# Recognition of transport methods from GPS and other mobile sensor data

Alvar San Martín

Instituto Superior de Engenheira de Coimbra

Abstract

With the introduction of smartphones, we are able to carry a big number of sensors that collect data in real time with us, making possible to compute information on the run and sending it in real time or in batches once a certain activity is done or we have access to a network.

This article explores the collection of data from different smartphone devices over Portugal, recording most of the data in the city of Coimbra. The data is recorded from several different phones from students of the ISEC all of them using an Android application developed for the project "*SenseMyCity*" [1] this provides a framework to store and access the raw data, but sadly not to visualize it anymore.

We can combine the previous mentioned data with classification algorithms to automatically determine the activity, in this case the transport method of a user with a good accuracy.

*Index terms*: Sensors, GPS, Mobile, Ubiquitous computing, Transport classification

## 1. Resume

We start from the premise of the SenseMyCity project [1] of using ubiquitous computing and the current networking structure of our cities to get valuable information in order to engineer solutions to optimize the citizen interaction with its environment, giving him more information or predicting its necessities, like transport, consumption, socialization etc.

In order to face these challenges, we are collecting data from different smartphones in unknow positions and applying Ambien Intelligence[1] and ubiquitous computing[2] to extract information we can use on machine learning algorithms that can classify which transport method are we using.

This article involves the development of some programs and scripts to process the data after is forked from the SenseMyCity server, which we do not control, once we have the data, its pre-processed, features are extracted and finally we train a machine learning algorithm capable of doing a classification according to the transport methods we have previously recorded, obviously the aim would be to implement the resulting model in a program that uses that real-time classification to support other tasks.

In a nutshell the objective of the article and the program developed is to create a method to understand the user environment better and not to actually create solutions but the base to develop them.

[1] https://en.wikipedia.org/wiki/Ambient_intelligence
[2] https://en.wikipedia.org/wiki/Ubiquitous_computing

## 2. Data collection

The data collection is performed by students of the ISEC in Coimbra using the Android application from SenseMyCity [1] which records data in a standardized way, enabling the data processing to be relatively straight froward considering we are using very different phones.

| Model | A-GPS | 4G(LTE) | Memory | RAM | Sensors | O.S. |
|---|---|---|---|---|---|---|
| Xiaomi Mi A2 | X | X | 64GB (4GB RAM) | Up to 21h | Accelerometer, Gyroscope, Proximity | Android 9 |
| Xperia XA1 | X | X | 3GB RAM | | Proximity, Light, Accelerometer, Compass, Hall | Android 7 |
| Xiaomi Mi 10 Lite | X | X | 128GB (6GB RAM) | Up to 24h | Accelerometer, Gyroscope, Proximity | Android 11 |
| Huawei P30 Lite | X | X | 128GB (4GB RAM) | Up to 21h | Accelerometer, Gyroscope, Proximity | Android 10 |
| Realme X2 | X | X | 128GB (8GB RAM) | Up to 30h | Accelerometer, Gyroscope, Proximity | Android 11 |
| Samsung S10 | X | X | 128GM (6GB RAM) | Up to 33h | Accelerometer, Gyroscope, Proximity, barometer, heart rate, SpO2 | Android 12 |
| Sony Xperia Z5 Compact | X | X | 32GB (2GB RAM) | Up to 11h | Accelerometer, Gyroscope, Proximity | Android 7.1 |
| Smasung A31 | X | X | 128GB (6GB RAM) | | Accelerometer, Gyroscope, Proximity | Android 11 |
| Xiaomi Mi 10T Pro 5G | x | x | 128 GB (8GB RAM) | | Accelerometer, Gyroscope, Proximity | Android 11 |
| Samsung Galaxy S9 | x | x | 64GB + 4GB RAM | | Accelerometer, Gyroscope, Proximity, barometer, heart rate, SpO2 | Android 10 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Samsung Galaxy S8 | x | x | 64GB (4GB RAM) | Up to 20h | Accelerometer, Gyroscope, Proximity, barometer, heart rate, SpO2 | Android 9 |
| Asus Zenfone 3 ZE520KL | x | x | 64GB 4GB RAM | | Fingerprint (rear-mounted), accelerometer, gyro, proximity, compass | Android 8 |
| Medion B5532 | X | X | 64GB 4GB | | Accelerometer, Gyroscope, Proximity, barometer | Android 7 |
| Xiaomi Mi 10T Pro 5G | x | x | 128 GB (8GB RAM) | | Accelerometer, Gyroscope, Proximity, barometer | Android 11 |
| Samsung Galaxy A70 | X | X | 128GM (6GB RAM) | | Accelerometer, Gyroscope, Proximity | Android 11 |

*Figure 1: Phones used on the data collection*

Even though the phone models and brands are different the Android phone market is dominated by Qualcomm and MediaTek as processor manufactures [2] and Samsung as producer of other components like RAM memory, cameras or flash storage so the only outlier would be the Huawei phone, since they produce its own silicon. But at the end we cannot extract concussions from the device used because the app doesn't store the hardware used.

### 1.1.The Application

After using the mentioned application, in its version 4.6 for 3 months (April to June during 2022) we can summarize some perks, defects and challenges of the application of the software.

The process to save data is vey simple and effective, the app saves the data locally on a local database in the memory of the phone and when you have connection to a good enough

network you choose to send it to his server database, where is saved on your profile, identified by a google account [1].

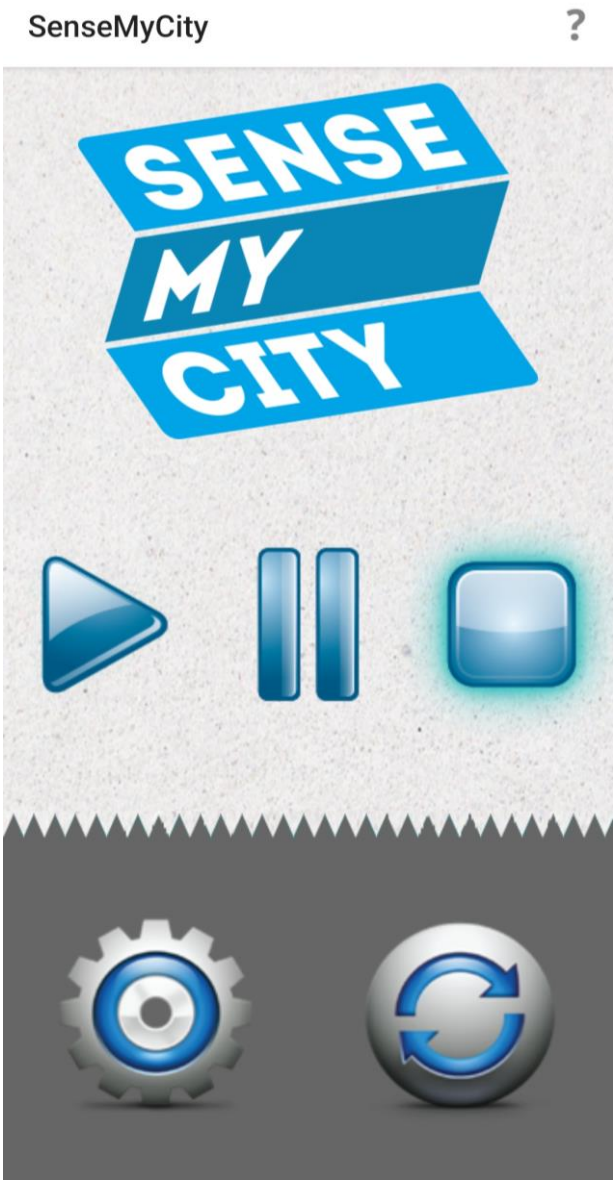


*Figure 2: Main screen of the Sense my city application*

The main controls (play, pause, stop) are mean to start the recording of a trip, pause the recording and stop and finish the recording, just below them we have two icon for settings (left) and to upload data to the server (right), but there is a “secret” button in that screen that is the top logo of SMC which is an monitor of the sensor of the phone, but only works when

we are recording a trip. Finally in the top right we can see a "?" icon which is supposed to provide use information about the app, but it crashes in Android 12.

Even if the application was updated in 2019 is clear that it was designed in 2011 for Android 4 and we are noticing some unexpected behaviour during the recordings and when we send it to the server like: sudden tops on the recording, failing to keep the GPS service active, failing to send the data to the server in certain networks, generally fails with Wi-Fi networks while it works on 4G – 5G ones, some other times the server just returns a "server in maintenance" error.

About the design problems I found some significant ones:

- Is not possible to label the recordings on the phone, which is a very possible source of human error when labelling data afterwards, since we have relied on other methods to keep a history of our activity and is easy to mistake routes if we have several in the same day.
- We can hardly control the sample rate for the accelerometer and gyroscope, the option by default is unknown and is just labelled as "Normal" but the rate for the accelerometer is 50hz and for the gyroscope 10hz. The other options are "Fast", "Faster" and "Fastest" but we don't know the exact meaning of the classification.
- The way the data is stored is unknown in most of the times because not enough information is provided in the documentation of the application.

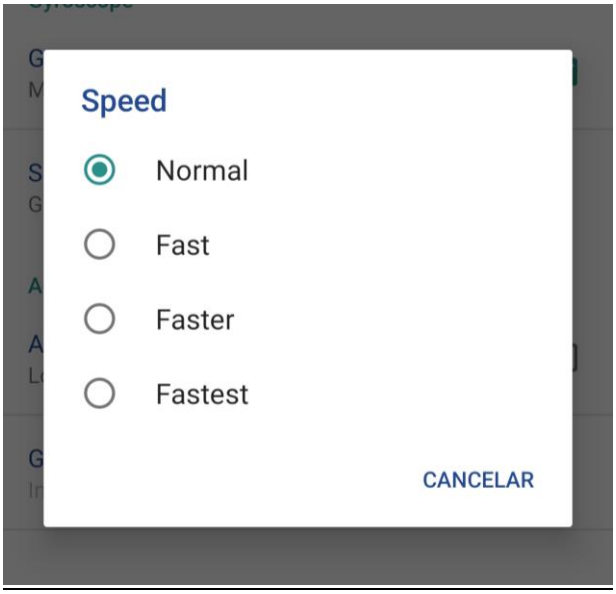


*Figure 3: Accelerometer speed options*

### 1.2.The sever and visualization

Once we have sent the information to the server it is stored on a relational database [1] , we cannot access directly but trough a web application which let us make some queries and return them on the shape of csv files compressed in a zip file. The web application[3] is supposed to provide an interface to visualize our trajectories on a map, but it does not work

[3] https://sensemycity.pt/smc/

anymore, so we cannot know the exact GPS positions unless we use external tools.

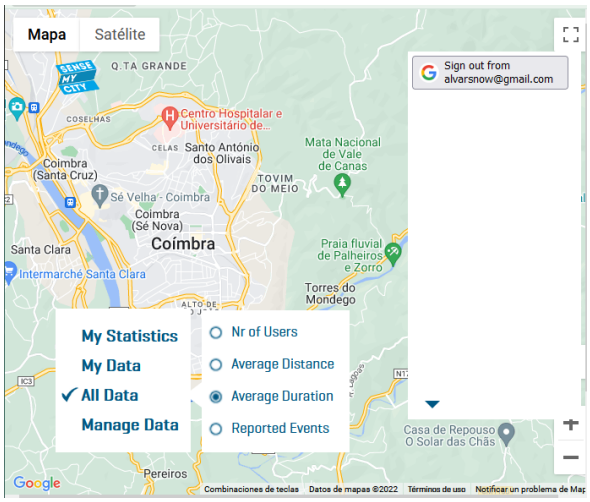


*Figure 4: SMC web application*

### 1.3. Data processing from SMC

The first challenge we face is obtaining all the information from the different students and merging it together so we can use it on the project, for this propose an FTP server is used, and every student is responsible for labelling and uploading its own recordings.

Once we have collected all of the recordings from the server a filter is needed to separate the ones lacking sensor files, some of the recording updated don't have all the files needed for the project and we need 3 basic ones:

- Accelerometer
- Gyroscope
- GPS

Most of the times the missing file was the Gyroscope one, rarely the accelerometer or the GPS, this can happen due the sensor being disable on the phone or the recoding being disabled on the SMC app configuration, the exact reason is unknown.

When the data is filtered it is divided in trips and instants of that trips, of one second each and is uploaded to PostgreSQL server so we have flexibility adding new lines and querying the existing ones. In addition a extra table is added to enable spatial visualizations.

At the end of the process we have 33461 instants that equal 33461 seconds of recordings, in total the space occupied is 80MB which is relatively small compared to other projects like in *Sussex-Huawei Locomotion-Transportation Recognition Challenge* [3] where the complete dataset weighted 58 GB, but had significantly more columns and transport methods to classify.

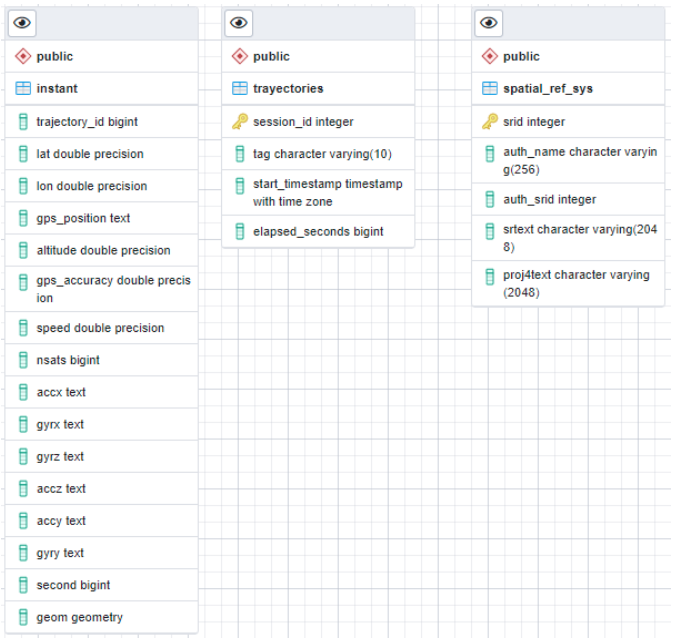


*Figure 5: Local database structure*

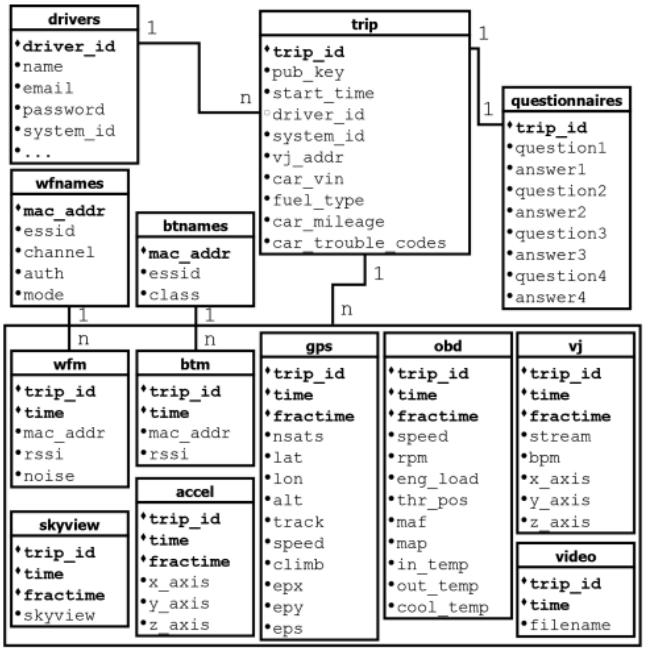


*Figure 6: SMC Database structure [1]*

## 2. Data analysis, pre-processing and feature extraction

Once we have our data divided in seconds over some trajectories done in a single transport method, we can start describing the process to reach a usable dataset for training and testing a model.

### 2.1.The classifications/labels

We have only information about the transport methos we normally use in Coimbra, which are: *Walking*, *Car*, *Bike*, *Bus*, *Run* and standing *Still*. Because the data collection was not planned in depth, and we had to discard some trips the resulting dataset is unbalanced, as seen in the next table (Figure 7):

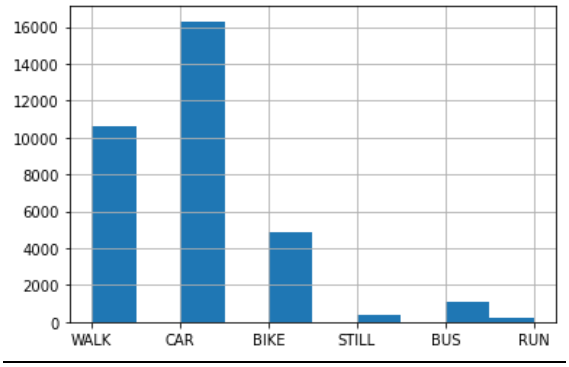


*Figure 7: Histogram of the transport methods in the dataset*

## 2.2. Sensor information

When we analyse each trip, we also find some inconsistencies in the rate of sampling of the accelerometer, in most of the cases the sample rate changes unexpectedly in apparently random points, as shown in Figure 8, but it is not a big issue considering windowing of data.

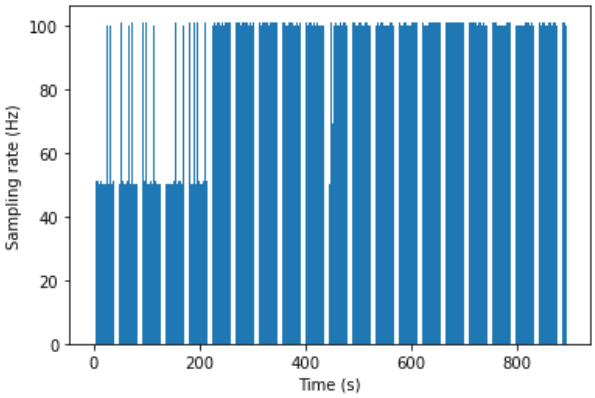


*Figure 8: Sampling rate in Hz over time for a trip*

In the other hand we have the gyroscope recordings, running at a sampling rate of 5hz in some devices and in a rate 100hz, which unless we want to create features involving the sync of the gyroscope and accelerometer, don't really need to process.

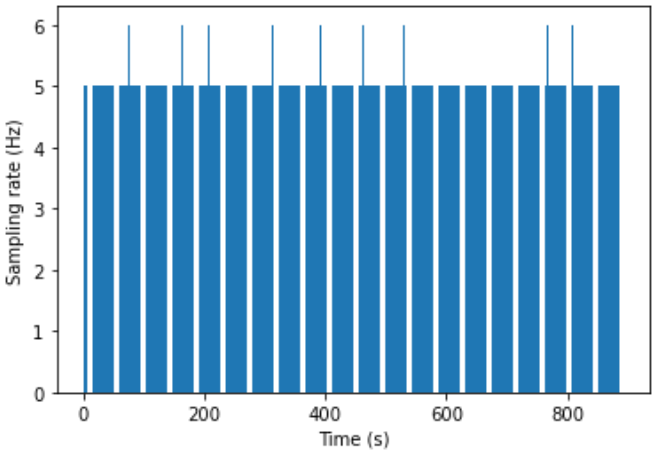


*Figure 9: Device sampling at 5hz*

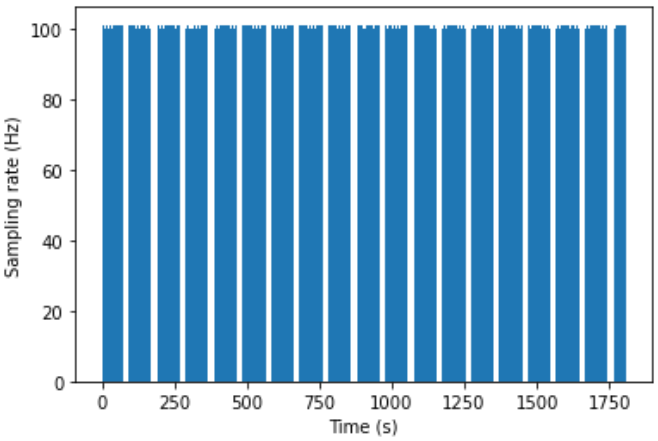


*Figure 10: Device sampling at 100hz*

After we have found the differences between recording, we have to focus on what they have in common, all of them can be processed as waves so we can make calculations and representations based on this fact.

In the case of the accelerometer and gyroscope we have 3 readings (Figure 11), one for each axis, and because we don't know the position of the device in each moment or at the start of the recording, we cannot know the orientation of the phone with precision so instead of keeping 3 separated readings we are adding all in a single wave that represents acceleration in one dimension (Figure 12).

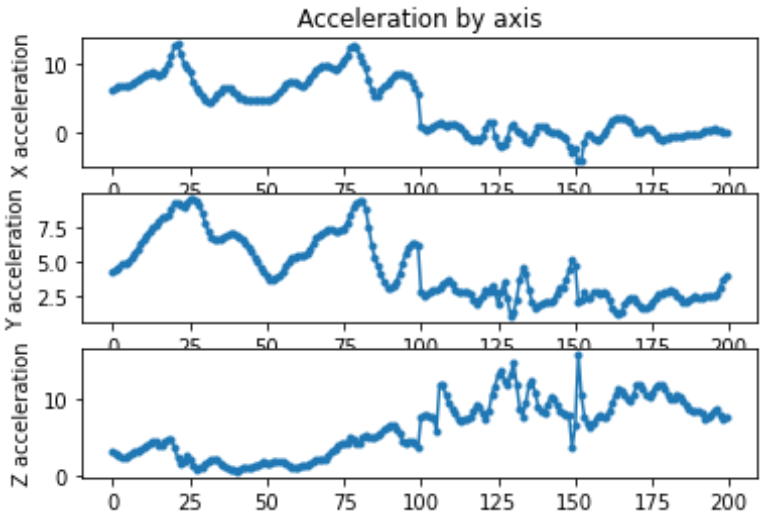


*Figure 11: Acceleration during a walk, 3 axes*

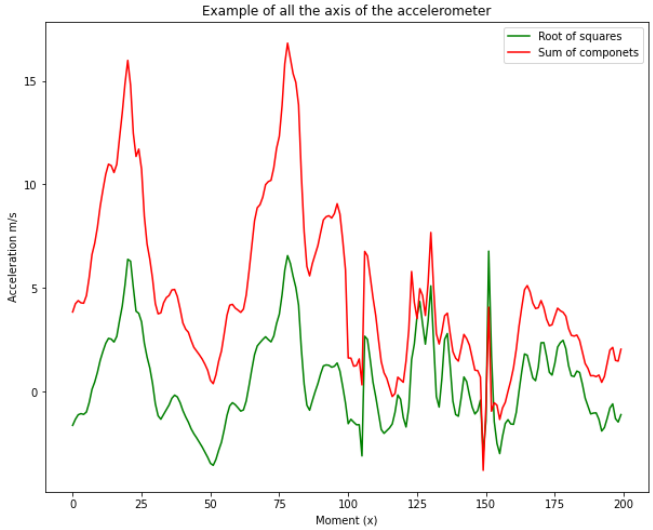


*Figure 12: Combined axis of the accelerometer during a walk*

I have experimented with 2 methods to combine the axes, normal sum of the components and root of the squared sum of components (Figure 12). In the case of the accelerometer the estimated value of gravity is subtracted to compensate the natural bias of the accelerometer recording the gravitational force in a certain axis.

$$X_{acc} = a_x + a_y + a_z$$

*Figure 13:Sum of components*

$$Z_{acc} = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

*Figure 14: Root of squared sum of components*

I have chosen to use the second option because it helps reducing the effect of the outliers in the signal. The process for the accelerometer data and gyroscope data is the same since both are accelerations.

Once we have a single wave for each sensor we can extract features, not only in time domain but in the frequency's domain, using a Fourier Transform, and get amplitudes of certain frequencies, equally as useful as the acceleration values in the time domain [4]. The algorithm used is the *Fast Fourier Transformation* (FFT) that haves a algorithmic complexity of O(nlog(n)) instead of the $O(n^2)$ complexity of a *Discrete Fourier Transformation* (DFT) which means that if we are computing a window of 10 seconds with a sample rate of 100hz a FFT is going to take 3.000 units of time instead of the 1.000.000 units of the DFT [4], which is 333 times faster.

In the case of the frequency's domain, because is symmetrical and most of the frequencies are useless or considered noise, first we delete half of the space, then we discard the DC frequency, which in this case is always the 0hz and finally we apply a high-pass filter which only keeps the top 25 percent of the frequencies.

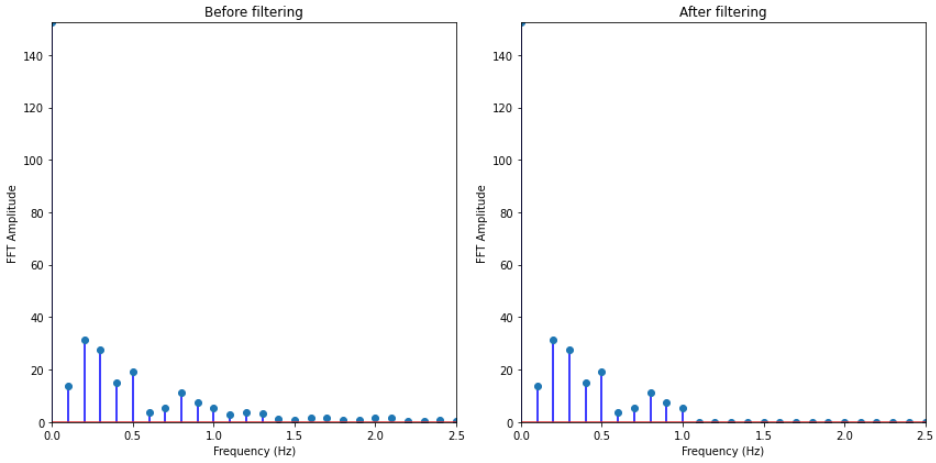


*Figure 15: Before and after a High-Pass*

### 2.3.Window and feature extraction

Having analysed how the data can be interpreted in the different sensors we can create a dataset with the data we are using for machine learning.

First, we have to choose a window, a big window size makes the data easy to classify, but in the real world we need it to be as small as possible to make the classification as fast as possible [5]. In this aspect we are constrained by the granularity of the data, because while the accelerometer and gyroscope have a granularity of at least 5hz the speed is recorded via GPS only at 1hz. To overcome this limitation, I have applied some overlap in the windows, so a certain range of data will be in 2 windows, having those windows different time ranges. This means we can classify data at twice the speed, but with similar latency, for example, if we have a window of 10s and an overlap of the 50%, the first window will be classified in 10s, but the second in 5s, but because contains data of the first window it may not detect a change in the activity until the next window.

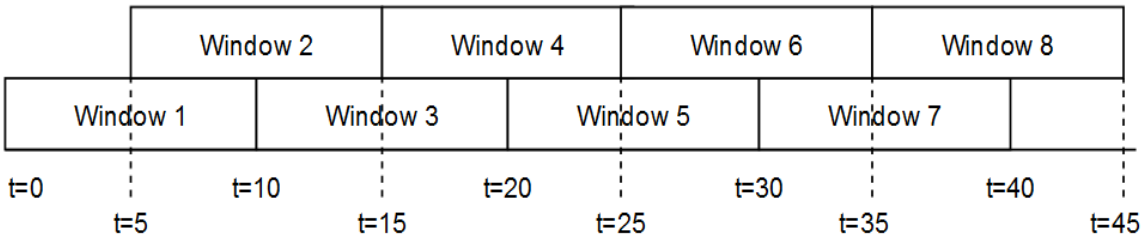


*Figure 16: Overlapping windows*

Another perk of the overlapping is the ability to get more information to train the model, specially in the cases where we have small amounts of data, like in the case of running and standing still.

The features are calculated from windows, and they are essentially a collection of common statistical measures:

- Mean
- Median

- Standard deviation
- Average absolute deviation
- Minimum
- Maximum
- Difference between maximum and minimum values
- Median absolute deviation
- Interquartile range
- Negative values count
- Negative values percentage
- Positive values count
- Positive values percentage
- Number of values above mean
- Number of peaks
- Skewness

This metrics are applied over this sets of data:

- Accelerometer, time domain
- Accelerometer, frequencies domain
- Gyroscope, time domain
- Gyroscope, frequencies domain
- Speed, time domain

After all is applied and some features are discarded for being useless, like the amount of positive and negative values in sets where there are no negative values, we have 62

features, not counting the label column, and a total of 6589 windows or rows with a 10s window, 50% offset.

## 3. Machine learning training

### 3.1. Algorithm selection

In other works the algorithm chosen is a decision tree [6] or a random forest [3] but, because all the features are numerical and can be scaled, this time I chosen logistic regression, one of the simplest methods to classify data, with a "*saga*" solver, the only in which we archive convergence with the amount and scale of the data we currently have. The number of iterations is set to 10.000, a usable amount in this case where we are using small amounts of data, and it takes a trivial amount of time to finish the training.

Once the model is trained and tested on shuffled datasets, we obtained these results:

```
Accuracy: 0.9730652503793626

 -------------Classification Report-------------

              precision    recall  f1-score   support

        BIKE       0.92      0.95      0.93       405
         BUS       0.91      0.86      0.88        78
         CAR       0.99      0.98      0.98      1294
         RUN       1.00      0.94      0.97        18
       STILL       0.96      0.96      0.96        27
        WALK       0.99      0.98      0.98       814

    accuracy                           0.97      2636
   macro avg       0.96      0.95      0.95      2636
weighted avg       0.97      0.97      0.97      2636
```

*Figure 17: Results of the classification with logistic regression*

This is not totally representative of how it could perform in the real world, since most of the classifications are made only on Bike, Car and Walk, due to the unbalance of the dataset, if we look at the confusion matrix, we notice some 2 common confusions, the Still - Car and the Car - Bike. It is expected to mistake Still with Car because in several times the Car could be stopped and the classifier is actually right, but there is no justification for the Bike - Car mistakes

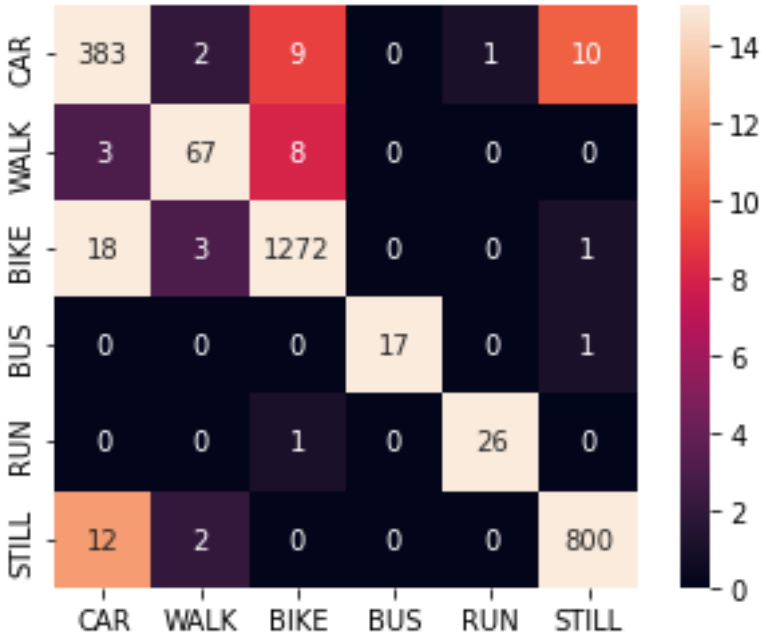


*Figure 18: Confusion matrix of the logistic regression model*

### 3.2. Validation

We cannot assume our model had good results just on the test set, so *Cross-Validation* is applied to try to estimate the performance on a more realistic environment. When the validation is performed, with 5 folds we obtain a accuracy of 83% and a standard deviation of 17% between the scores and with 10 we obtain a accuracy of 91% and a standard deviation of 10%, which is a more credible score considering the situation.

## 4. Conclusions

The process of extracting data and applying it in a useful context is never easy, even when all the users involved are similar, in this case the students, and using the same software and in this case, we have faced several challenges that had to be solved, most of the time thanks to python scripts to make the data usable.

In the machine learning area was surprising the accuracy we can archive with just 3 sensors and a few trips recorded, the performance of the logistic regression proves that in this case a sophisticated algorithm is not needed, but a correct analysis of the features we need.

## Referencias

[1] A. A. F. V. J. B. a. J. P. S. C. João G. P. Rodrigues, «A Mobile Sensing Architecture for Massive Urban Scanning,» *14th International IEEE Conference on*, pp. 1132-1137, 2011.

[2] T. Alsop, «Statista,» Statista, 6 May 2022. [En línea]. Available: https://www.statista.com/topics/7885/qualcomm/#topicHeader__wrapper. [Último acceso: 10 July 2022].

[3] M. G. C. M. D. M. N. R. M. L. a. M. G. Vito Janko, «ross-location transfer learning for the sussex-huawei locomotion recognition challenge.,» *Association for Computing Machinery*, p. 730–735, 2019.

[4] T. S. A. B. Qingkai Kong, Python programming and numerical methods, Academic Press, 2020.

[5] B. A. A. S. Young-Ji Byon, «Real-Time Transportation Mode Detection via Tracking Global Positioning System Mobile Devices,» *Journal of Intelligent Transportation Systems Technology Planning and Operations* , vol. 13, nº 4, pp. 161-170, 2009.

Table index

Figure 1: Phones used on the data collection.....5

Figure 2: Main screen of the Sense my city application.....6

Figure 3: Accelerometer speed options.....8

Figure 4: SMC web application.....9

Figure 5: Local database structure.....10

Figure 6: SMC Database structure [1].....11

Figure 7: Histogram of the transport methods in the dataset.....12

Figure 8: Sampling rate in Hz over time for a trip.....12