash table, a list of keys or an associative array according to the language in which the JSON file is imported;

- An ordered list of values. In most languages this is accomplished with an array, a vector, a list or a sequence.

These are universal data structures. Virtually all modern programming languages support them in both forms. A data-exchange language which interacts with different programming languages must be based on these structures.

The JSON structure is divided in objects which are unordered series of names / values. An object starts delimited into curly braces. Each name is followed by colon and the name / value pair are separated by comma. For example : $\{"jsonObject : "name", "date" : "21/01/2019"\}$ Objects may contain :

- **values** : can be integers, doubles, string, boolean or null;

- **arrays** : sorted set of values, included in square brackets with values separated by comma.
- **objects**;

Objects can be nested; JSON does not support octals and hexadecimal formats.

2.5.6 Session Data struct

To allow storage in JSON data format, a widely used markup language in client-server communication applications, `SessionData` was defined as a `Codable` struct: this allows the export of objects and structs in the swift language in JSON files, ready to be exchanged.

The `SessionData` provides data structures to save locally :

- user acceleration data through the three axis: x, y, z ;
- gyroscope data through the three axis: x, y, z;
- device attitude along the three angles pitch, roll and yaw;
- heartRate with relative timestamp to reallocate the value in the correct time window;
- GPS measured speed with relative timestamp to reallocate the value in the correct time window;
- the label associated with the acquisition;
- the device (iPhone or Apple Watch) used for acquisition;
- the acquisition frequency;
- the timestamp relative to start acquisition and stop acquisition;
- the string relative to start acquisition and stop acquisition;
- the duration of acquisition in seconds;

This struct also provides `get` and `set` methods and a `saveData` method, which uses the upmentioned `Storage` class and his `store` method.

2.5.7 WCSession delegate

To provide communication between Apple Watch and iPhone in the more efficient and less energy consuming way, Apple provides the `WCSession` class, which is designed to use different communication strategies[24]:

1. if devices are available and in range, Bluetooth connection is established. This connection has the highest priority due to its low effort in energy consumption and stability;
2. if Bluetooth is not available but the devices are both in the same local Wi-Fi network, devices are connected using Wi-Fi protocol;

There are several communication ways between iPhone and Apple Watch provided by this class according to Apple documentaton [25]:

- `updateApplicationContext`
- `sendMessage` and `sendMessageData` transfers data to a reachable counterpart. These methods are intended for immediate communication between your iOS app and WatchKit extension.
- `transferUserInfo` transfers a dictionary of data in the background. The dictionaries are queued and transfers continuously when the current app is suspended or terminated.
- `transferFile` which sends the specified file and optional dictionary to the counterpart in background;

All the methods were tested in the developing phase, and only `transferFile` guarantees a proper data delivery for file in JSON format.

Relying on the fact that both devices are active and reachable, the `SendMessage` is used in order to:

- allow starting and stopping data collection from Apple Watch pushing the Start and Stop button on iPhone user interface;
- allow status communication between the devices, so the user can monitor the status on labels present on both interfaces. In details:
  - can monitoron Apple Watch side when the label is received and JSON file has finished transferring;
  - on iPhone side, communicate with AppleWatch when JSON file has been received;

In both iPhone and Apple Watch controllers, loading the view, a `WCSession` object is created and the controller, through delegation, is allowed to exploit the `WCSession` methods and through the `activate` method, the connection is established.

Chapter 3

Methods

In this chapter, it will be discussed about methods for achieving HAR in a supervised ambient.

The first section describes how the data has been collected using MotionDataLogging app.

The second section describes the process of data wrangling and aggregation useful to create a dataset.

The third section is about feature calculation from data and signals.

The fouth section presents the methods used for feature selection.

The fifth section presents the models used to achieve HAR.

3.1 Data collection

To provide data , two subjects contribute to the collection. The subjects involved are described in the table below.

Subject	Gender	Age
1	Male	26
2	Male	27

Table 3.1. Subjects involved in data collection

Data were all collected from Apple Watch, placed on the left wrist of the subject during the acquisition.

Data related to all activities were acquired in uncontrolled way, except for those who requires a support or a machinary.

For this analysis, 8 activities were performed:

1. **walk**: performed by subjects as fast, medium and slow;
2. **run**: performed by subjects as slow and medium;
3. **cycling** outdoor by subjects as slow and medium;
4. **cycling** in stationarky bike by subjects as slow and medium;
5. **car** driving performed in urban streets and highway;
6. **smoke**
7. **rowing indoor**
8. **elliptical machine**

For acquisition in uncontrolled way, we intend that all activities were performed without any constriction, introducing some noise, for example trying to move the left arm, where the Apple Watch was placed, during the data acquisition.

We tried to reproduce the normal performance of human activities in the best possible way. Indeed, in the HAR definitions given in the introduction it is assumed that at each time window only one activity is performed: this assumption does not reflect the real conditions under which the activities are carried out. In fact, during data acquisition, we tried not to perform activities defining a "standard" movement pattern, but we tried to reproduce activities as they are performed in common life situation.

This protocol described in *figure 3.1* was used to collect the data.

Since background transfers of large amounts of data were not always successful in case of long acquisitions, in order to have a good compromise between acquisition continuity and data transfer speed, it was decided to maintain the length of the acquisitions of duration up to 3 minutes. In this way, it is possible to obtain the acquisition file on the iPhone Files app in times in the order of some seconds, so as to be able to proceed with the collection of more data in a relatively short time.

All data were sampled with an sampling frequency of $50Hz$. This choice is justified by the sampling frequency found in literature: In [8], it was observed that the most used sampling frequency which allows to obtain good recognition performances are 20, 32 and 50 *Hz*; also 100 *Hz* has a significant use in literature: Siirtola et al. [28], Heng et al. [29] in their work use this sampling frequency, as well as [6]. Always in [8] it is enlighted that frequency in range from 2 to 150 *Hz* have been widely employed in HAR works. The choice of 50 *Hz* has been made in order to keep a good compromise between completeness of information and the ability of devices to transmit data: In fact, using higher sampling frequencies, problems could have been encountered in transferring data to the background, thus not allowing a correct collection.

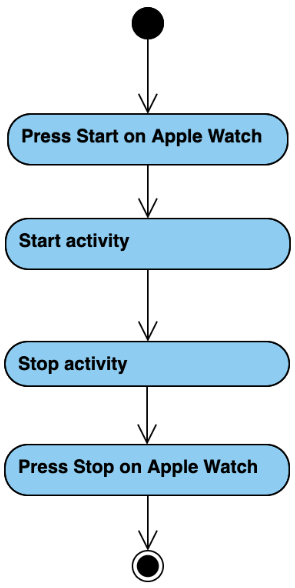


Figure 3.1. Protocol of data acquisition

3.2 Data aggregation and data wrangling

A folder has been created on the elaboration server in which all JSON files have been stored.

A pipeline was created in Python v.2.7 in order to automize reading of JSON file and obtain a Pandas DataFrame.

3.2.1 from JSON files to dataframe

Figure 3.1 describes the pipeline used to convert a series of Json file to a DataFrame on Pandas, a useful Python library which allows easily data manipulation and wrangling in order to proceed with data analysis.

During the course of the pipeline, the windowing operation was carried out as described in the following passages:

- The `createDataset` function opens every JSON file in the dataset folder calling `timeWindowingJsonAcquisitionTimestamp` for every file which:

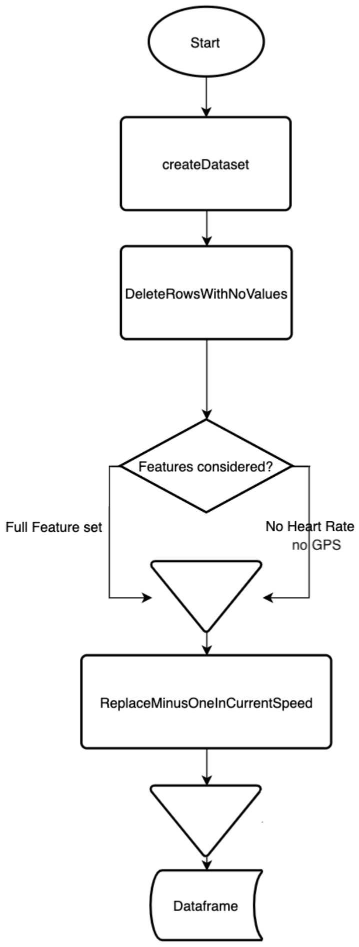


Figure 3.2. Workflow of algorithm for Dataframe creation

– based on startDataAcquisition and stopDataAcquistion Timestamp recorded on SessionData, the first and the last 4 seconds of every acquisition were removed: the justification for this procedure is that data acquisition was often performed by manually pressing the Start button on the app interface for WatchOS, just as the end of the recording was done by pressing the Stop button. Consequently, in the initial and final windows there was undoubtedly the presence of non-informative signals and to avoid to use them in generating learning models.

  Based on the new start and stop, a non-overlapping 5 second window was used to split up acquisition and generate samples.

  – using the new start and stop and the timeStamp acquired relative to GPS Current Speed and Heart Rate, every acquisition of those two features is assigned to the relative time-Window.

  A dataframe with the single JSON file data is created.

- the `concatenateSamples` function joins every single-JSON dataset in order to create the complete dataframe.
- the `removeRowsWithNoValues` deletes from the dataframe every window in which there's no value of current speed or heart rate ;
- the `ReplaceMinusOneInCurrentSpeed` function arises from the observation of the dataset: in fact, in case of of failure to gather current speed values , the CLLocationManater returns the value -1, clearly impossible to obtain in speed measurement. Therefore this strategy was adopted:
  – if the mean of current speed in the window is negative, then the row is discarded;
  – else, the -1 value is replaced with the mean of the time window current speed, calculated excluding the -1.

Dataset	Observations
full	1213
after RemoveNoValue	1103
after ReplaceMinusOnes	1054

Table 3.2. Number of observation in dataset after data wrangling and manipulation operations

3.2.2 Windowing

The choice of the window length was not accidental: analyzing the literature identify examples of similar duration: Altun et al. [30], as well as in [6], use 5 *s*, but numerous examples of similar length can be found of window duration in range from 1 to 4 seconds.[31]

Table 3.2 describes some window type found in the literature analysis. From these examples, the choice used in this thesis work is fully justified.

Work	Window length (s)	Overlapping (%)
[30], [6]	5	0
[16]	3	0
[29]	1.5	50
[11]	2	50

Table 3.3. Windowing in literature review

3.2.3 Dataframe structure

The dataset thus obtained is composed as described in the table 3.4.

Signals	AccelerationX, AccelerationY, AccelerationZ, GyroX, GyroY, GyroZ, Pitch, Yaw, Roll, HeartRate, CurrentSpeed
Observations	1054

Table 3.4. Final dataset structure

To explore the relevance of the analyzed signals, taking into account the considerable battery consumption that involves the acquisition of speed via GPS or heart rate, different combinations of signals have been explored so as to be able to compare the results with or without these last two. Moreover, those data are considered pretty sensitive in privacy terms. The combinations are described in table 3.5.

Dataset Name	Feature set
A	complete feature set
B	no Heart Rate and Current Speed in feature set

Table 3.5. Combination of time series set considered

3.3 Feature calculation

From the raw dataset of the signals in time domain, features have been extracted in the time domain, frequency and taking into account the correlation of the signals acquired between the axes. The choice of those features is justified by [11], which provides an exhaustive description in his work about features in HAR. Most of the features in time and frequency are considered also in [6], [7].

3.3.1 Features in time domain

From all the signals extracted from the database, the following features have been extracted in the time domain:

- **mean** : the mean of the signal

$$\bar{x} = \frac{\displaystyle\sum_{i=1}^{N} x_i}{N}$$

- **standard deviation**

$$\sigma = \sqrt{\frac{\displaystyle\sum_{i=1}^{N}(x_i - \bar{x})}{(N-1)}}$$

- **variance** :

$$\sigma^2 = \frac{\displaystyle\sum_{i=1}^{N}(x_i - \bar{x})}{(N-1)}$$

- **skewness**: a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive, with *tail* on the left or negative, with *tail* on the right, or undefined.

$$s_k = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^3}{\sigma^3}$$

- **kurtosis**: deviation from the normal distribution, with respect to which there is a greater flattening (*platicurtic* distribution) or a greater elongation (*leptocurtic* distribution)

$$k = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^4}{\sigma^4}$$

- **maximum** : max_x is the highest value that can be found in the observation;
- **minimum** : min_x is the smallest value that can be found in the observation;
- **25 percentile** :*Q1* : data relative to a value *q1* which divides a set of n linearly ordered data, so that the number of values lower than *q1* constitutes the 25% of n;

- **75 percentile** : *Q3* : data relative to a value *q3* which divides a set of *n* linearly ordered data, so that the number of values lower than *q3* constitutes the 75% of *n*
- **interquartile range** :
$$IQR = Q3 - Q1$$
- **peak-to-peak** :
$$PTP = max_x - min_x$$
- **mean crossing** :the number of time that signal crosses the mean value.
- **SMA** : is the normalized sum of accelerometer components.

$$SMA = \frac{1}{T}\sum_{t=1}^{T} |a_x(t)| + |a_y(t)| + |a_z(t)|$$

3.3.2 Features in frequency domain

From all the signals relative to acceleration signals and angular speed from the database, the following features have been extracted in the frequency domain. A power spectrum density estimation (PSD) of acceleration and angular speed along the three axis was computed using Welch periodogram, which is defined as :

$$P_{xx} = \mathcal{F}\{x(t) * x(-t)\} = X(f) * X(\text{-}f) = |X(f)|^2$$

The Welch method allows the presence of overlap between two adiacent windows up to 50% of windows length to have better spacial resolution. In this study, the overlap was set to 50% of the length of the window (5 s). P_{xx} is the power spectrum estimate and f the frequency vector, with $0 \leq f_i \leq f_{sampling}/2$, according to Nyquist sampling theorem[26].

From PSD calculation, several features were computed:

- **mean frequency of PSD**:
$$MNF = \frac{\int_{i=1}^{N} P_{xx}(f_i) * f_i}{\int_{i=1}^{N} f_i}$$
- **median frequency of PSD**:
$$MDF = \int_{i=1}^{M} P_{xx}(f_i) * f_i$$

such as

$$MDF = \frac{\int_{i=1}^{N} P_{xx}(f_i) * f_i}{2}$$

with M $\neq$ N

- **principal frequency of PSD** :

$$PF = f(max(P_{xx}))$$

, with $f \neq 0$.

- **Shannon Entropy of PSD** : H $= \sum_{f=0}^{f_s/2} P_{xx}(f) log_2[P_{xx}(f)]$

Also PSD spectrum mean amplitude, PSD spectrum stardard deviation, PSD spectrum skewness, PSD spectrum kurtosis, PSD spectrum amplitude maximum, PSD spectrum amplitude peak-to-peak were calculated in order to investigate their discriminative power.

Data on heart rate and speed detected via GPS have been excluded from the frequency analysis since they are not signals detected but an estimate averaged over a period of time, it would make no sense to analyze them in the frequency domain.

Furthermore the signals relative to the orientation of the device were excluded. (Pitch, Yaw, Roll).

At the end, 214 features were extracted from data relative to dataset A, 196 relative to dataset B.

3.3.3 Features in correlation domain

For the signals related to the axes x, y, z (acceleration, angular speed and device attitude) the Pearson correlation between the axes relative to the same signal was calculated.

Pearson correlation is defined as:

$$\rho_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y}$$

and it's an index of linear relation between two instances x and y, expressed in range $\rho_{xy} = [-1,1]$. For negative values, a negative linear correlation is assumed to exist between x and y; on the contrary, for positive values a positive one.

3.4 Feature normalization

Data normalization, also known as feature scaling or data standardization, is a techinque used to readjust the values to a more readable scale without losing the information contained in them in order to make objective functions work properly in a faster way.

In this work, *Min-Max scaling* was used for dataset A and B, with trasformation defined as :

$$X_{scaled} = \frac{X - min(X)}{max(X) - min(X)}$$

where X is a feature in the feature set. X_{scaled} values are in range [0,1].

3.5 Feature selection

In order to enhance generalization of the model, reduce dimensionality of the feature set and speed up training times, a feature selection is required.

There are different feature selection techniques :

- **filter**: usually are methods that applies statistical functions to features in order to rank them and discard the ones considered less informative; filter methods are considered effectives in computation time and robust to overfitting;

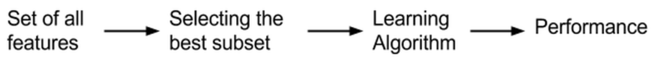


Figure 3.3. filter method for feature selection (ref.Wikipedia)

- **wrapper**: consists of an evaluation of the subset of features based on an apriori choice of the model used for learning: disadvantages may be a trend to overfitting when the number of observations is not sufficient and the high computational time when a large number of features is considered;
- **embedded**: this strategy tries to combine the two methods illustrated above, combining the feature selection process specific for the model selected and performing simultaneously classification;

In this work, a two steps filter feature selection has been performed:

- **correlation-based feature selection**: for each pair of all feature calculated in dataset, the Pearson correlation was computed[32]. For Pearson correlation, an inference process must be activated to evaluate how much we

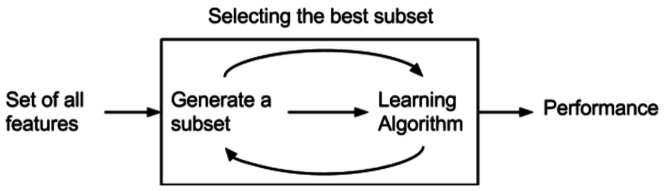


Figure 3.4. wrapper method for feature selection (ref. Wikipedia)

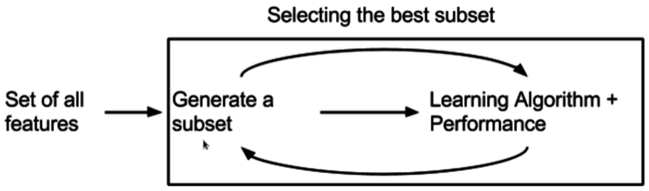


Figure 3.5. embedded methods for feature selection(ref. Wikipedia)

can trust about the correlation; in fact,samples on which the correlation is calculated could present, randomly, a linear relationship while the population from which it is extracted may not or samples on whose correlation has been calculated does not have a linear structure, while the population has. Defining H_0 as the null hypothesis, according to which there is no correlation between the variables, and the alternative hypothesis H_1 according to which correlation is present, the correlation between the two variables x and y is calculated on a sample of amplitude n. This procedure is repeated an infinite number of times creating a sampling distribution of the correlations that can be assumed as a normal distribution. The probability of the correlation under consideration is calculated, setting an α parameter (usually set to $\alpha = 0.05$, which is the region of non-acceptance of the normal distribution of correlations) to which corresponds a critical value above which the alternative hypothesis can be assumed to be true.

The Pearson correlation coefficient, considered in his absolute value, may indicate a correlation of three types:

– if $0 \leq \rho_{xy} < 0{,}3$ a *weak* linear correlation is present between x and y;
– if $0{,}3 \leq \rho_{xy} < 0{,}7$ a *moderate* linear correlation is present between x and y;
– if $0{,}7 \leq \rho_{xy} \leq 1$ a *strong* linear correlation is present between x and y;

Based on this range, a threshold of correlation value is assumed to 0,7. If ρ_{xy}

> 0.7 and $\alpha < 0.05$, it can be assumed that the features are highly linearly correlated, thus containing the same type of information. Based on this, one of them is discarded.

After this step, from an original set of 214 feature relative to dataset A, a subset of 41 features is selected. For dataset B,from 195 feature a subset of 36 is selected. In table 3.6, there are the feature selected for each dataset after this step of feature selection.

- A feature selection **setting a threshold on the variance** of the individual features has been applied: this process has been operated to further reduce the dimensional space of the features. We have tried to avoid a feature extraction process like PCA in order to maintain the original meaning of the features given to the selected models.

  The idea is that features with lower variance can express less informative patterns, so the that they can be considered less informative and are assumed to have less predictive power: so the variance of each feature has been computed and, looking at the distribution of variance values, **the threshold was set to the median value of all variances** so as to halve the number of features. In table 3.7, there are the feature selected for each dataset after this step of feature selection.

  Obviously, since it starts from the same data set (the data set B can be considered a subset of the data set A), the results are completely comparable: the set of features selected in the B dataset is in turn a subset of the feature set obtained from the A dataset, which is desirable having applied deterministic methods in the feature selection process.

3.6 Classification models

In this section, a description of the model used to perform HAR is provided. Considered the size of the dataset, four learning models have been tuned,trained and tested for dataset A and B : Decision Tree, Support Vector Machine, K Nearest Neighbor as single model classifiers and Random Forest as ensamble classifier.

All the classifiers were trained following these steps:

1. The dataset is split into **training** and **test** set: in the split phase the proportion of the samples in the test and in the training set was kept among the training sessions, taking into account the unbalancing in the number of samples related to different classes.

2. To tune classifiers, different combination of hyperparameters have been employed: in the tuning process, a k fold cross validation with k=5 is performed on the training set. The tuning process is defined as :

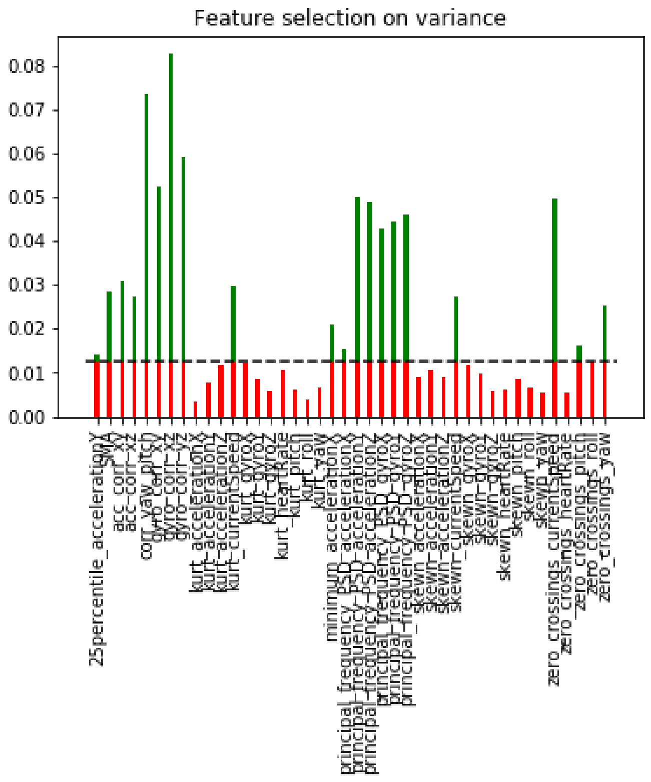


Figure 3.6. Feature selection based on variance threshold for dataset A. Threshold t is set to the median value of all variance.

(a) choose the classifiers and set the hyperparameters to be tested;

(b) find best parameters relying on the best accuracy score retrieved through a 5 fold cross validation greed search;

(c) evaluate the best classifier performance with the tested parameters with an ulterior 5 fold cross validation to check the effective goodness of hyperparameters;

Once found the best pameters relying on the best accuracy, an ulterior evaluation of the goodness of tuning hyperparameters was carried out through the indicators of:

- **Mean Accuracy** : it's the mean of accuracy for every class, where

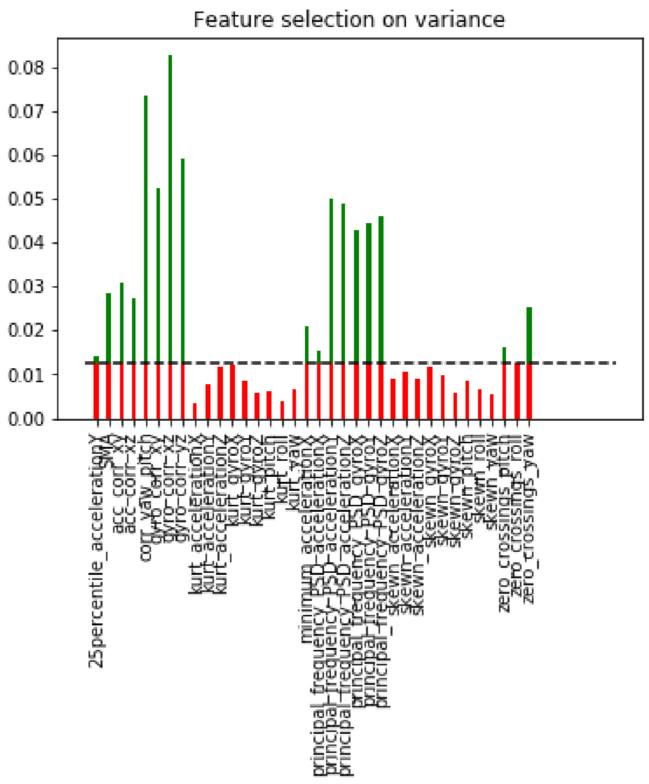


Figure 3.7. Feature selection based on variance threshold for dataset B. Variance threshold was set to median value of all feature set variances.

accuracy per single class is defined as as:

$$Acc_{class} = \frac{TP + TN}{FP + TP + FN + TN}$$

Mean Accuracy is defined consequently as:

$$MeanAccuracy = \frac{\sum_{i=0}^{n} Acc_{class_i}}{n}$$

where n is the number of classes considered.

- **Mean Precision** : it's the mean of precision for every class, where precision per single class is defined as as:

$$Prec_{class} = \frac{TP}{TP + FP}$$

Mean Precision is defined consequently as:

$$MeanPrecision = \frac{\sum_{i=0}^{n} Prec_{class_i}}{n}$$

- **Mean Recall**: it's the mean of recall for every class, where recall per single class is defined as as:

$$Rec_{class} = \frac{TP}{TP + FN}$$

Mean Recall is defined consequently as:

$$MeanRecall = \frac{\sum_{i=0}^{n} Rec_{class_i}}{n}$$

- **Mean F1 Score**: is the armonic mean between Mean Precision and Mean Recall, defined as :

$$MeanF1Score = \frac{2 * MeanPrecision * MeanRecall}{MeanPrecision + MeanRecall}$$

- **Fitting time** : time needed to train the model during fitting.
- **Scoring time** : time needed to perform scores during the evaluation of the fold left out for crossvalidation.

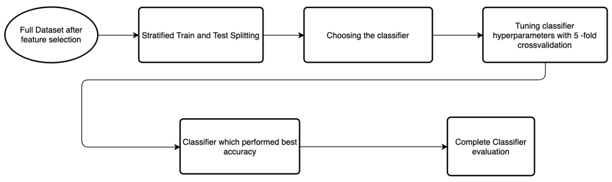

Full Dataset after feature selection
Stratified Train and Test Splitting
Choosing the classifier
Tuning classifier hyperparameters with 5 -fold crossvalidation
Classifier which performed best accuracy
Complete Classifier evaluation

Figure 3.8. Training and tuning process for selected models

3.6.1 K Nearest Neighbor

The k nearest neighbor is an instance-based algorithm defined as a non parametric method, that means not based on a priori hypotheses on the samples population, widely used in machine learning. Instances are vectors in a multidimensional feature space, each one associated with a label.[33] It's a method used both in classification and regression fields. The difference between this model in this two learning fields is in the outcome:

- for regression, it's a property value of the instance, which is the mean of the property values of the k nearest instances;
- for classification, it's a class membership, obtained as the most frequent class which appears in the majority of nearest instances;

Relying not on statistical parameters but on distance between samples, data spacial distribution plays a fundamentamental role in correct classification[33].

The fundamental hyperparameters to tune using this methods are:

- **k**: the number of neighbors taken in consideration during the classification phase; because it's a majority voting class it's suggested not to use k as an even number. During the tuning phase, k was evaluated in range

$$R_k = [1,3,5,7,9];$$

- the **algorithm** used to divide instances in the feature space. In this work, we explored :
  - **Ball tree**: a partitioning data structure which splits data points into a nested set of hyperspheres known as "balls" which containin a subset of the points to be searched. Each node splits the data points into two disjoint sets, each one associated with different balls. If a point belongs to two balls, it's assigned to one of the two intesecting balls according to its distance from the ball's center. Each leaf node in the tree defines a ball and enumerates all data points inside that ball[34].
  - **K-D tree**: in k-d tree, every leaf node is a k-dimensional point, every non-leaf node can generate an hyperplane which splits in two the sample space. Every node in the tree is associated with one of the k dimensions, with the hyperplane perpendicular to that dimension's axis. Points in the subset will be assigned to one of the half space according to the dimension considered for splitting.[35].
  - **Brute-force search**: consists of systematically enumerating all possible features for the splitting and checking whether each feature splits best the samples in leaf node.

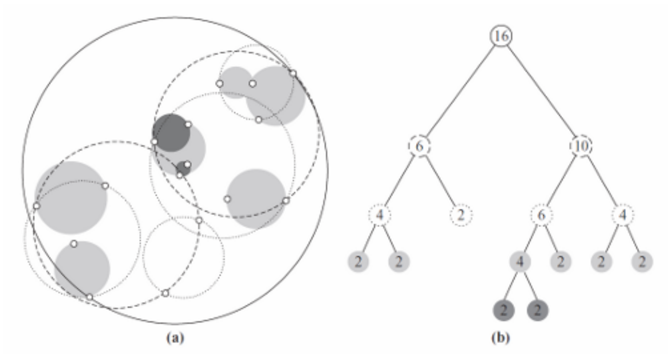


Figure 3.9. Example of ball tree algorithm in decision tree building

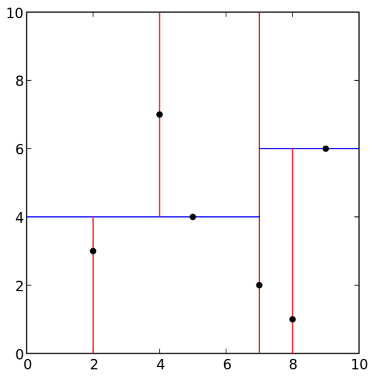


Figure 3.10. Example of k-d sample space decomposition

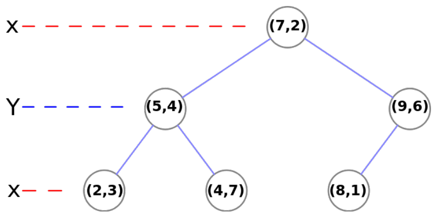


Figure 3.11. Example of tree generated from k-d space decomposition

- in case of using one of the tree feature space dividing algorithm, also **leaf size** of the tree was taken in consideration. Leaf size was evaluated in range

$$R_{lf} = [10,15,20];$$

- the **metric** used to evaluate distance between samples.In the tuning stage, three different distances have been evaluated :
  - **Euclidean distance**: for an n-dimensional space, given two points $A = (a_1, a_2, ..., a_n)$ and $B = (b_1, b_2, ..., b_n)$, the Euclidean distance is calculated as

    $$d_{Euc} = \sum_{i=1}^{n} \sqrt{(a_i^2 - b_i^2)}$$

  - **Manhattan distance**:for an n-dimensional space, given two points $A = (a_1, a_2, ..., a_n)$ and $B = (b_1, b_2, ..., b_n)$, the Manhattan distance is defined as

    $$d_{Man} = \sum_{i=1}^{n} |a_i - b_i|$$

  - **Chebyshev distance**: for an n-dimensional space, given two points $A = (a_1, a_2, ..., a_n)$ and $B = (b_1, b_2, ..., b_n)$, the Manhattan distance is defined as

    $$d_{Cheb} = \max_i \{|a_i - b_i|\}$$

In figures 3.12-19 are showed the mean results of training with a 5-fold stratified crossvalidation.

3.6.2 Support Vector Machine

Support Vector Machine is an algorithm used in supervised learning used in classification and regression analysis; this model provides a representation of samples in space in a way that instances belonging to different classes are separated by a gap. Wider is the gap, better will be the predictive capability of the model : for this purpose, a *margin* should be defined as the gap between the closest sample to the separating plane and the plane itself.
Linear SVM is able to classify data maximizing this margin. Support vectors, the subset of samples closest to the decision boundaries, are the ones on which the function depends. A misclassification margin zone defined as *soft margin* could be defined, assuming that on data could be affected from noise, which doesn't allow a correct classification performed by the model. Thus, performing this operation, a linear hyperplane could not be sufficient to divide well high-dimensional data. To overcome this problem, SVM admits using a kernel to carry out a transformation

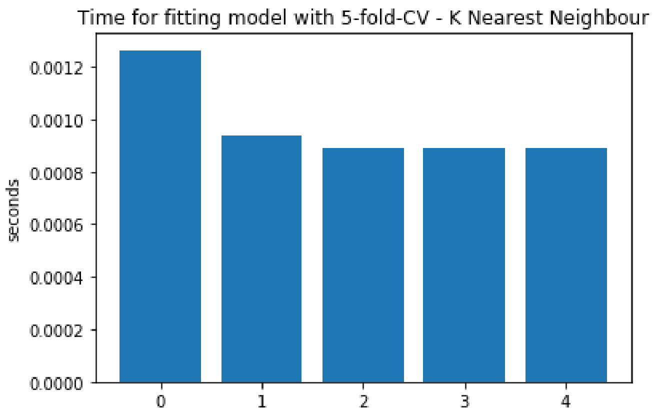


Figure 3.12. Fit time related to Dataset A for every cross validation fold

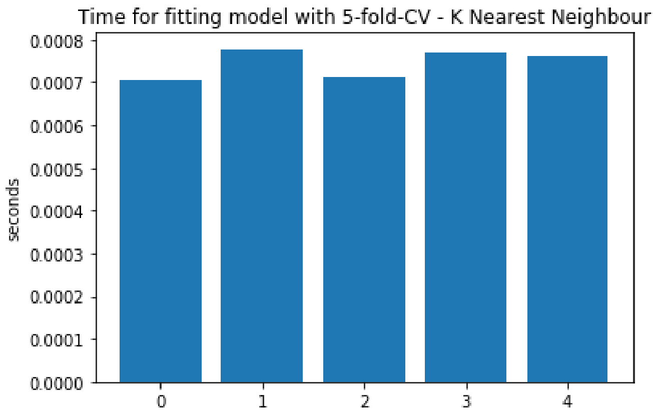


Figure 3.13. Fit time related to Dataset B for every cross validation fold

that increases the dimensionality of the space of the solutions, thus allowing an easier data separation. Obviously, this operation increases computational time. As for KNN, it's mainly used for classification and regression analysis[36]. The hyperparameters evaluated for tuning SVM performance were:

- **kernel**: several kernels were tested trying to explore the best spacial transformation to separate data.
  - **linear** kernel;

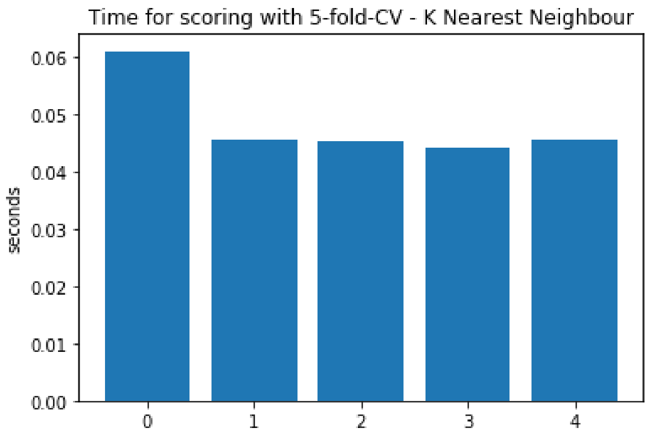


Figure 3.14. Scoring time related to Dataset B for every cross validation fold

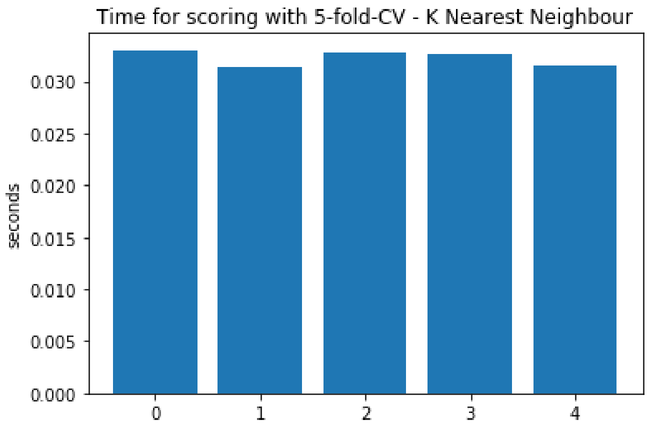


Figure 3.15. Scoring time related to Dataset B for every cross validation fold

- **polynomial** kernel;
- **Radial Basis Function** kernel. It's defined, between two feature vectors **a** and **b** as:

$$K(a,b) = \exp\left(-\frac{||a-b||^2}{2\sigma^2}\right),$$

  where $||a-b||^2$ is defined as the squared Euclidean distance;

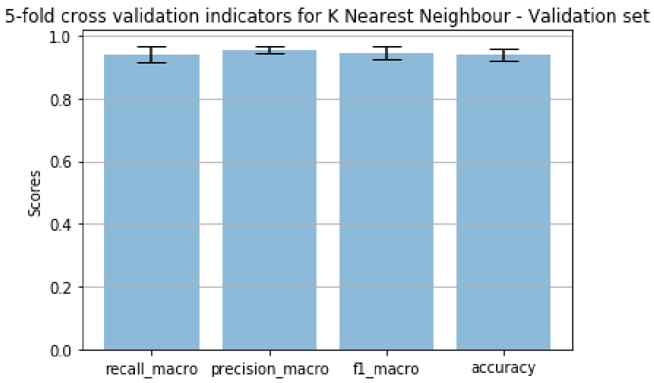


Figure 3.16. Results for accuracy, precision, recall and F1 score averaged among the 5 crossvalidation folds related with tuning SVC on Dataset B. Those results are obtained with the best parameters presented in table 3.8

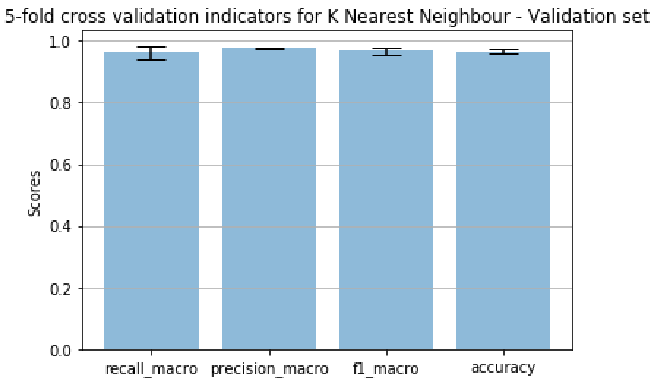


Figure 3.17. Results for accuracy, precision, recall and F1 score averaged among the 5 crossvalidation folds related with tuning KNN on Dataset A. Those results are obtained with the best parameters presented in table 3.8

- **C**: is the parameter relative to the *slack variable*, which allows a misclassification margin. In tuning, C was in range [0.1, 10] with step = 1.
- γ: is defined, just for Radial Basis Function, as:

$$\gamma = \frac{1}{2\sigma^2},$$

  admitting RBF definition as:

$$K(a,b) = \exp(-\gamma ||a-b||^2).$$

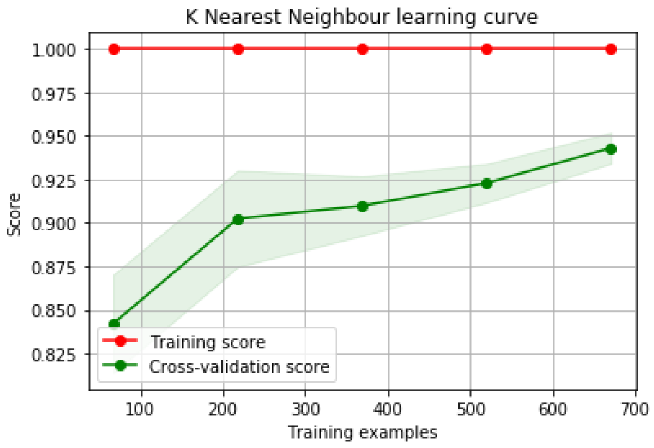


Figure 3.18. Learning curve for training and validation set using KNN on Dataset A with number of samples on the x axis and accuracy score on the y axis

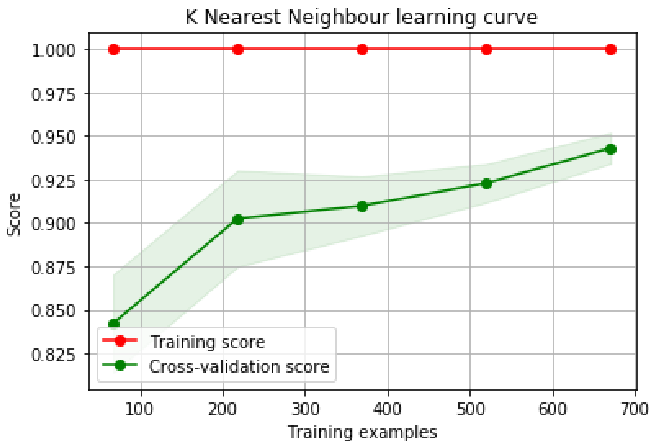


Figure 3.19. Learning curve for training and validation set using KNN on Dataset B with number of samples on the x axis and accuracy score on the y axis

It controls the tradeoff between error due to bias and variance in the model: for large values, the model can overfit and be prone to high variance. In tuning, γ was in range [1, 10] with step = 1.

- **decision function**: SVM for multiclass models can work in two modalities.

- **One-vs-rest**: builds a binary classifier for every class; each classifier finds the best hyperplane to separate each class from the rest of samples;
- **One-vs-one**: buids a binary classfiers for every possible combination of two classes; the classification output is obtained by majority voting from all the classifiers built.

  every time two classes were considered, a binary classifier between them is built: the class returned from majority of the classifiers is the classification output;

In figures 3.20-3.27 the mean results are shown of training with a 5-fold stratified crossvalidation.

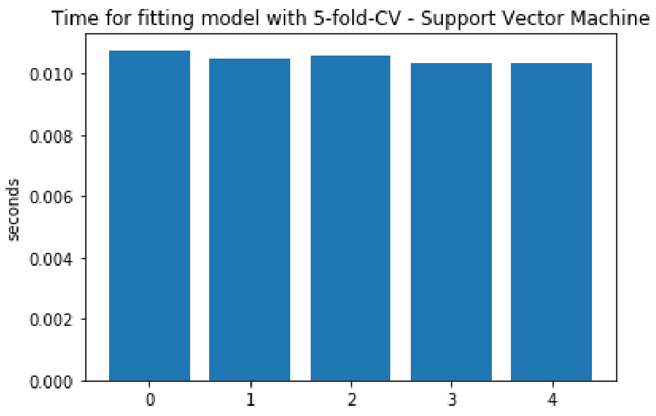


Figure 3.20. Fit time related to Dataset A for every cross validation fold

3.6.3 Decision Tree

In machine learning a decision tree is a predictive model based on a directed acyclic graph, in which:

- each internal node represents a variable;
- an arc towards a child node represents a possible value for that variable;
- a leaf represents the predicted value for the target variable based on the values of the other properties;

The tree is represented by the path from the root node to the leaf node. This method providesa very intuitive way to get a graphical representation of the classification process.

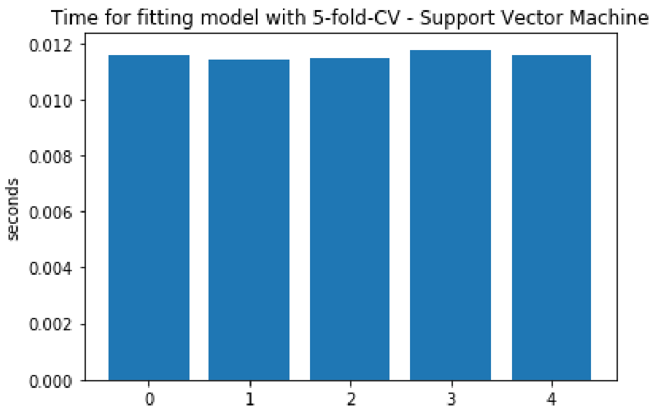


Figure 3.21. Fit time related to Dataset B for every cross validation fold

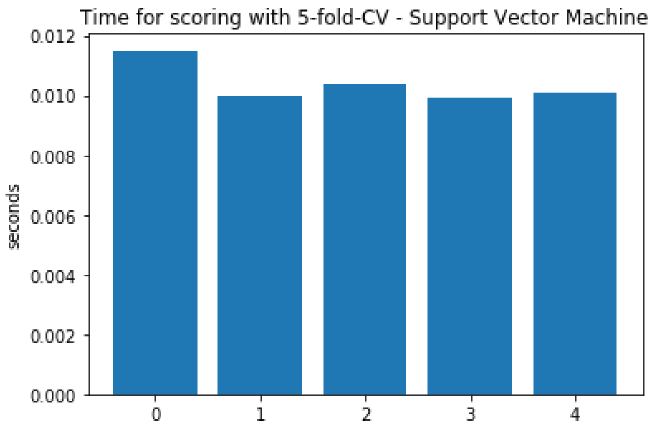


Figure 3.22. Scoring time related to Dataset A for every cross validation fold

The procedure is simple: for each node, the algorithm looks for the feature which best splits the samples until the leaf contains instances of the same class[37].

The hyperparameters evaluated for tuning Decision Tree performance were:

- **Splitting criterion**: it's choosen between two different criteria, Gini Index and Entropy Information gain, which give a measure of homogeneity of the leaf node. The homogeneity is intended as how much a leaf node contains instances of a single class and it is in range (0,1] for all classes considered in classification: ower the score, higher the homogeneity in the leaf node. The

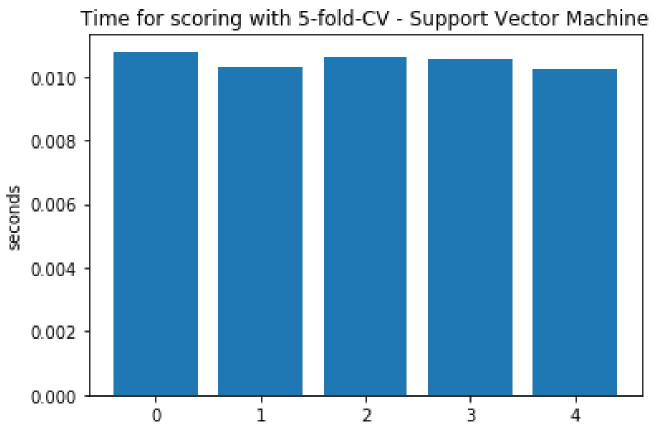


Figure 3.23. Scoring time related to Dataset B for every cross validation fold

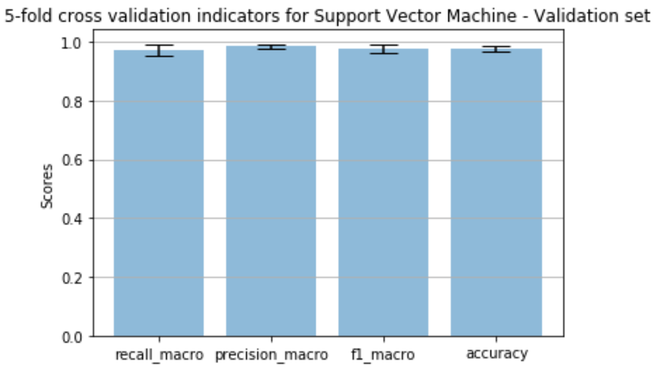


Figure 3.24. Results for accuracy, precision, recall and F1 score averaged among the 5 crossvalidation folds related with tuning SVC on Dataset A. Those results are obtained with the best parameters presented in table 3.8

two criteria are defined as:

– **Gini impurity**: measures the hetereogenity of a dataset by computing the product between the probability of a sample to belong to a class versus the probability to belong to another after splitting the leaf node on a feature; for a dataset with J classes, Gini impurity for every node

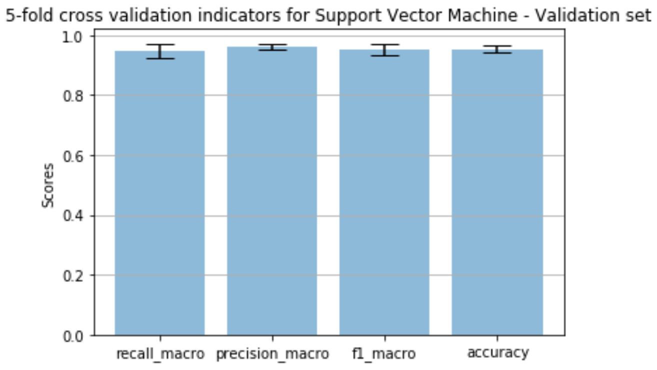


Figure 3.25. Results for accuracy, precision, recall and F1 score averaged among the 5 crossvalidation folds related with tuning SVC on Dataset B. Those results are obtained with the best parameters presented in table 3.8

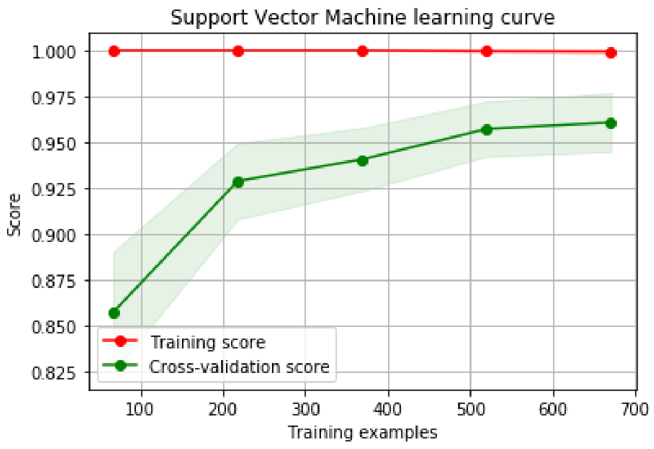


Figure 3.26. Learning curve for training and validation set using SVC on Dataset A with number of samples on the x axis and accuracy score on the y axis

is defined as:

$$I_G(p) = 1 - \sum_{i=1}^{J} p_i^2,$$

where p_i is the fraction of i labeled samples in the node.

– **Entropy Information gain**: based on information theory, defining

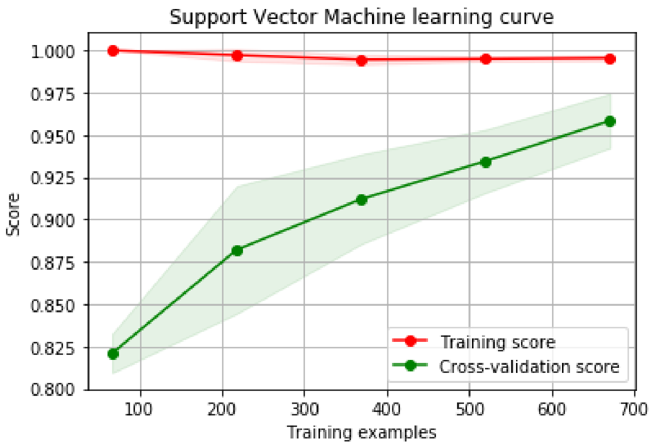


Figure 3.27. Learning curve for training and validation set using KNN on Dataset B with number of samples on the x axis and accuracy score on the y axis

entropy as:

$$H(T) = -\sum_{i=0}^{J} p_i \log_2 p_i.$$

Information gain is defined as the difference between entropy in the parent node T and the weighted sum of entropy in the children node relative to attribute a used for splitting.

$$I_G(T,a) = H(T) - H(T|a),$$

with

$$H(T|a) = -\sum_{a} p(a) \sum_{i=1}^{J} -Pr(i|a) \log_2 Pr(i|a)$$

- **number of features** taken in consideration when splitting. In tuning was taken into account as number of features to be considered:
  - all the features in dataset;
  - the square root of all features in dataset;
  - the log_2 of all features in dataset;

In figures 3.28-3.35 are showed the mean results of training with a 5-fold stratified crossvalidation.

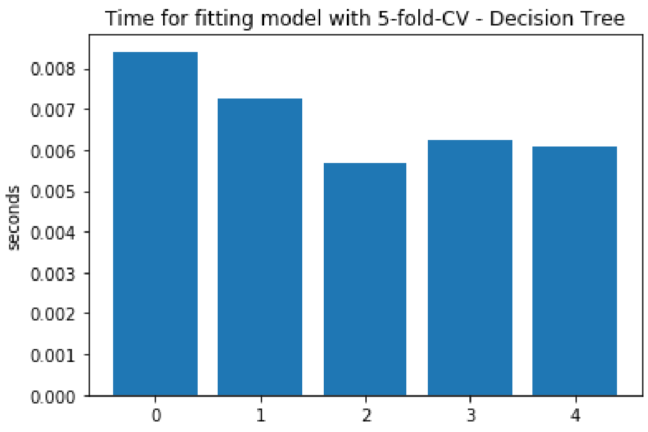


Figure 3.28. Fit time related to Dataset A for every cross validation fold

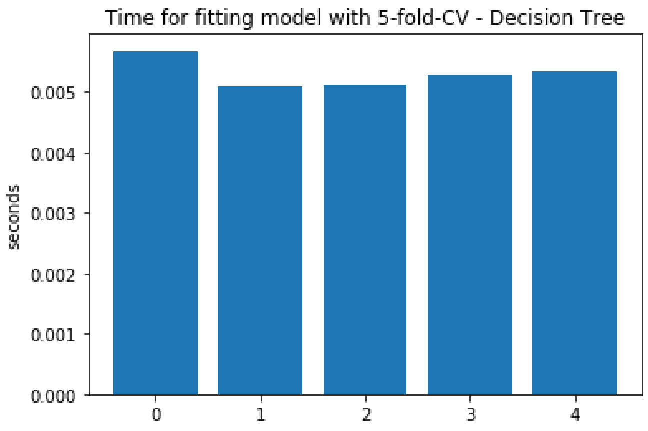


Figure 3.29. Fit time related to Dataset B for every cross validation fold

3.6.4 Random Forest

Random forest is an ensemble method which is made up of different decision trees, giving as output the mode of the class label[38].

The hyperparameters evaluated for tuning Decision Tree performance were:

- **Number of trees**: number of decision tree used for voting. In tuning the range considered was [2, 4, 8, 16, 32, 64, 100, 200]

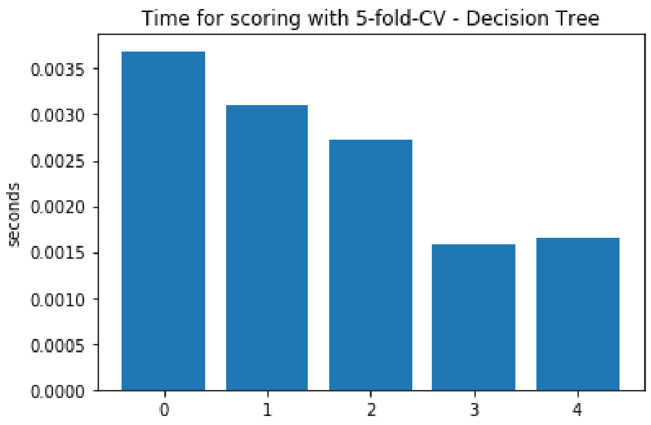


Figure 3.30. Scoring time related to Dataset A for every cross validation fold

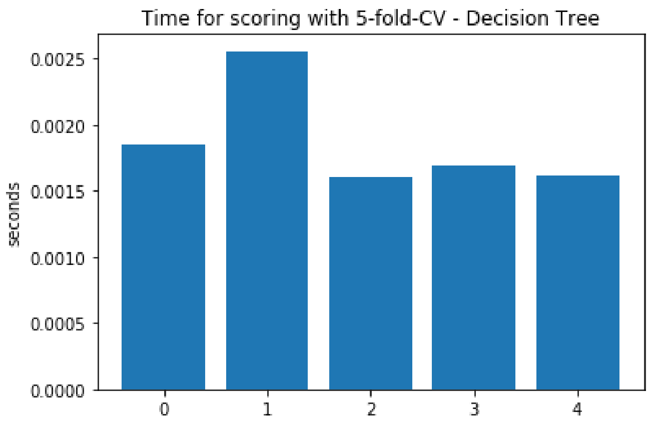


Figure 3.31. Scoring time related to Dataset B for every cross validation fold

- **Splitting criterion**: Gini index and Entropy Information gain as defined for decision tree;
- **number of features** taken in consideration when splitting as for decision tree.

The results of parameter tuning are showed in table 3.8.

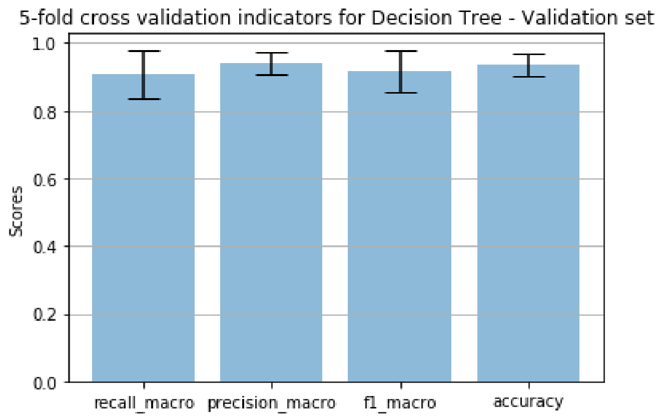


Figure 3.32. Results for accuracy, precision, recall and F1 score averaged among the 5 crossvalidation folds related with tuning Decision Tree on Dataset A. Those results are obtained with the best parameters presented in table 3.8

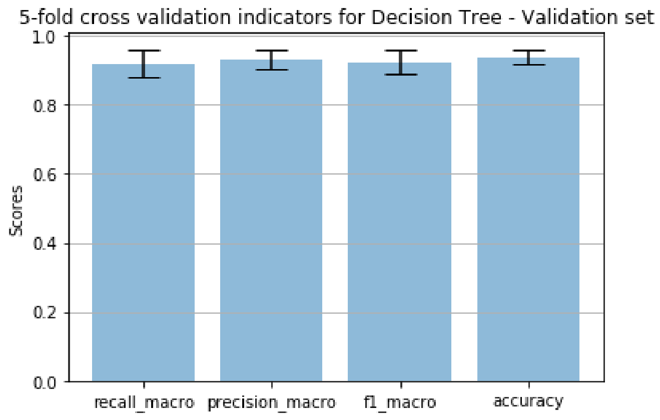


Figure 3.33. Results for accuracy, precision, recall and F1 score averaged among the 5 crossvalidation folds related with tuning Decision Tree on Dataset B. Those results are obtained with the best parameters presented in table 3.8

3.6.5 Computational time

Undoubtedly the KNN and the decision tree have exhibit the lower computational time among the considered classifiers: as far as SVC is concerned, time requirements seem to be at least one order of magnitude higher both in terms of training and testing, reaching even three orders of magnitude of difference compared with

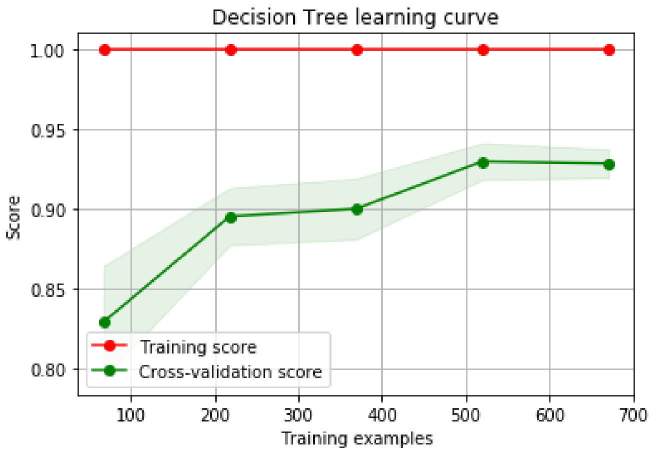


Figure 3.34. Learning curve for training and validation set using Decision Tree on Dataset A with number of samples on the x axis and accuracy score on the y axis

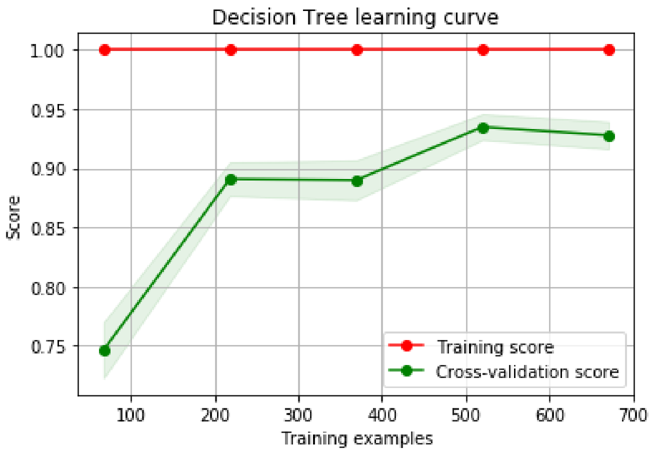


Figure 3.35. Learning curve for training and validation set using Decision Tree on Dataset B with number of samples on the x axis and accuracy score on the y axis

Random Forest, for which time requirements are of tenths of a second. It should also be noted that in the B dataset, without heart rate and GPS, the times for both fitting and scoring are slightly lower: this result is consistent since the lower

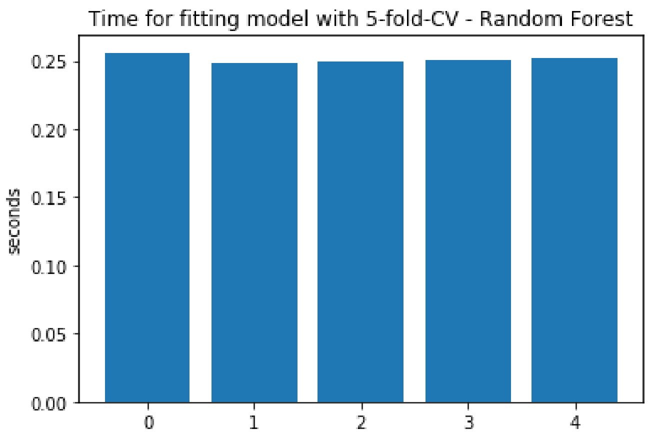


Figure 3.36. Dataset A

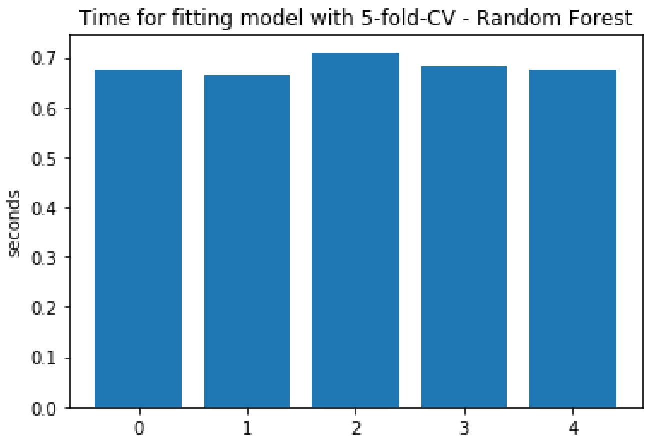


Figure 3.37. Fit time related to Dataset B for every cross validation fold

is the number of features on which the model must recognize the class, the lower are computational times.

3.6.6 Validation set score

The models obtained from the tuning of the parameters are satisfying and completely comparable. The standard deviations of the evaluation parameters are

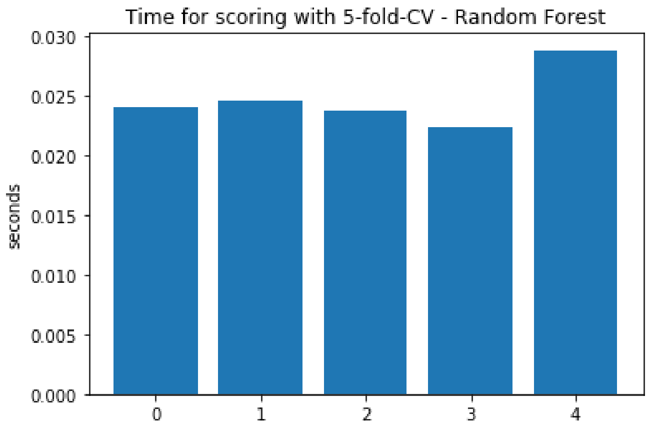


Figure 3.38. Fit time related to Dataset A for every cross validation fold

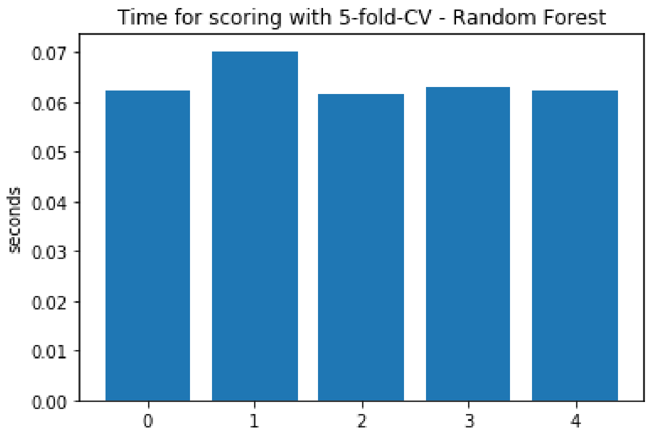


Figure 3.39. Scoring time related to Dataset B for every cross validation fold

considerable in the case of the decision tree, which suggests a strong variability of the goodness of prediction . Not by chance, using the Random Forest which mediates between the results of 200 decision trees, the standard deviation decreases significantly: affirm that, when robust models are considered, we can expect high percentages of success regardless of the fold on which the validation is carried out, that we can expect high percentages of success in recognition activity considering not only in terms of accuracy, but also in terms of precision and recall.

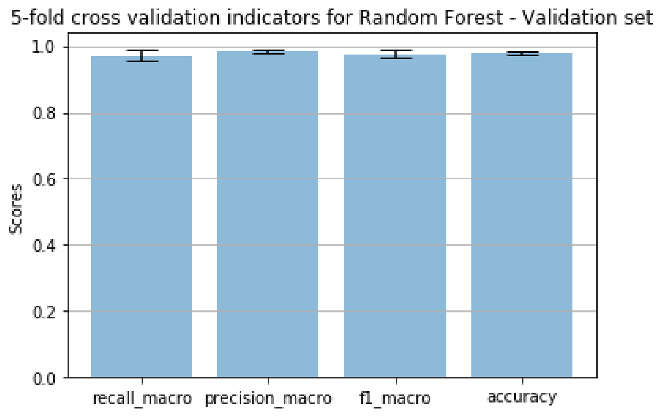


Figure 3.40. Results for accuracy, precision, recall and F1 score averaged among the 5 crossvalidation folds related with tuning Random Forest on Dataset A. Those results are obtained with the best parameters presented in table 3.8

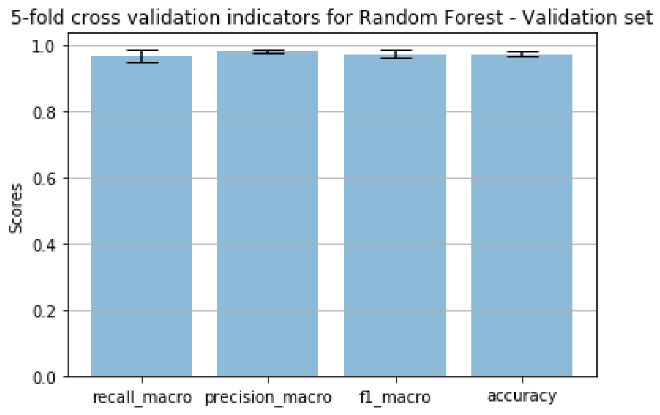


Figure 3.41. Results for accuracy, precision, recall and F1 score averaged among the 5 crossvalidation folds related with tuning Random Forest on Dataset B. Those results are obtained with the best parameters presented in table 3.8

3.6.7 Learning curve

From what can be seen by observing the trends of accuracy as a function of tnumber of samples provided in the training stage, undoubtedly with a higher number of samples very high success rates is reached. From the curves it seems that there is no convergence between the accuracy scores of the training and those of the

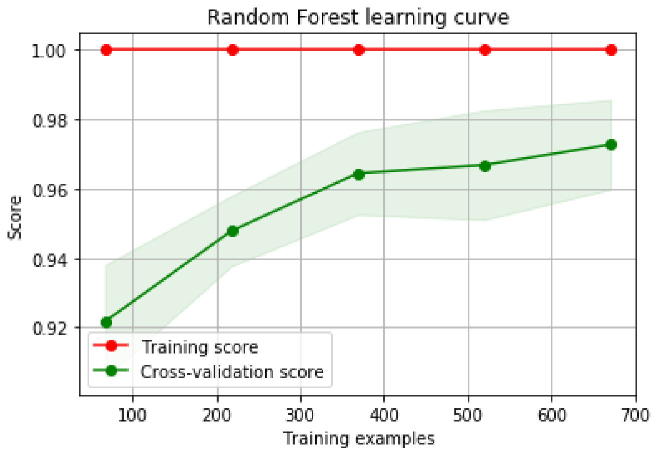


Figure 3.42. Learning curve for training and validation set using Random Forest on Dataset A with number of samples on the x axis and accuracy score on the y axis

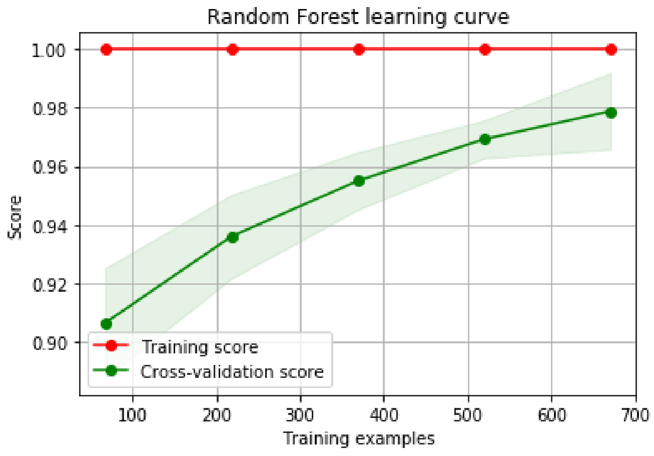


Figure 3.43.

crossvalidation, which would mean that the model does not adapt perfectly to the dataset, allowing a good generalization. However, the success rate in recognition seems to be comfortable. It can also be observed that the standard deviation of the accuracy obtained on the crossvalidation folds tends to decrease as the number of samples increases, except for the random forest, for which the thinning of the variability of the accuracy does not take place. The decision tree seems to have more stable results since the standard deviation on the obtained accuracy is the smallest among the analyzed models.

	Dataset A	Dataset B
Feature set	Q1 acceleration Y, kurtosis accelerationX, kurtosis accelerationY, kurtosis accelerationZ, kurtosis currentSpeed, kurtosis gyroX, kurtosis gyroY, kurtosis gyroZ, kurtosis heartRate, kurtosis pitch, kurtosis roll, kurtosis yaw, min accelerationX, PF accelerationX, PF accelerationY, PF accelerationZ, PF gyroX, PF gyroY, PF gyroZ, skewness accelerationX, skewness accelerationY, skewness accelerationZ, skewness currentSpeed, skewness gyroX, skewness gyroY, skewness gyroZ, skewness heartRate, skewness pitch, skewness roll, skewness yaw, mean crossings currentSpeed, mean crossings heartRate, mean crossings pitch, mean crossings roll, mean crossings yaw, acc. correlation xy, acc.correlation xz, correlation yaw pitch, gyro correlation xy, gyro correlation xz, gyro correlation yz, SMA	Q1 accelerationY, kurtosis accelerationX, kurtosis accelerationY, kurt_accelerationZ, kurtosis gyroX, kurtosis gyroY, kurtosis gyroZ, kurtosis pitch, kurtosis roll, kurtosis yaw, min accelerationX, PF accelerationX, PF accelerationY, PF accelerationZ, PF gyroX, PF gyroY, PF gyroZ, skewness accelerationX, skewness accelerationY, skewness accelerationZ, skewness gyroX, skewness gyroY, skewness gyroZ, skewness pitch, skewness roll, skewness yaw, mean crossings pitch, mean crossings roll, mean crossings yaw, acc.correlation xy, acc. correlation xz, correlation yaw pitch, gyro correlation xy, gyro correlation xz, gyro correlation yz, SMA
Number of features	42	36

Table 3.6. Feature selected after Pearson correlation index (threshold fixed to 0,7) feature selection

	Dataset A	Dataset B
Feature set	Q1 accelerationY, kurtosis currentSpeed, min accelerationX, PF accelerationX, PF accelerationY, PF accelerationZ , PF gyroX, PF gyroY, PF gyroZ, skewness currentSpeed, mean crossings currentSpeed, mean crossings pitch, mean crossings roll, mean crossings yaw, correlation acc xy , acc correlation xz, correlation yaw pitch, gyro correlation xy, gyro correlation xz, gyro correlation yz, SMA	Q1 accelerationY, min accelerationX , PF accelerationX, PF accelerationY, PF accelerationZ, PF gyroX, PF gyroY, PF gyroZ, mean crossings pitch, mean crossings roll, mean crossings yaw, acc correlation xy, acc correlation xz, correlation yaw pitch, gyro correlation xy, gyro correlation xz, gyro correlation yz, SMA
Number of features	21	18

Table 3.7. Number of features and feature selected for each dataset after variance threshold feature selection

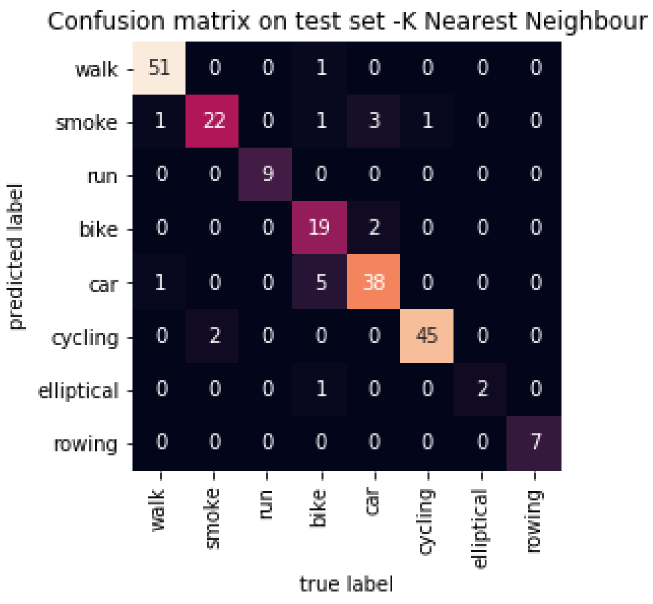
	Best parameters Dataset A	Best Parameters Dataset B	Accuracy Dataset A	Accuracy Dataset B
KNN	algorithm: ball tree leaf size = 10 metric : Manhattan neighbors: 9	algorithm: ball tree leaf size = 10 metric : Manhattan neighbors: 1	0.955	0.943
SVC	C = 8.1 decision function shape: one-vs- rest $\gamma = 1$ kernel: RBF	C = 8.1 decision function shape: one-vs- rest $\gamma = 1$ kernel: RBF	0.962	0.958
Decision Tree	Splitting criterium : Gini impurity Features considered: all feature set Splitter : best	Splitting criterium : Information gain Features considered: all feature set Splitter : best	0.926	0.931
Random Forest	Estimators : 200 Splitting criterium : Gini impurity Features considered: $\log_2 feat.set$ Splitter : best	Estimators : 200 Splitting criterium : Gini impurity Features considered: $\log_2 feat.set$ Splitter : best	0.975	0.979

Table 3.8. Sum table of tuning parameters for dataset A and B

Chapter 4

Results

In this chapter, the results on the test set of the four classifiers tuned as explained in the previous chapter, are presented and discussed.



Confusion matrix on test set -K Nearest Neighbour

predicted label

	walk	smoke	run	bike	car	cycling	elliptical	rowing
walk	51	0	0	1	0	0	0	0
smoke	1	22	0	1	3	1	0	0
run	0	0	9	0	0	0	0	0
bike	0	0	0	19	2	0	0	0
car	1	0	0	5	38	0	0	0
cycling	0	2	0	0	0	45	0	0
elliptical	0	0	0	1	0	0	2	0
rowing	0	0	0	0	0	0	0	7

true label

Figure 4.1. Confusion Matrix for K Nearest Neighbor evaluated on test set - Dataset A

First of all, it must be specified that the poor support of the classes labeled as run, elliptical and rowing is the main cause of the performance: in every case they

	Accuracy	Precision	Recall	F1- Score	Support
walking	0.980	0.962	0.980	0.974	52
smoking	0.786	0.917	0.786	0.846	28
running	1.000	1.000	1.000	1.000	9
cycling out.	0.905	0.704	0.905	0.792	21
car driving	0.864	0.884	0.864	0.874	44
cycling ind.	0.958	0.978	0.957	0.968	47
elliptical	0.667	1.000	1.000	1.000	3
rowing	1.000	1.000	1.000	1.000	7
AVG	**0.895**	**0.921**	**0.915**	**0.915**	

Table 4.1. Classification Report for K Nearest Neigbors evaluated on test set - Dataset A

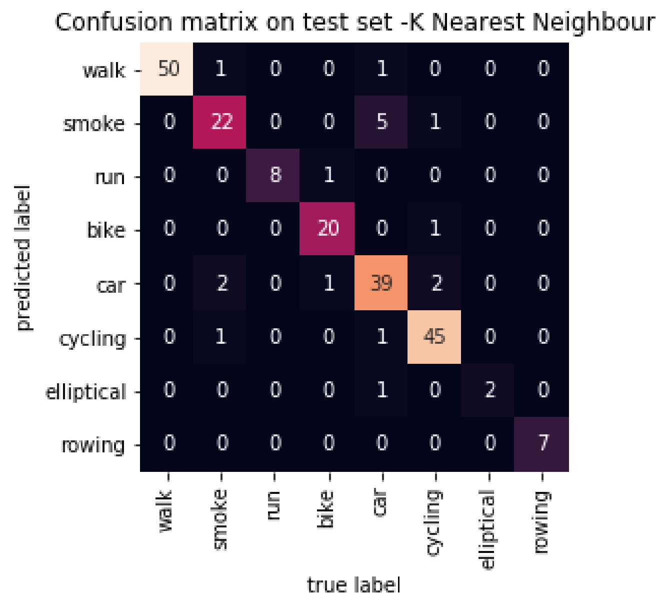
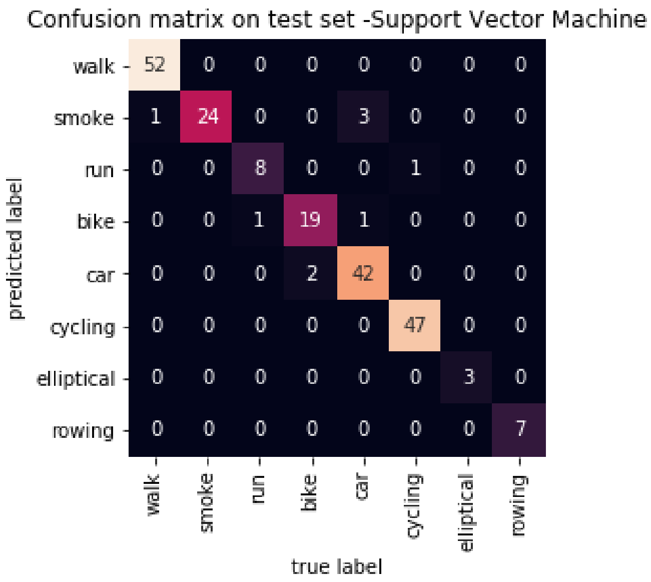


Figure 4.2. Confusion Matrix for K Nearest Neighbor evaluated on test set - Dataset B

were included in the model to test the model recognition capacity. Only one sample belonging to of those classes is misclassified in both test sets (A,B) by the KNN, Decision Tree and Random Forest respectively . Regardless of the model, it is clear that the activities have been optimally recognized: this underlines the quality of the features obtained from the selection. Overall, better results are observed on

	Accuracy	Precision	Recall	F1- Score	Support
walking	0.961	1.000	0.961	0.980	52
smoking	0.786	0.846	0.786	0.815	28
running	0.889	1.000	0.889	0.941	9
cycling out.	0.952	0.909	0.952	0.930	21
car driving	0.886	0.829	0.886	0.857	44
cycling ind.	0.957	0.918	0.957	0.973	47
elliptical	0.667	1.000	0.667	0.800	3
rowing	1.000	1.000	1.000	1.000	7
AVG	**0.887**	**0.916**	**0.915**	**0.915**	

Table 4.2. Classification Report for K Nearest Neighbor evaluated on test set - Dataset B



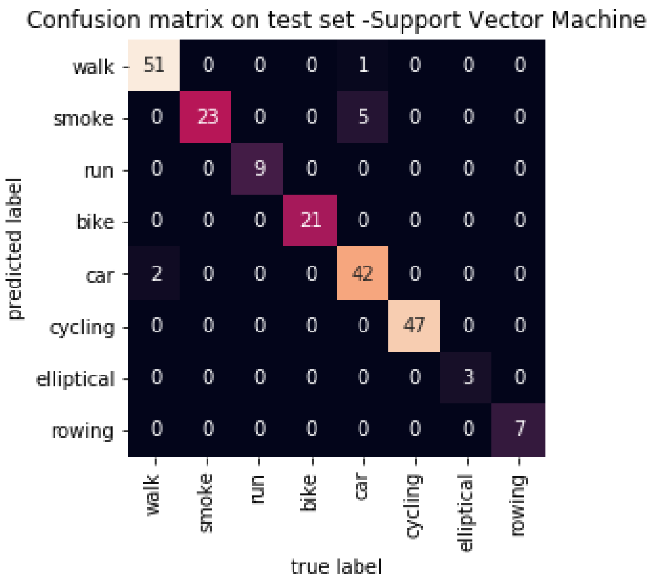
predicted \ true	walk	smoke	run	bike	car	cycling	elliptical	rowing
walk	52	0	0	0	0	0	0	0
smoke	1	24	0	0	3	0	0	0
run	0	0	8	0	0	1	0	0
bike	0	0	1	19	1	0	0	0
car	0	0	0	2	42	0	0	0
cycling	0	0	0	0	0	47	0	0
elliptical	0	0	0	0	0	0	3	0
rowing	0	0	0	0	0	0	0	7

Figure 4.3. Confusion Matrix for Support Vector Machine evaluated on test set - Dataset A

the test set which does not include heart rate and speed via GPS: this indicates the features related to these acquisitions globally do not improve the performance of the model. However, it can be noticed that, with the exception of the decision tree, in the absence of these data we can see misclassifications between the smoke class and the car class, not present for the corresponding model trained on the entire

	Accuracy	Precision	Recall	F1- Score	Support
walking	1.000	0.981	1.000	0.990	52
smoking	0.857	0.981	0.857	0.923	28
running	0.889	0.889	0.889	0.888	9
cycling out.	0.905	0.905	0.905	0.905	21
car driving	0.954	0.931	0.954	0.933	41
cycling ind.	1.000	0.979	1.000	0.989	47
elliptical	0.889	1.000	1.000	1.000	3
rowing	1.000	1.000	1.000	1.000	7
AVG	**0.951**	**0.958**	**0.957**	**0.957**	

Table 4.3. Classification Report for Support Vector Machine evaluated on test set - Dataset A



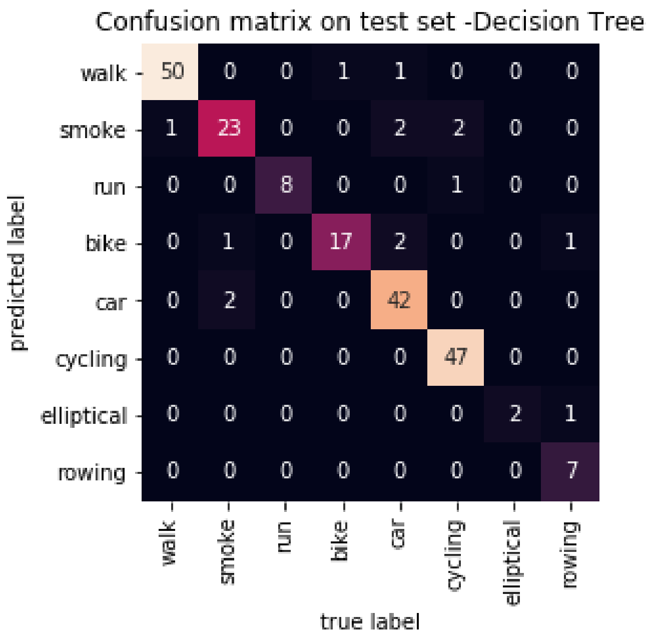
predicted \ true	walk	smoke	run	bike	car	cycling	elliptical	rowing
walk	51	0	0	0	1	0	0	0
smoke	0	23	0	0	5	0	0	0
run	0	0	9	0	0	0	0	0
bike	0	0	0	21	0	0	0	0
car	2	0	0	0	42	0	0	0
cycling	0	0	0	0	0	47	0	0
elliptical	0	0	0	0	0	0	3	0
rowing	0	0	0	0	0	0	0	7

Figure 4.4. Confusion Matrix for Support Vector Machine evaluated on test set - Dataset B

feature set: this can demonstrate how this feature have potential to be further investigated in order to be discriminant for the recognition of specific classes. The best performing model proves to be the Random Forest, with better results on the B dataset, although all models have well over 90% performance for all parameters, as shown in the tables in the previous chapter[tab.3.8].

	Accuracy	Precision	Recall	F1- Score	Support
walking	0.981	0.962	0.981	0.971	52
smoking	0.821	1.000	0.821	0.902	28
running	1.000	1.000	1.000	1.000	9
cycling out.	1.000	1.000	1.000	1.000	21
car driving	0.954	0.875	0.954	0.913	44
cycling ind.	1.000	1.000	1.000	1.000	47
elliptical	1.000	1.000	1.000	1.000	3
rowing	1.000	1.000	1.000	1.000	7
AVG	**0.969**	**0.962**	**0.962**	**0.961**	

Table 4.4. Classification Report for Support Vector Machine evaluated on test set - Dataset B



predicted \ true	walk	smoke	run	bike	car	cycling	elliptical	rowing
walk	50	0	0	1	1	0	0	0
smoke	1	23	0	0	2	2	0	0
run	0	0	8	0	0	1	0	0
bike	0	1	0	17	2	0	0	1
car	0	2	0	0	42	0	0	0
cycling	0	0	0	0	0	47	0	0
elliptical	0	0	0	0	0	0	2	1
rowing	0	0	0	0	0	0	0	7

Figure 4.5. Confusion Matrix for Decision Tree evaluated on test set - Dataset A

	Accuracy	Precision	Recall	F1- Score	Support
walking	0.961	0.980	0.961	0.971	52
smoking	0.821	0.885	0.821	0.852	28
running	0.889	1.000	0.889	0.941	9
cycling out.	0.801	0.893	0.810	0.923	21
car driving	0.954	0.893	0.954	0.923	41
cycling ind.	1.000	0.940	1.000	0.969	47
elliptical	0.667	1.000	0.667	0.800	3
rowing	1.000	0.778	1.000	0.875	7
AVG	**0.888**	**0.931**	**0.928**	**0.928**	

Table 4.5. Classification Report for Decision Tree evaluated on test set - Dataset A

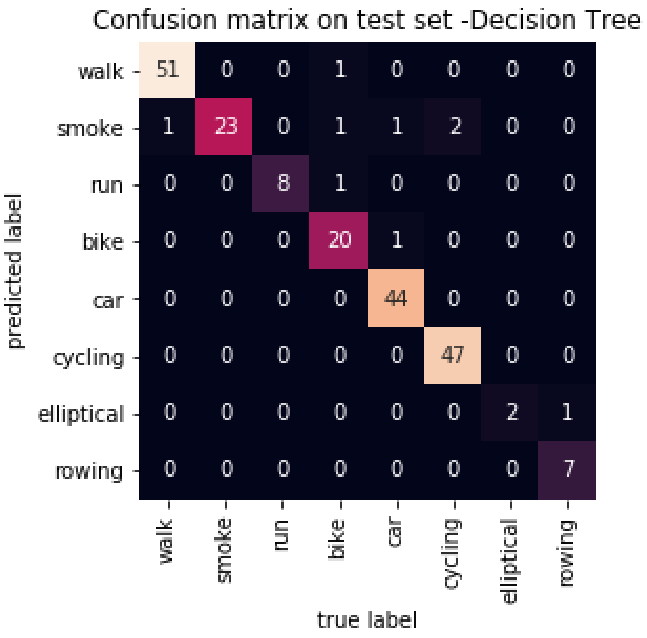
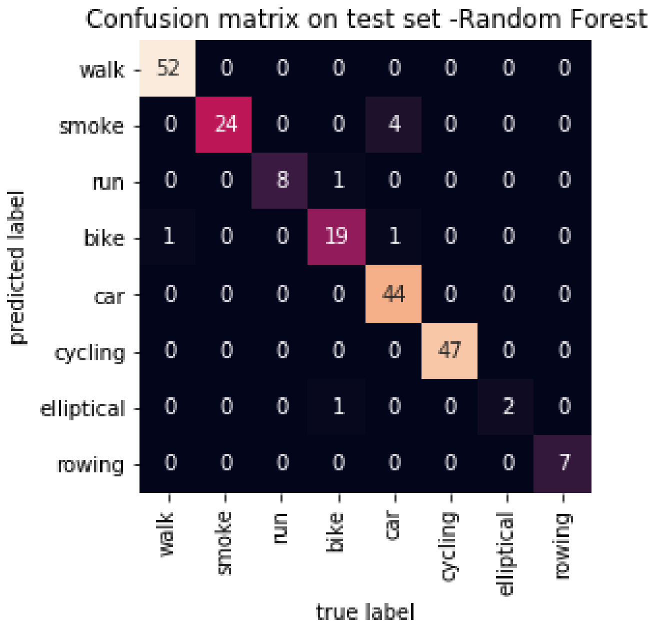


Figure 4.6. Confusion Matrix for Decision Tree evaluated on test set - Dataset B

	Accuracy	Precision	Recall	F1- Score	Support
walking	0.981	0.981	0.981	0.981	52
smoking	0.821	1.000	0.821	0.902	28
running	0.889	1.000	0.888	0.942	9
cycling out.	0.952	0.869	0.952	0.909	21
car driving	1.000	0.956	1.000	0.978	44
cycling ind.	1.000	0.959	1.000	0.979	47
elliptical	0.667	1.000	0.667	0.800	3
rowing	1.000	0.875	1.000	0.933	7
AVG	**0.913**	**0.960**	**0.957**	**0.956**	

Table 4.6. Classification Report for Decision Tree evaluated on test set - Dataset B

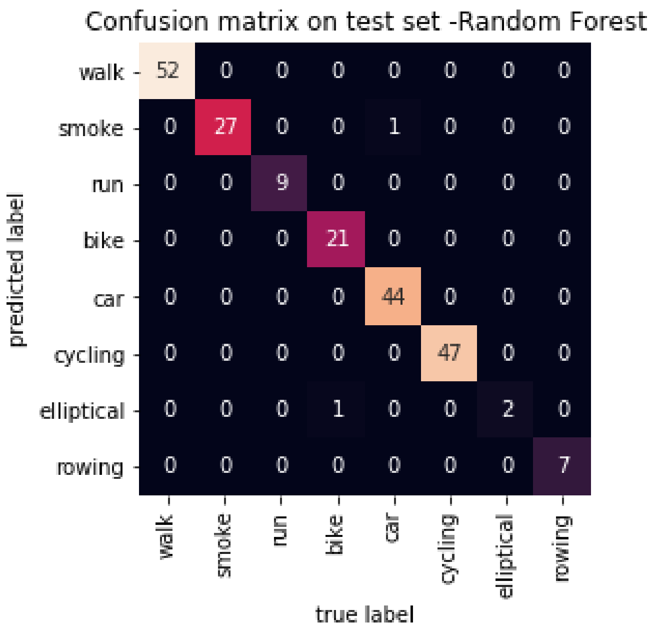


	walk	smoke	run	bike	car	cycling	elliptical	rowing
walk	52	0	0	0	0	0	0	0
smoke	0	24	0	0	4	0	0	0
run	0	0	8	1	0	0	0	0
bike	1	0	0	19	1	0	0	0
car	0	0	0	0	44	0	0	0
cycling	0	0	0	0	0	47	0	0
elliptical	0	0	0	1	0	0	2	0
rowing	0	0	0	0	0	0	0	7

Figure 4.7. Confusion Matrix for Random Forest evaluated on test set - Dataset A

	Accuracy	Precision	Recall	F1- Score	Support
walking	0.981	0.981	0.981	0.981	52
smoking	0.928	1.000	0.923	0.946	28
running	1.000	1.000	1.000	1.000	9
cycling out.	0.945	0.954	1.000	0.978	21
car driving	1.000	0.956	1.000	0.978	44
cycling ind.	1.000	1.000	1.000	1.000	47
elliptical	0.667	1.000	0.667	1.000	3
rowing	1.000	1.000	1.000	1.000	7
AVG	**0.945**	**0.981**	**0.981**	**0.980**	

Table 4.7. Classification Report for Random Forest evaluated on test set - Dataset A



predicted \ true	walk	smoke	run	bike	car	cycling	elliptical	rowing
walk	52	0	0	0	0	0	0	0
smoke	0	27	0	0	1	0	0	0
run	0	0	9	0	0	0	0	0
bike	0	0	0	21	0	0	0	0
car	0	0	0	0	44	0	0	0
cycling	0	0	0	0	0	47	0	0
elliptical	0	0	0	1	0	0	2	0
rowing	0	0	0	0	0	0	0	7

Figure 4.8. Confusion Matrix for Random Forest evaluated on test set - Dataset B

	Accuracy	Precision	Recall	F1- Score	Support
walking	1.000	1.000	1.000	1.000	52
smoking	0.964	1.000	0.963	0.982	28
running	1.000	1.000	1.000	1.000	9
cycling out.	1.000	0.954	1.000	0.977	21
car driving	1.000	0.978	1.000	0.989	44
cycling ind.	1.000	1.000	1.000	1.000	47
elliptical	0.667	1.000	0.667	0.800	3
rowing	1.000	1.000	1.000	1.000	7
AVG	**0.954**	**0.991**	**0.990**	**0.990**	

Table 4.8. Classification Report for Random Forest evaluated on test set - Dataset B

Chapter 5

Conclusions

In this work, the possibility of using the Apple watch as a recognition device for human activities was therefore investigated. Through the development of Motion Data Logging, it is possible to create a dataset of 8 distinct activities. The goodness of the acquired datasets have been proved experimentally in this thesis, by performing HAR by means of the most basic and widespread classifiers
This work provides an experimental evidence that the information that can be extracted from the apple watch sensors is remarkable and can be succesfully exploited by simple classifiers in order to discriminate between a number of possible human activities of interest. The data acquisition and preprocessing is performed internally, fact that facilitates the development of user-customed activity recognition systems. The choice to use simple feature selection algorithms and simple recognition models has been considered in the perspective of subsequent works that include real time recognition: other types of algorithms widely used such as Dynamic Time Warping have been discarded a priori due to the high computational cost they require. All of the basic recognition models studied in this work have achieved promising perdormances. Random forest in particular have achieved the best results in all parameters evaluated in both validation and test set, but every model performance can be considered excellent.
Further suggested works related may be the further development of the software, trying to investigate real-time recognition by exploiting local computing capacity or upgrading it using the latest generation Apple Watch, which allows to record of the entire ECG trace from the derivation of the wrist in which it is located, so as to investigate how this entire can be informative in the context of the recognition of human activities.
Finally, the introduction of partially supervised models that envisage the recognition of new activities based on the user experience could be an interesting perspective for the continuation of the work done so far.

Bibliography

[1] S. Waltzer, *Global Smartwatch Vendor Market Share by Region: Q4 2018*, Strategy Analytics, feb. 2019.

[2] J.B. Su, *Apple Watch 4 Is Now An FDA Class 2 Medical Device: Detects Falls, Irregular Heart Rhythm* , Forbes, 2018.

[3] O. Lara e M.A. Labrador, in IEEE COMMUNICATIONS SURVEYS & TUTORIALS *A Survey on Human Activity Recognition using Wearable Sensors*, vol. 15, n. 2 ,2013.

[4] B. Hu, Y. Chen, E, Keogh, *Time Series Classification under More Realistic Assumptions*, Data Mining Knowledge Discovery, 2016, 30:403?437

[5] Z.S. Abdallah, M. M. Gaber, B. Srinivasan, S. Krishnaswami, *Activity Recognition with Evolving Data Streams: A Review*, ACM Computing Surveys, vol. 51, n. 4, art. 71, 2018

[6] G. De Leonardis, S.Rosati, G.Balestra, V.Agostini, E.Panero, L.Gastaldi, M.Knaflitz,*Human Activity Recognition by Wearable Sensors Comparison of different classifiers for real-time applications*

[7] S.Rosati, G. Balestra, M. Knaflitz, *Comparison of Different Sets of Features for Human Activity Recognition by Wearable Sensors, Sensors*, 2018

[8] I. Amezzane , Y. Fakhri, M. El Aroussi and M. Bakhouya,*Towards an Efficient Implementation of Human Activity Recognition for Mobile Devices, EAI Endorsed Transactions on Context-aware Systems and Applications*, 2018

[9] H. Xu, J. Wang, Z.Huang, Z. Kang, *Study on Fast Human Activity Recognition Based on Optimized Feature Selection* , 2017 ,16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science

[10] A.Jordao, L.A. Borges Torres, W.R.Shwartz, *Novel approaches to human activity recognition based on accelerometer data* , Signal, Image and Video Processing (2018) 12:1387-1394, 2018

[11] J. Suto, S.Oniga, P.Pop Sitar, *Feature Analysis to Human Activity Recognition*, INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL ISSN 1841-9836, 12(1):116-130, February 2017

[12] `https://www.macrumors.com/2014/09/26/iphone-6-6-plus-two-accelerometers`

[13] https://mbientlab.com/community/discussion/1608/problems-with-acc-gyro-sample-rate-in-ios

[14] http://www.yfp.elisacorteggiani.com/spettrofotometro.html

[15] https://developer.apple.com/documentation/coremotion/cmmotionmanager

[16] D. Micucci, M. Mobilio and P. Napoletano , *UniMiB SHAR: A Dataset for Human Activity Recognition Using Acceleration Data*, Applied Sciences, 2017

[17] https://developer.apple.com/documentation/coremotion

[18] https://developer.apple.com/documentation/corelocation/cllocationmanager

[19] https://developer.apple.com/documentation/healthkit/setting_up_healthkit

[20] https://developer.apple.com/documentation/healthkit/data_types

[21] https://developer.apple.com/documentation/healthkit/reading_data_from_healthkit

[22] https://medium.com/@sdrzn/swift-4-codable-lets-make-things-even-easier-c793b6cf29

[23] www.json.org

[24] https://support.apple.com/it-it/HT204562

[25] https://developer.apple.com/documentation/watchconnectivity/wcsession

[26] https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

[27] Abdullah Mueen, Eamonn J. Keogh, *Extracting Optimal Performance from Dynamic Time Warping*, KDD 2016: 2129-2130

[28] P. Siirtola, P. Laurinen, E. Haapalainen, J. Roning , *Clustering-based Activity Classification with a Wrist-worn Accelerometer Using Basic Features*

[29] X.Heng, Z. Wang and J. Wang, *Human activity recognition based on transformed accelerometer data from a mobile phone* INTERNATIONAL JOURNAL OF COMMUNICATION SYSTEMS Int. J. Commun. Syst. 2016, 29:1981-1991

[30] K. Altun, B. Barshan, O. Tunc-el, *Comparative study on classifying human activities with miniature inertial and magnetic sensors*, Pattern Recognition ,2010, 43: 3605-3620

[31] T. Huynh, B. Schiele, *Analyzing Features for Activity Recognition*, Joint sOc-EUSAI conference, 2005

[32] W.Kirch, *Encyclopedia of Public Health*, 2008, 1090-1091

[33] C.Bishop, *Pattern Recognition and Machine Learning* , 2006, 338

[34] https://en.wikipedia.org/wiki/Ball_tree

[35] https://en.wikipedia.org/wiki/K-d_tree

[36] C.Bishop, *Pattern Recognition and Machine Learning* , 2006, 124

[37] C.Bishop, *Pattern Recognition and Machine Learning* , 2006, 663

[38] T.Ho, *Random Decision Forest* , 1995