Behavioral Context Recognition

Silvio Di Martino
s.dimartino21@studenti.unisa.it
Universitá degli studi di Salerno
Salerno, Italy

Gerardo Martino
g.martino11@studenti.unisa.it
Universitá degli studi di Salerno
Salerno, Italy

Gigi Jr Del Monaco
g.delmonaco1@studenti.unisa.it
Universitá degli studi di Salerno
Salerno, Italy

ABSTRACT

The ability to automatically recognize a person's behavioral context can contribute to health monitoring, aging care and many other domains. Validating context recognition in-the-wild is crucial to promote practical applications that work in real-life settings. This paper illustrates methods and results of the exploration of the *Extrasensory* dataset, which contains measurements from sensors from the user's personal smartphone and smartwatch devices. The application of methodologies for *data analysis*, *data cleaning* and *data preprocessing*, and the final use of machine learning methods aims to distinguish, starting from the proposed dataset two different classes: *static behaviour* and a *movemented behaviour*. This can be useful in future domain research to detect users lifestyle, looking for sedentary people in order to propose some actions and some practices to improve their state of health. The purpose of our study will be presented, starting from the analysis and trasformation activity then a summary that expose the comparison between the models used and which of it yields give the best prediction.

KEYWORDS

Behavioral Context, Mobile sensors, Artificial intelligence, Machine learning, Human activity recognition.

1 INTRODUCTION

The advancement of smart devices with built-in sensors and intelligence can importantly helps to automatically recognize, monitor and in a certain circumstance support a person's lifestyle. The progress of smartphones and smartwatches in general are being used extensively to collect our daily activities, information that for the scientific field are very important to predict context behaviors and makes important contribution to improve methodology or applications in a health domain, they can contribute to recognize problems in real time and automatically solve or prevent them. Automatically recognition context will help detect critical situation and offer immediate support (e.g. an alcoholic patient may be at high risk of craving or lapse when the context is "at a bar, with friends"). The Internet of Things and IoT Data Analytics are another branches with very high potential, integrated in this scenario allow the study and analysis of more complex contexts, as well as being used for a stand alone applications. Therefore, can the data extracted from sensors in smartphones and wearable devices be used to recognize human activity and suggest how to improve various qualitative aspects of lifestyle? Can these data improve a personal assistant system to better serve the patient? In order to answer this question and achieve our goal, we start the exploration and analyze the dataset, with a general overview about the time spent performing each activity, the composition and consistency of the Extrasensory dataset and than the selection of a subset of labels considered necessary to address our study, labels that identify a static behaviours *(like 'sitting')* and dynamic behaviours *(like 'at the gym')*. The two classes identified are the starting point for further analysis about the behaviour of *single user* and *all the users* in the dataset, looking for a differences by comparing some features values recorded during a static and a movement activity. This stage has been followed by *data transformation, feature selection, data imputation and normalization* and all the other steps taken in order to make the dataset consistent for the training of the classification models and finally create a model for recognising human activity through the use of machine learning algorithms. The classification has been conducted with three different models of batch learning (*Logistic Regression*, *KNearest Neighbor* and *Gaussian Naive Bayes* and *MOA* regarding the online learning. In the next sections, we will discuss the state of the art of the problem, then we will analyse the dataset in more detail, as well as our approach for the prediction, explaining all our choices and our intentions.

2 STATE OF ART

The most important work in the behavioral context recognition field has been conducted by Vaizman et al., proposing an ExtraSensory dataset that consists of sensory data and labels collected from 60 users (34 female subjects and 26 males subjects), for approximately 7 days, were the users involved conduct their own behaviour life in an authentic subjective and manner. Vaizman et al. [6], showed that those sensors are very useful and the features helped users recall their past behaviour. The additional watch devices has been used and turned out to improve the data collecting and keep the users interaction from interfering with natural behaviour.

However other attempts to collect data in-the-wild have been conducted with different approaches to acquire context labels: for example Pirsiavash et al. [3], present a novel dataset and novel algorithms for detecting activities of daily living in first person camera views, with wearable cameras. This approach provides a practical advantage of ease of capture and so it can be also integrated with other approaches.

Another approach is in-situ self-reporting, where participants report their own context (e.g. location, activity, emotion, etc.) in real-time. The Experience Sampling Method (ESM) [5], is a technique where the participant is prompted at different times to fill a short form and report their context. Compared to retrospective reports of feelings, ESM is less influenced by memory biases.

Han et al. [1], conducted a study starting from the ExtraSensory dataset and stated that audio is an important feature for the dataset

as it has a strong relative accuracy component to the context in which the user records the activity on the app.

Several studies have used MLP and other models as black box tools to recognize activities from mobile sensors. Mantyjarvi et al. [2], used two accelerometers on the waist to recognize four body movements. The use of principal component analysis and independent component analysis with a wavelet transform is tested for feature generation. Recognition of human activity is examined with a multilayer perceptron classifier. Best classification results for recognition of different human motion was between 83% to 90%, and they were achieved by utilizing *independent component analysis and principal component analysis.*

Radu et al. [4], used Neural Networks, in particular CNN (convolutional neural network) and DNN (Deep Neural Network), to compute a classification in order to obtain information about context behavior, using wearable sensors.

So in the behavioral context recognition field different approaches have been used for both collecting data and computing a classification. We want to use machine learning classification models and pay attention on two specific context group activities: *"movement"* and *"static"*, aiming to understand what people do in an ordinary day, in order to improve the quality of life. Then, our focus will be only on this two aspects, in order to study deeply this context.

3 DATA DESCRIPTION AND FIRST ANALYSIS

The *ExtraSensory* dataset has thousands of instances by 60 users. These data are distributed over 278 features between various sensors, typically taken in intervals of one minute. These data were collected from personal smartphones, and possibly additional convenient devices, like watches (56 out of the 60 agreed to wear the watch). The users used their own phone with *Extrasensory application* for collecting in-the-wild data, including sensor measurements and self-reported labels describing people's behavioral context, for example: driving, eating, in class, etc. Every minute the app records a 20sec window of sensor measurements from the phone and watch, collecting data from the accelerometer, magnetometer, location, and phone-state sensors and as well accelerometer and compass from the additional smartwatch. The data collection performed on real-life scenarios was accompanied by first label describing the main activity, like *walking, sitting etc...*, and a secondary label describing more specific context in different aspects like: *sports (e.g. playing basketball, at the gym), transportation (e.g. drive - I'm the driver, on the bus), basic needs (e.g. sleeping, eating, toilet), company (e.g. with family, with co-workers), location (e.g. at home, at work, outside).*

3.1 Our Work

The *ExtraSensory* dataset has been extracted in a specific drive folder in order to prepare and transform the data into a way that is more suitable for modelling. The use of a defined method helps to parse the *CSV* files from the dataset, in order to get all the data for a *single user or all of them.* As said before, we select a subset of labels and divide them in two different groups, as shown in Figure 1:

```
static_labels = ['Sitting', 'Lying down', 'Sleeping', 'In a meeting',
'In class', 'Computer work', 'Watching TV', 'Drive - I\'m a passenger']

movement_labels = ['Walking', 'Cooking', 'Cleaning', 'Shopping',
'Running', 'Bicycling', 'Drive - I\'m the driver', 'Strolling', 'Doing
laundry', 'At the gym', 'Stairs - going up', 'Stairs - going down']
```

Figure 1: Labels subsets

After doing this, the first study was carried out by extracting the data of a single individual chosen randomly, looking for how much time he spent performing each activity selected in those two group of labels shown in Figure 2.:

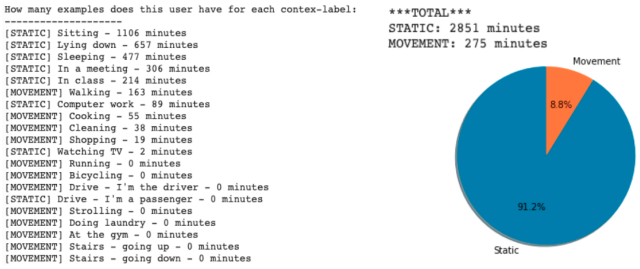

Figure 2: Example for single user

We can immediately see from the pie chart that this user tends to spend more time in a *static position* than in *movement* ones. We expanded this first analysis to all the 60 users, and the idea has been confirmed: all the users spend more time in a static position than in a movement one, as shown in Figure 3.

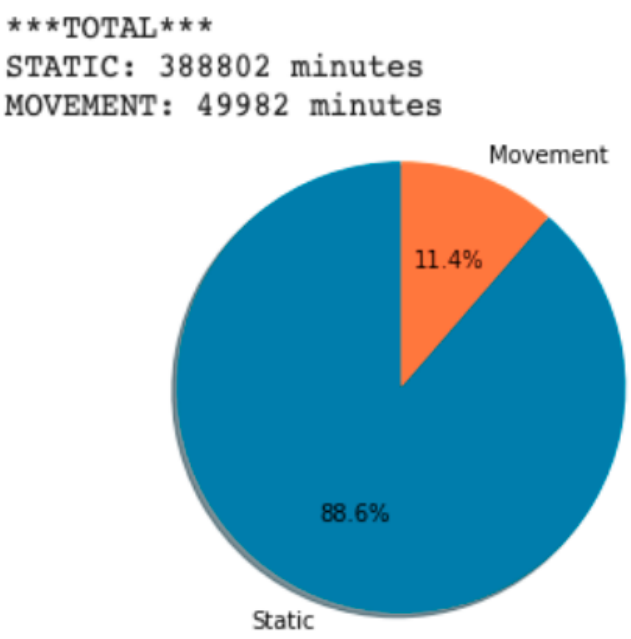

Figure 3: Static and Movement chart for all users

Since our final goal is to detect the static behaviours and the movement ones, we are going to calculate the overall time spent doing each static or movement activity.

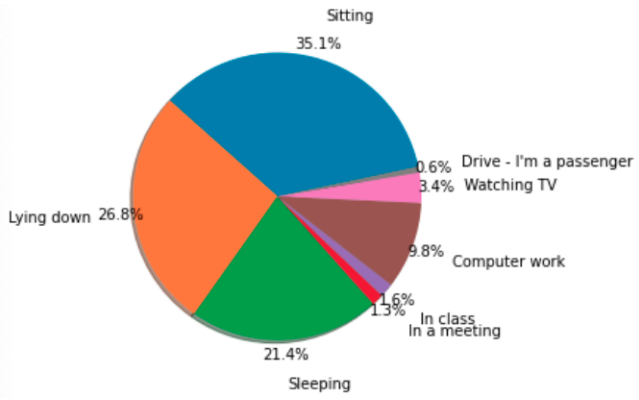

Figure 4: Static labels chart exploration

In the Figure 4, we immediately see that in the static labels *lying down* and *sitting* have a very high percentage, followed by *sleeping* and all the others that have a small percentage with respect to the other.

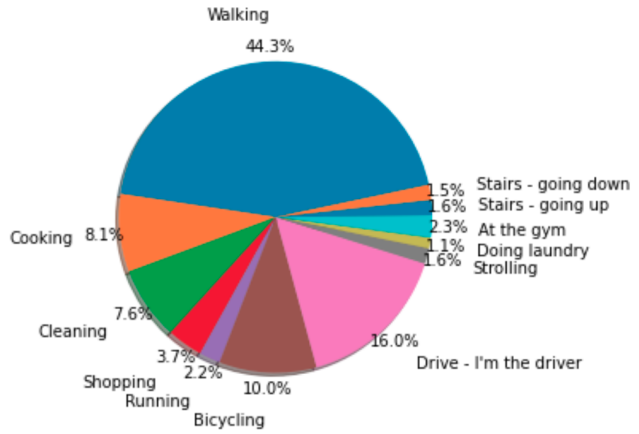

Figure 5: Movement label chart exploring

In the Figure 5, the movement activities, *walking* has a higher percentage by far, while all the others range between 16% and 1.5%. In summary, we can state that in the overall dataset, data referring to *static* behaviors are predominant with respect to *movement* ones. At this point, the work requires further investigations: we want to detect the most influential features that clearly differentiate a static behaviour from a movement one. According to the studies conducted by *Vaizman et al.* [6], there are different features that play a role as well: let's see them.

4 FURTHER ANALYSIS AND FEATURES SELECTION

4.1 Single-user study

The *Extrasensory* dataset has features extracted from ten types of sensors. Except for phone state sensors which have discrete categories such as battery state and WiFi availability, other sensors have many extracted features coming from the same source of time-sequential raw data: this means that, related to the acceleration, the dataset provide the mean, the standard deviation, the percentiles and so on. The features are less intuitive to interpret than the labels. They were calculated from the raw-measurements from the various sensors on the smartphone or smartwatch. We can start by looking at the feature names, where the prefix of each feature indicates the sensor where it cames from. For convenience, we add a function to parse the feature names and provide the *code-name* of the sensor they belong to. We'll use the code-names as they appear in the original ExtraSensory paper: Acc (phone-accelerometer), Gyro (phone-gyroscope), WAcc (watch-accelerometer), Loc (location), Aud (audio), and PS (phone-state) as shown in Figure 6.

```
 0) Acc        raw_acc:magnitude_stats:mean
 1) Acc        raw_acc:magnitude_stats:std
 2) Acc        raw_acc:magnitude_stats:moment3
 3) Acc        raw_acc:magnitude_stats:moment4
 4) Acc        raw_acc:magnitude_stats:percentile25
 5) Acc        raw_acc:magnitude_stats:percentile50
 6) Acc        raw_acc:magnitude_stats:percentile75
 7) Acc        raw_acc:magnitude_stats:value_entropy
 8) Acc        raw_acc:magnitude_stats:time_entropy
 9) Acc        raw_acc:magnitude_spectrum:log_energy_band0
10) Acc        raw_acc:magnitude_spectrum:log_energy_band1
11) Acc        raw_acc:magnitude_spectrum:log_energy_band2
12) Acc        raw_acc:magnitude_spectrum:log_energy_band3
13) Acc        raw_acc:magnitude_spectrum:log_energy_band4
14) Acc        raw_acc:magnitude_spectrum:spectral_entropy
15) Acc        raw_acc:magnitude_autocorrelation:period
16) Acc        raw_acc:magnitude_autocorrelation:normalized_ac
17) Acc        raw_acc:3d:mean_x
```

Figure 6: Some Extrasensory features

Knowing this, let's consider some potentially important features that must be useful to detect different behaviours and let's see how they vary overtime, performing a specific kind of activity. We define a function that takes as input two labels and the feature's index that we want to analyze (the feature index is taken from the list of features shown above).

According to the state of art and our assumption, we continue the study focusing on the following features:

- `raw acceleration mean`
- `watch acceleration mean`
- `min speed`
- `max speed`
- `audio properties: max abs value`
- `battery level`

We made a first comparison between data labeled as *sitting* and data labeled as *walking* for the *single user*, considering the distribution value of the *raw acceleration mean*.

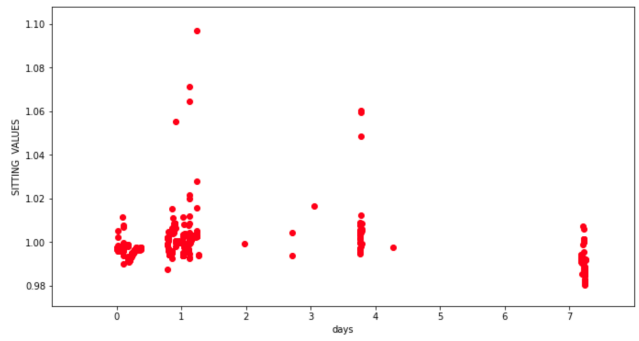

Figure 7: Sitting distribution

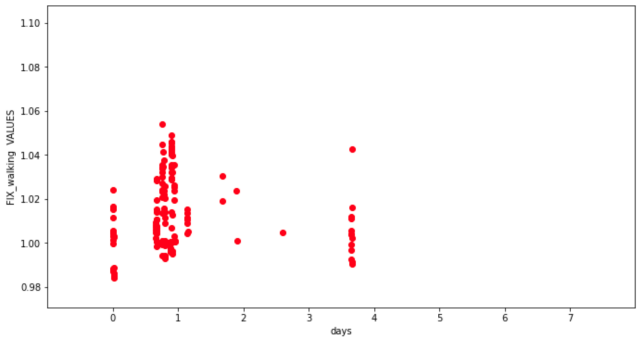

Figure 8: Walking distribution

Analyzing the two graphs (Figure 7 and 8), we noticed that the values are more constant in *sitting* then *walking*. In Figure 8, after the 4th days, the graph shows no longer *walking* activity, that means that one of the plausible reasons is that this specific *user* tend to be more proactive during the working days, so that they spend more time in logging their daily activities via the extrasensory applications, same reasoning will be adopted for the*sitting* value, as shown in Figure 8.

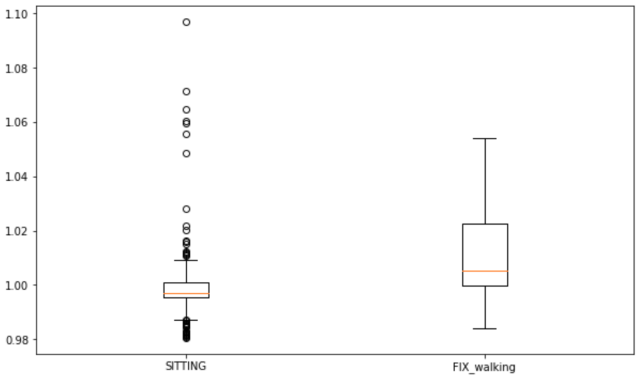

Figure 9: Sitting vs Walking comparison

The Figure 9 shown evident differences in values between *sitting and walking*; a basic statistical analysis shows that for *sitting* values we have a simmetric data distribution, so that we will analyze the two box plots step by step. From the boxplot of the *sitting* distribution the values are concentrated below 1.00, especially between the 25% and 50% percentile, we also note the presence of other widespread values between 1% and 25% and above 75%, they are undoubtedly outliers. Probably these outliers are attributed to events like traveling on private or public transportation as a passenger, and therefore would explain the behaviour of these values. The hypothesis is also supported by the fact that there are no outliers in the *walking* label, probably because the values are more consistent with each other, and with a more consistent distribution, ranging from 1.00 to 1.02 in the range of 25% and 75% of percentile.

Now we are going to make another comparison, again between the two labels, but analyzing the values of the *watch acceleration mean*:

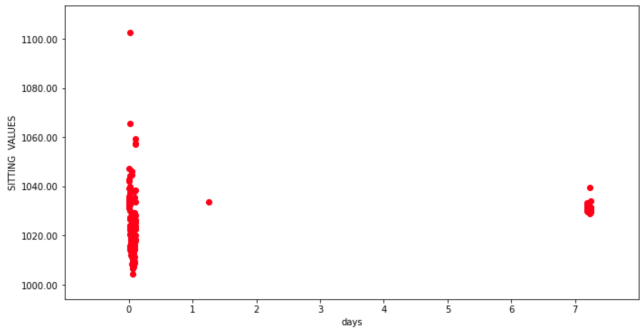

Figure 10: Sitting acceleration mean distribution

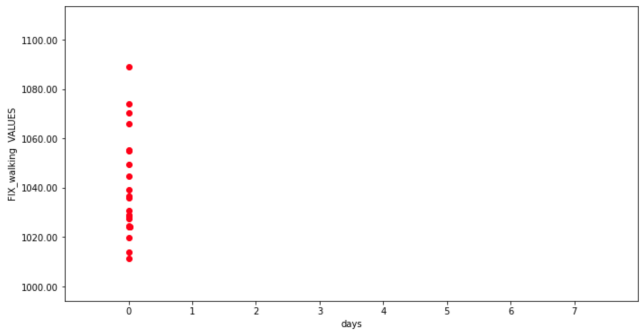

Figure 11: Walking acceleration mean distribution

As can be seen from the scatter histograms (Figure 10, 11) and the box plots (Figure 12), the quantity of values of *sitting* label and *walking* is too low to be able to draw an accurate conclusion, looking at the two scattered graphs the values appear only on the first few days out of 7. The explanation of that inconsistency can be that the user didn't use the wearable devices, so the acceleration values haven't taken. The box plot is also different from what we have analyzed for raw acceleration, as we see higher values for both

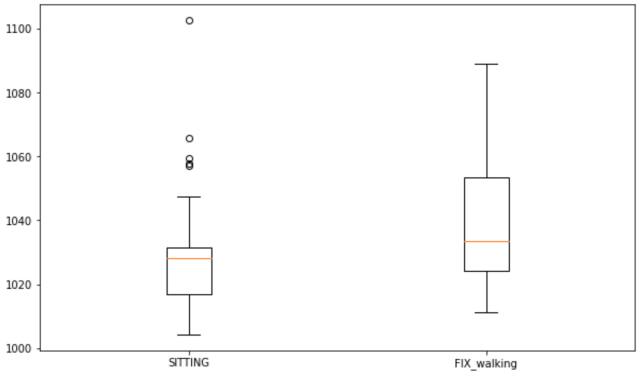

Figure 12: Sitting vs Walking acceleration mean distribution

labels, but poorly in quantity. The analysis continue considering two more features. We compared *min_speed and max_speed* of the user, during the *sitting* and *walking* events, as shown in Figure (13 to 15) Here the plots related to min_speed:

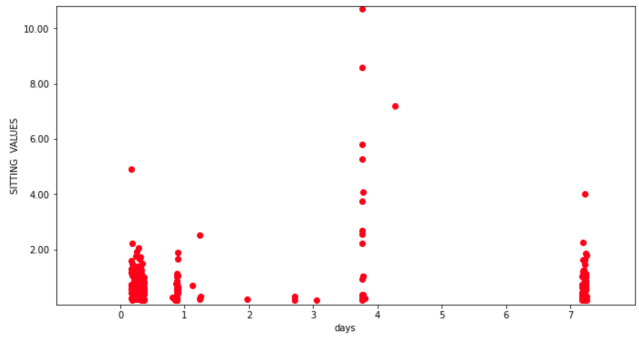

Figure 13: Sitting min speed distribution

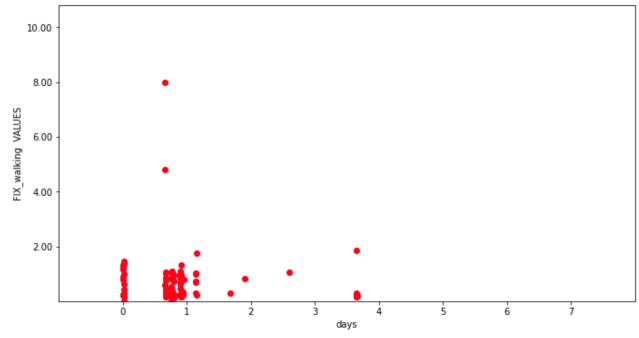

Figure 14: Walking min speed distribution

Regarding the *min speed* values, we can see almost the same values for both the labels, but, in the *sitting* activities these values are higher than the *walking* ones. The hypothesis is always attributes to the fact that probably in those moments the user was moving on a public or private transportation as a passenger. The data is also

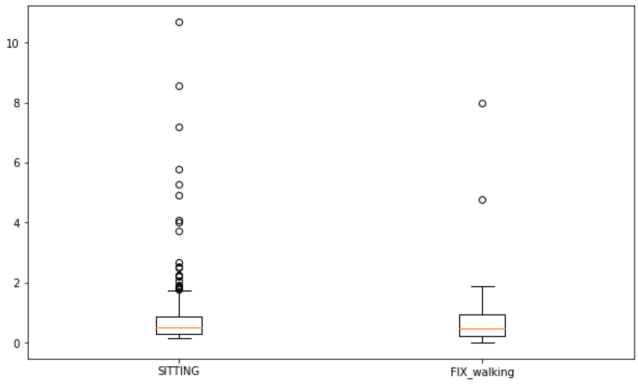

Figure 15: Sitting vs Walking min speed distribution

quite consistent with raw acceleration. What seems to be recurring, in this feature analysis and in the previous ones, is that after the 4th day, we have few data distribution or sometimes are totally absent with exception in certain case, when we can see some data spot, as shown in Figure (16 to 18)

Here the plots related to the *max speed*:

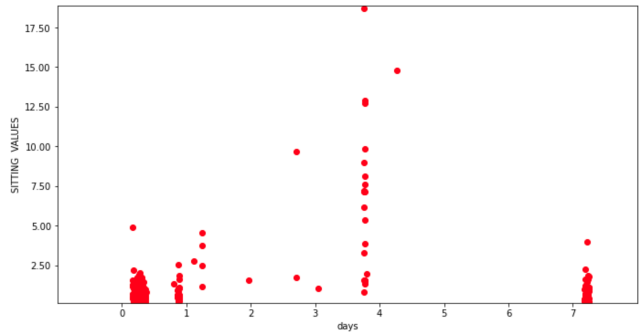

Figure 16: Sitting max speed distribution

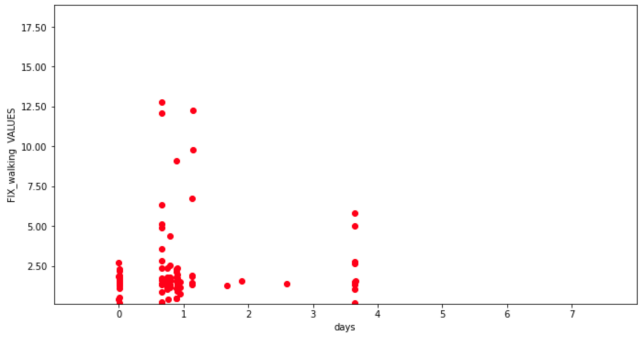

Figure 17: Walking max speed distribution

Here we have an expected behaviour: the values are more numerous for the *sitting* activities than the *walking* ones as always, but

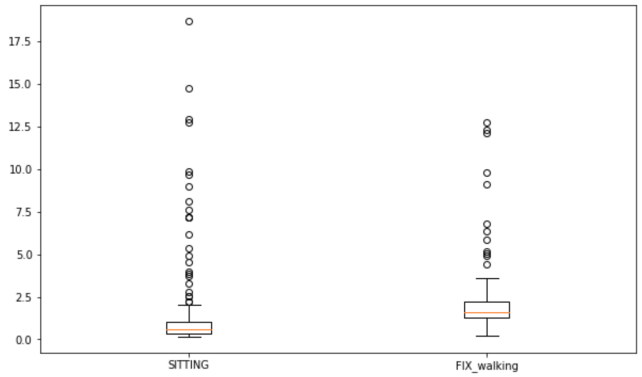

Figure 18: Sitting vs Walking max speed distribution

the boxplots are telling us that almost the majority of the walking values is concentrated around 2.0, while for the *sitting* ones we have clearly lower values. Also for *max speed* the behaviour is the same, the hypothesis that often the user forgot to activate the app or describe the activity becomes more plausible, another analysis is the comparison of the extracted audio values.

Considering the audio feature we compared the *maximum audio abs value*.

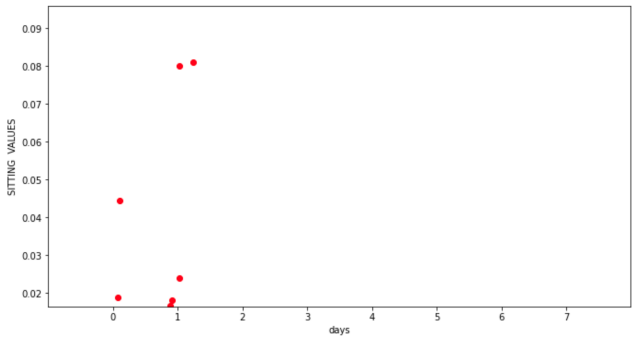

Figure 19: Sitting max audio abs distribution

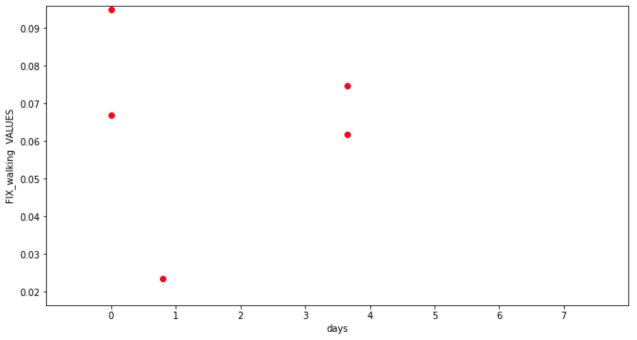

Figure 20: Walking max audio abs distribution

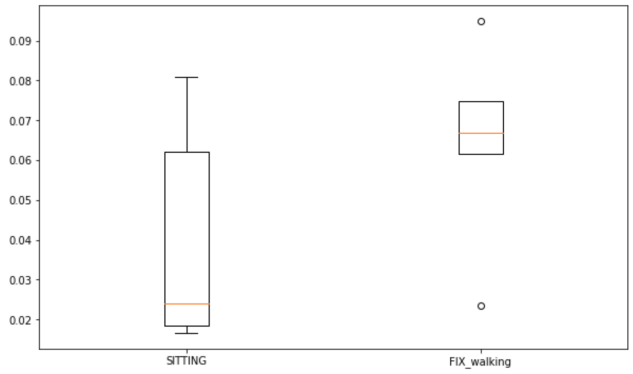

Figure 21: Sitting vs Walking max audio abs distribution

As for the Figures 19 to 21, In this case we have only a few data available: in order to have a meaningful analysis, we should analyze data of different users. Finally let's see what happens with the *battery indicator* of the user's phone: a feature that in our opinion can be important since it can be useful to identify a *walking* behaviour from a *sitting* one. This hypothesis comes from the assumption that when the user is sitting, he's surely not at home. At home it's reasonable that he just charged the phone, or he's charging it at the right moment that he's performing the labelling activity on the Extrasensory app. Our consideration is that the battery level during the *sitting* events can be higher than *walking* events. Below in Figure 22 to 24 the results:

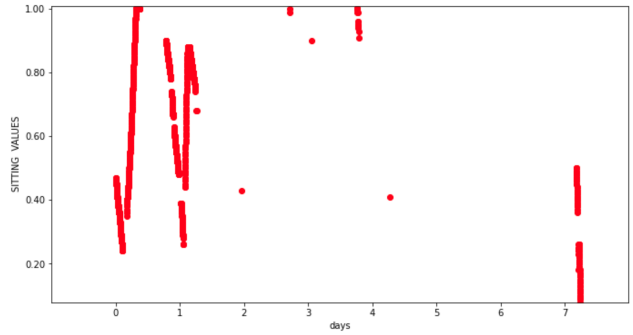

Figure 22: Sitting battery level distribution

There are, again, more values for the *sitting* activity than *walking* one. As expected, the boxplot of the *sitting* data is concentrated around values and generally higher than *walking* one.

4.2 All Users Study

To have a more precise view of the data, let's expand this analysis: we are going to take into consideration *all the users* for our study. To do this we define a method that, from this dataset, takes data related to a specific features and compare the behaviour of *all users* performing the transformation in order to have two most important labels that we encountered: *sitting* and *walking*. The difference with the previous method that we explain in the previous paragraph, is

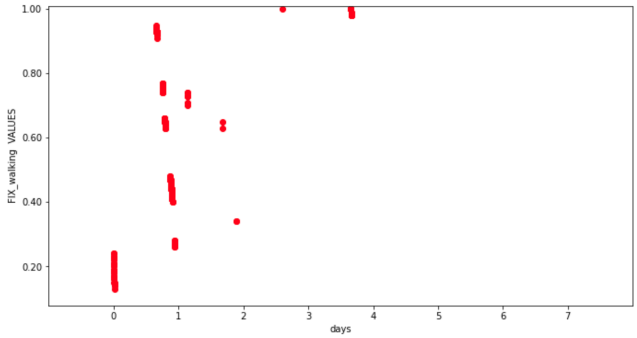

Figure 23: Walking battery level distribution

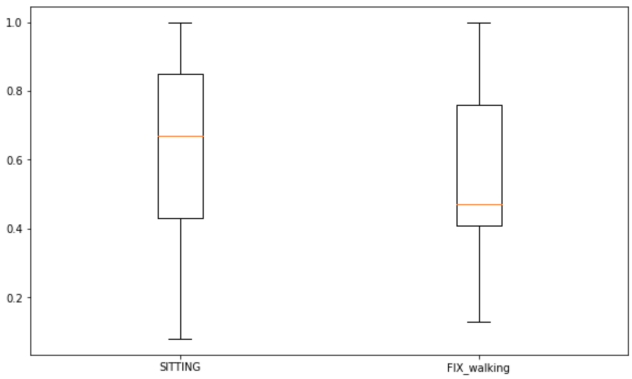

Figure 24: Sitting vs Walking battery level distribution

that in that case we compared data also with result of *single user* and analyze the feature's distribution. The next Figure (25 to 40) shown the result obtained by analysing the raw acceleration mean and the watch acceleration mean and convert the boxplot in histograms in order to have better visualization.

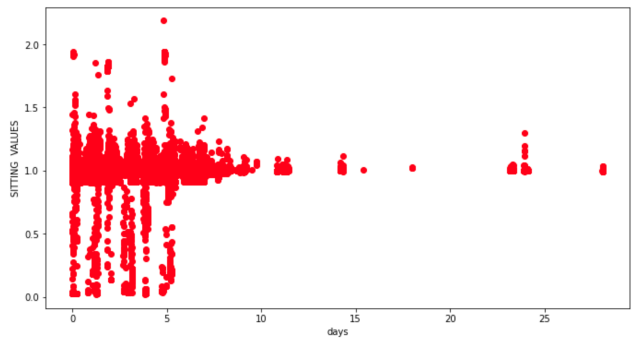

Figure 25: Sitting raw acceleration mean

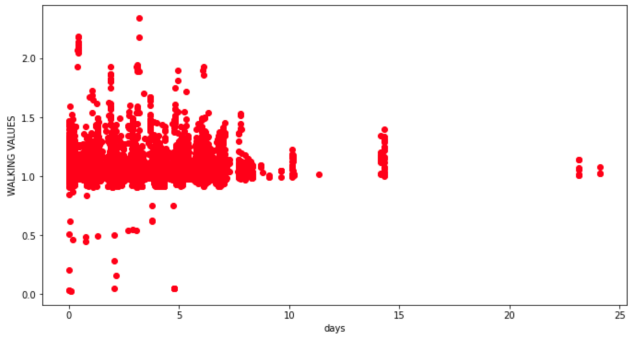

Figure 26: Walking raw acceleration mean

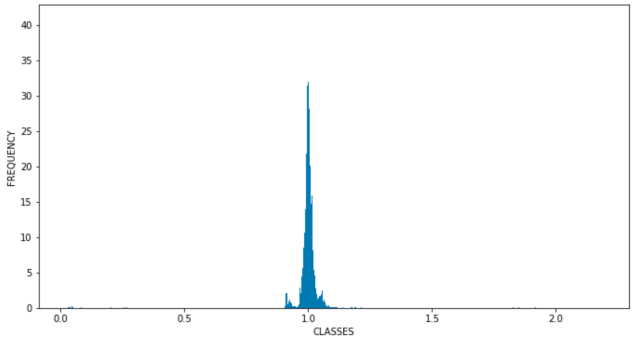

Figure 27: Sitting histogram raw acceleration mean

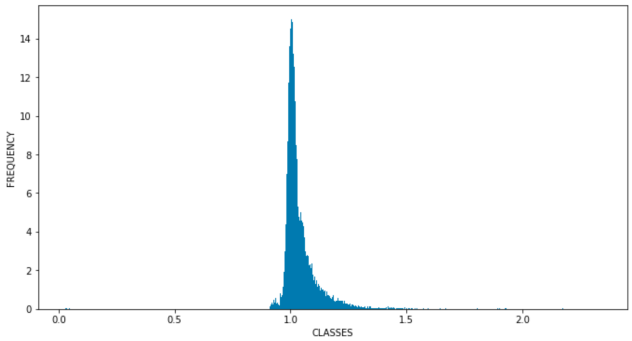

Figure 28: Walking histogram raw acceleration mean

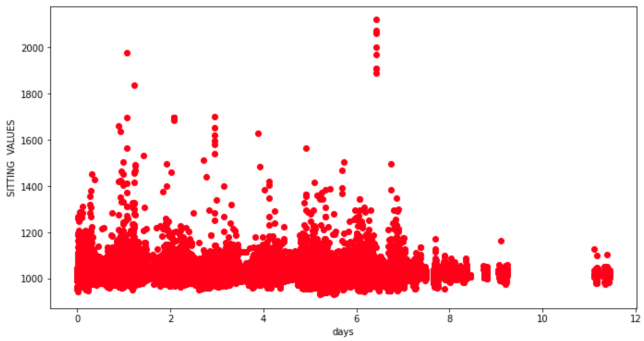

Figure 29: Sitting watch acceleration mean

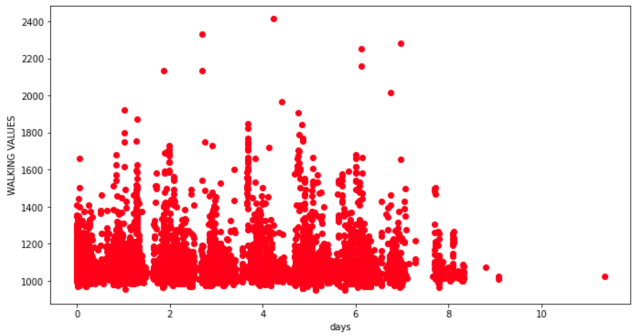

Figure 30: Walking watch acceleration mean

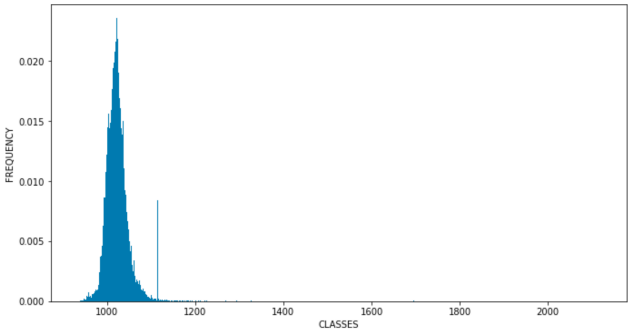

Figure 31: Sitting histogram watch acceleration mean

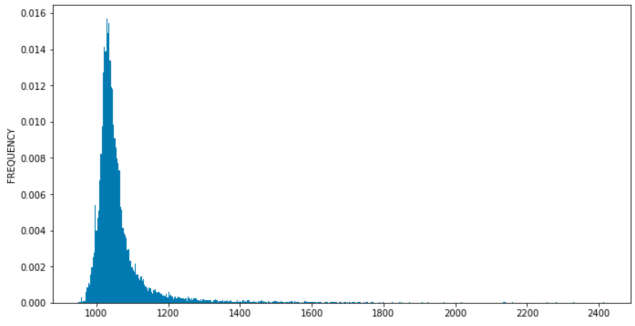

Figure 32: Walking histogram watch acceleration mean

After a comparation between the result from a *random single user* and the analysis of *all the user*, confirm the hypothesis that the data are generally more numerous values for the *sitting* activities but higher in value for the walking activities. Another consideration is related to the fact that the first few days users were more careful to monitor and record the various activities on the app (with an imbalanced values for *sitting* compared to *walking*), while in the last few days the data becomes less and less precisely, because not updated anymore and losing the consistency.

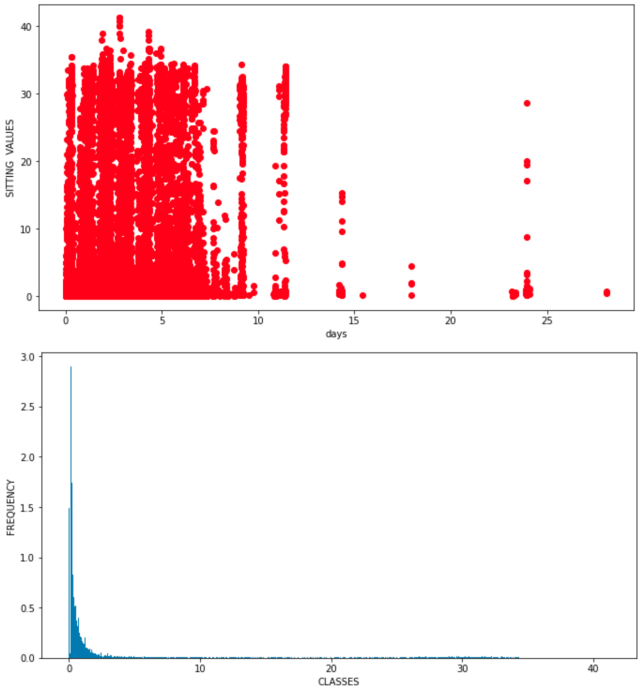

Figure 33: Sitting location: min_speed

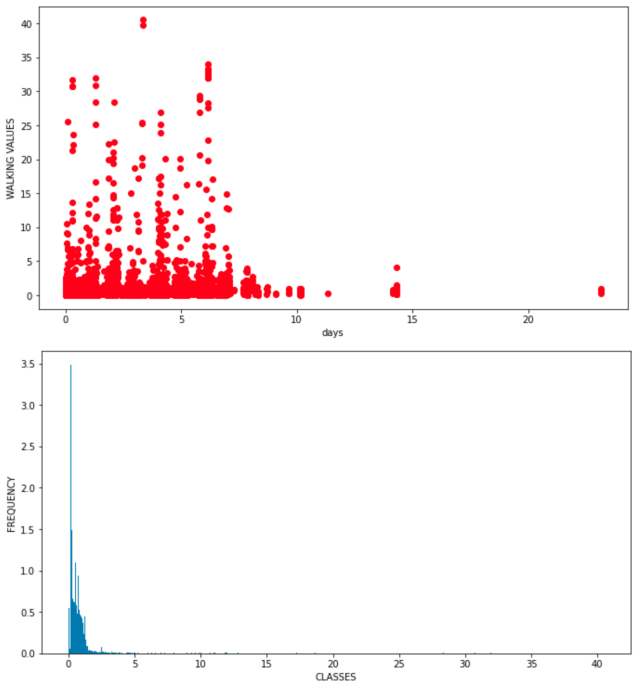

Figure 34: Walking location: min_speed

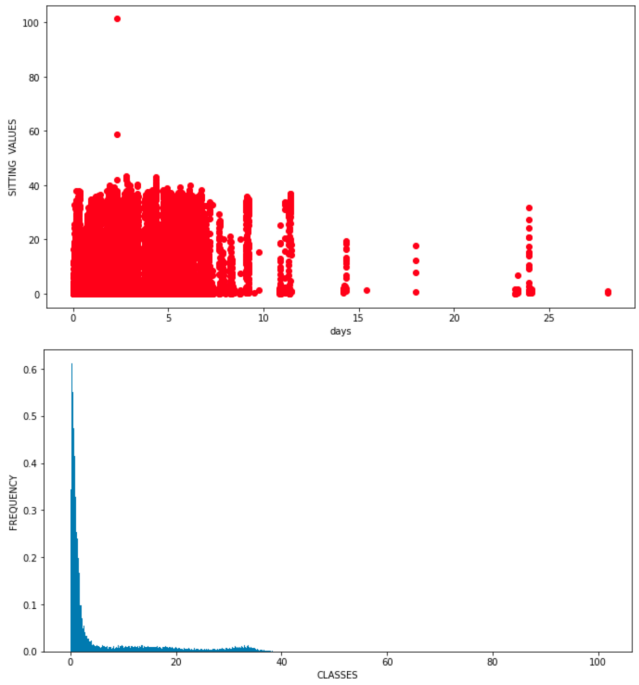

Figure 35: Sitting location: max_speed

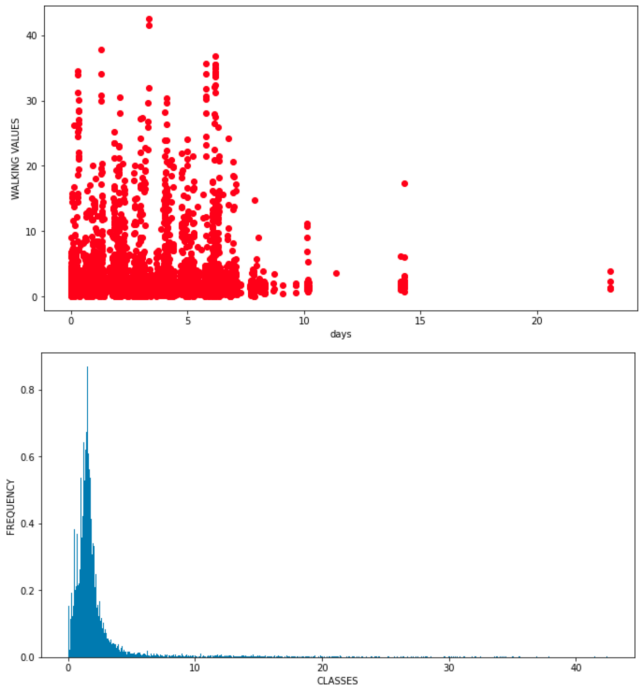

Figure 36: Walking location: max_speed

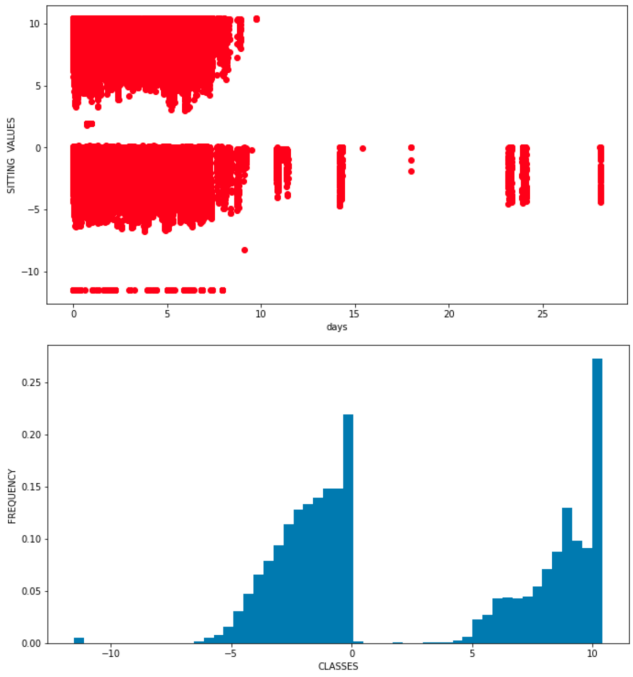

Figure 37: Sitting audio_properties: max_abs_value

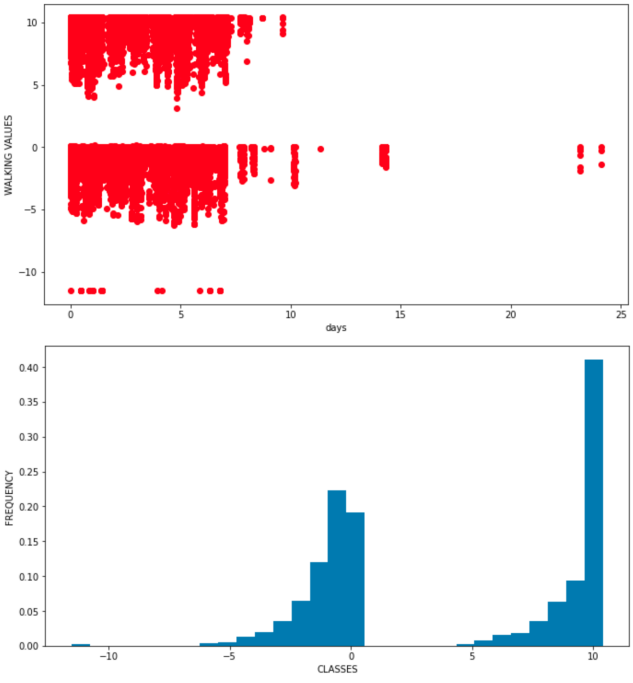

Figure 38: Walking audio_properties: max_abs_value

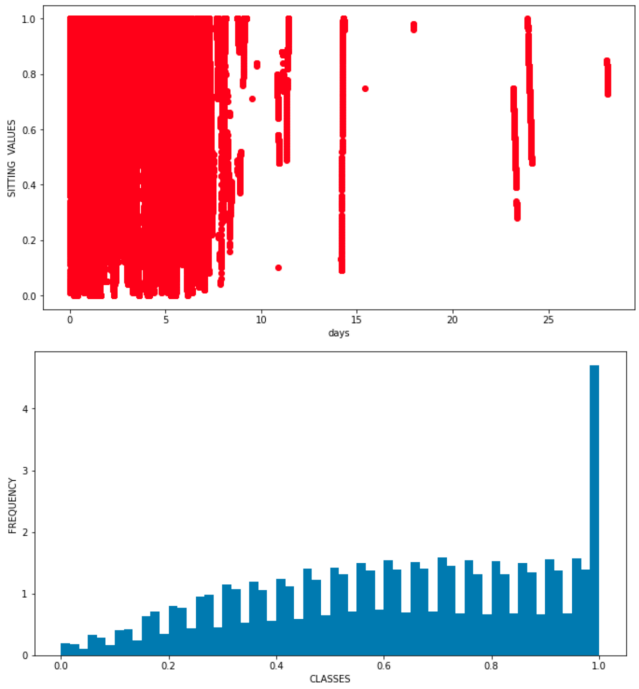

Figure 39: Sitting lf_measurements: battery_level

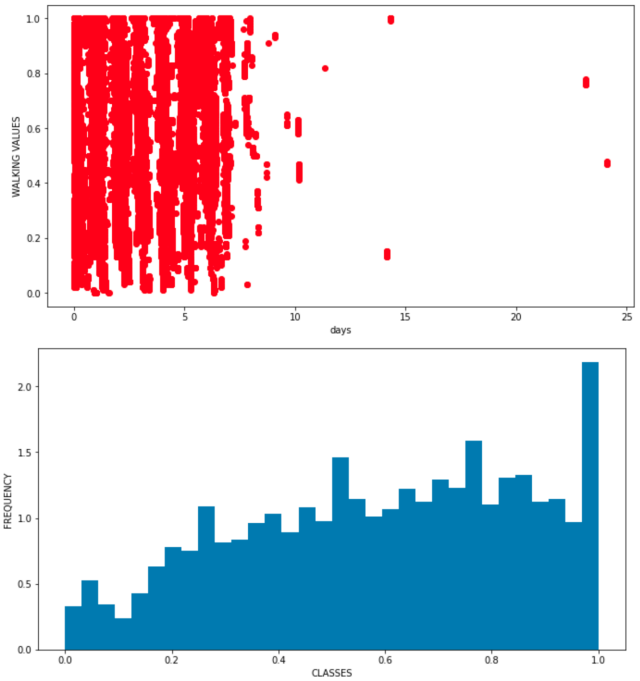

Figure 40: Walking lf_measurements: battery_level

5 DATA CLEANING AND PREPROCESSING

In our case, we found that many columns did not have values or in any case only some, so we decided to proceed with the removal of all the columns that have more than 90% of *NaN* values. From a further analysis however, a consistent presence of *NaN* values was found in the dataset, therefore a further removal of feature columns was done considering a percentage of the presence of *NaN* values greater than 50%. A further transformation of the dataset was the addition of the target column named 'CLASS' in order to have a binary classification which is identified as follows: 1 for static behaviours, 0 for movement behaviours. We compute this calculation by iterating over the rows of all the dataset: this is because the ExtraSensory dataset, as explained above, has different columns labeled with 'Sitting', 'Lying down' and all the other labels, with values of 1.0 if the user was performing that specific activity, 0 otherwise (for in-depth explanations go to the ExtraSensory dataset's link, in the reference section). Starting from this assumption, in order to populate our target column 'CLASS', we iterate over the rows of the dataset: if in the columns labeled with static labels there is at least an occurence of 1.0, we assign 1. Otherwise, there is at least an occurrence of 1.0 in the other group of columns labeled with movement labels: this leads us to assign 0. So at this point, we can proceed with the elimination of all the columns labeled with static labels, movement labels and also those not taken into account in this division, that were already present in the dataset: this is because our final dataset should contain only the features collected from the sensors and our target column 'CLASS'. The data cleaning is almost complete: we have chosen to consider only the features related to the 6 features analyzed in detail in the previous paragraph: this means that, if the raw_acceleration mean had been analyzed, all the features related to the latter such as the raw_acceleration standard deviation, related percentiles etc have been included in the dataset. Similar speech for watch_acceleration, location, audio and lf_measurements features. All the other features related to the proc_gyro, raw_magnet, the watch_heading, discrete: app_state and so on have been removed. Finally we decided to apply data imputation techniques, which consists in calculating a statistical value for each column (such as the mean value) and replacing all NaN values with this statistic: for this matter we use the *SimpleImputer* library of itshape Scikit Learn.

5.1 Data Balancing - SMOTE

Another important aspect considered is the distribution of the data example, most of the time it is possible to have imbalanced data examples across the classes, data examples introduced during data collection or errors made during data collection. Because the class distribution is not balanced, most machine learning algorithms will perform poorly and require modification to avoid misclassification error and accuracy. The most popular solution to avoid imbalanced classification problems is to change the composition of the training dataset, for our scope we use one of the most popular data sampling techniques called *SMOTE (Synthetic Minority Oversampling Technique). SMOTE* solves this problem by "oversampling" the examples in the minority class, randomly. The oversampling procedure is repeated enough times until the minority class has the same proportion as the majority class. Once this process is completed, the data is reconstructed and different classification models can be applied to the processed data.

6 COMPARISON CLASSIFICATION MODEL

From case studies applied in the context of extrasensory data, using the extra sensory app, we have seen that, as machine learning models, these 3 were used in particular: *KNN, Logistic Regression, Gaussian Naive Bayes.* All 3 are part of the Supervised Learning models. Once the dataset is prepared, we use the *Train_Test_Split* in order to use different training and testing dataset and to evaluate the performance of a machine learning algorithm. The first part is to train the algorithm, make a prediction and then in the second part evaluate the prediction against the expected results. The size of the split used is 70% for the training and 30% for testing, the reason is that the algorithm evaluation technique is fast and ideal for large datasets and also because it give as a strong evidence of the two splitted data to represent our problem. In addition to the specification of the size of the split, we also specify the random seed, in order to ensure that the results are reproducible, meaning that we ensure that trained and tested are always the same data. To find the best combination of parameters, we apply the *hyperparameter optimization* methodology called *GridSearch*, in order to build and evaluate the models for each combination specified in the grid. It's important to highlight that we evaluate the model twice: once with the unbalanced dataset, and once with the dataset balanced after the *SMOTE* operations, in order to check whether there are differences. We will also make another comparison: we used 3 models of batch learning, and also *MOA (Massive Online Analysis)* regarding online learning. Once we have tested each selected model, both of *Scikit and MOA*, we will make a comparison to verify which one gave results with greater accuracy. Below we will explain the reasons for choosing the individual models.

6.1 Brief introduction to MOA (MASSIVE ONLINE ANALYSIS)

MOA is an open-source framework designed specifically for data stream mining with concept drift. It is written in Java and developed at the University of Waikato, New Zealand. It allows to build and run experiments of machine learning or data mining on evolving data streams; it includes a set of learners and stream generators that can be used from the *Graphical User Interface (GUI)*, the command-line, and the Java API. First of all, both the MOA GUI and the Java API were used to perform three parallel classification tasks on the dataset; these results acted as a baseline for any future run. At this point, a first comparison among the three classifiers was made, in order to analyse their performances and costs. We then decided to repeat the same process in Python, to further elaborate on MOA's efficiency. Finally, a traditional batch approach was employed to additionally assess the potential gap with online learning

6.2 K-NEAREST NEIGHBOR (KNN)

K-Nearest Neighbor is a classifier that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in. K in KNN is a parameter that refers to the number of nearest neighbors to include in the majority of the voting process. In our case, it will take in

consideration similar features, voting for the *sitting* event or the Movement event. From similar cases of study, it outcome that KNN is not appropriate for the wild context, because there are too many dependent variables and the dataset could be too big, so the accuracy risks becoming bad. In our case study, we have different contexts, so we decided to take this model into consideration. After the standardization phase and consequent train / test of the model, thanks to GridSearchCV we passed several values for n_neighbors to the KNN classifier.

```
datas = {
    'name': 'KNN',
    'model': KNeighborsClassifier(),
    'params': {
    'n_neighbors': [7, 9, 11, 13, 15]
    }}
```

Figure 41: KNN hyperparameter

With the imbalanced dataset, the best results obtained are the following:

```
Best: 0.851045 using {'n_neighbors': 11}
accuracy:  0.8527612004843452
              precision    recall  f1-score   support

         0.0       0.70      0.49      0.58     19027
         1.0       0.88      0.95      0.91     73469

    accuracy                           0.85     92496
   macro avg       0.79      0.72      0.75     92496
weighted avg       0.84      0.85      0.84     92496
```

Figure 42: KNN Unbalanced Classification Accuracy

With the balanced dataset, here are the results:

```
accuracy:  0.8566889837326316
              precision    recall  f1-score   support

         0.0       0.81      0.94      0.87     73085
         1.0       0.93      0.78      0.84     73589

    accuracy                           0.86    146674
   macro avg       0.87      0.86      0.86    146674
weighted avg       0.87      0.86      0.86    146674
```

Figure 43: KNN Balanced Classification Accuracy

6.3 LOGISTIC REGRESSION (LR)

Logistic Regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". That's why this model fits well for our purposes. Also in this case, we use GridSearch in order to pass various parameters to the classifier.

```
datas = {
    'name': 'LC',
    'model': LogisticRegression(),
    'params': {
       'penalty': ['l2'],
       'C':  [1.0, 3.0, 6.0, 7.0, 10.0],
       'solver': ['sag', 'saga']
    }}
```

Figure 44: LR hyperparameter

An important role has been played by the *'C' parameter*: it is the inverse of the regularization strength, smaller values specify stronger regularization. It means that large values of C give more freedom to the model; conversely, smaller values of C constrain the model more. Regarding the choice of the solver, we found that for small datasets *'liblinear'* is a good choice, whereas *'sag' and 'saga'* are faster for large ones: that's why we only pass these 2 solvers to *GridSearchCV*. And this choice also implies the choice about the penalty: 'l2' is the most appropriate for these solvers.

With the *imbalanced* dataset, the best results are as follows:

```
Best: 0.826544 using {'C': 10.0, 'penalty': 'l2', 'solver': 'saga'}
accuracy:  0.8268465663380038
              precision    recall  f1-score   support

         0.0       0.73      0.25      0.37     19027
         1.0       0.83      0.98      0.90     73469

    accuracy                           0.83     92496
   macro avg       0.78      0.61      0.64     92496
weighted avg       0.81      0.83      0.79     92496
```

Figure 45: LR Unbalanced Classification Accuracy

With the *balanced* dataset, here are the results:

```
              precision    recall  f1-score   support

         0.0       0.76      0.65      0.70     73085
         1.0       0.70      0.80      0.74     73589

    accuracy                           0.72    146674
   macro avg       0.73      0.72      0.72    146674
weighted avg       0.73      0.72      0.72    146674
```

Figure 46: LR Balanced Classification Accuracy

6.4 Naive Bayes (MOA vs Sklearn)

Finally, *Naïve Bayes* is a classification technique based on Bayes theorem, it is part of a family of statistical classifiers used in machine learning. The *Naive Bayes* algorithm is a probabilistic and supervised learning algorithm suitable for solving binary and multi-class classification problems. The main peculiarity of the algorithm, in addition to making use of the Bayes theorem, is that it is based on the fact that all the characteristics are not related to each other. The presence or absence of one feature does not affect the presence or absence of others. Calculate the probability of each label for a given object by looking at its characteristics. Then, he chooses the label with the highest probability. When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. That's why we choose *Gaussian Naive Bayes*.

```
datas = {
    'name': 'GaussianNB',
    'model': GaussianNB(),
    'params': {
              'var_smoothing' : np.logspace(0,-9, num=100)
    }}
```

Figure 47: GNB hyperparameter

In the image above we can see that different values have been passed as var_smoothing, that is the portion of the largest variance of all features that is added to variances for calculation stability.

With the *imbalanced* dataset, the best results are as follows:

```
Best: 0.814089 using {'var_smoothing': 1.519911082952933e-07}
accuracy:  0.8103269330565646
              precision    recall  f1-score   support

         0.0       0.54      0.48      0.51     19027
         1.0       0.87      0.90      0.88     73469

    accuracy                           0.81     92496
   macro avg       0.71      0.69      0.70     92496
weighted avg       0.80      0.81      0.81     92496
```

Figure 48: GNB Unbalanced Classification Accuracy

With the *balanced* dataset, here the results:

```
Best: 0.697944 using {'var_smoothing': 2.310129700083158e-07}
accuracy:  0.6978741971992309
              precision    recall  f1-score   support

         0.0       0.82      0.51      0.63     73085
         1.0       0.64      0.89      0.75     73589

    accuracy                           0.70    146674
   macro avg       0.73      0.70      0.69    146674
weighted avg       0.73      0.70      0.69    146674
```

Figure 49: GNB Balanced Classification Accuracy

6.5 Naive Bayes - MOA

As we stated before, we chose to address the problem also by adopting online machine learning models, so we fed *MOA* with our dataset, by setting the parameters as follow:

- `instance limif`: 100.000.000
- `timeLimit`: -1
- `sampleFrequency`: 100.000
- `memCheckFrequency`: 100.000

Let's start our consideration about the results obtained with the dataset imbalanced.

We can see, with the table below, that over the time the learner improves its performance, obtaining increasingly correct predictions, up to an accuracy of 94.6%.

classifications correct (percent)	Kappa Statistic (percent)
73.8	73.8
90.3	90.3
88.6	88.6
94.6	94.6

Figure 50: GNB - MOA Unbalanced prediction accuracy

Here with balanced data: we can see that we have an opposite behaviour. It starts with good results in predicting the instances correctly but overtime, probably due to the presence of those values added after the mean imputation that 'wrongly' populates the dataset, leading the classifier to wrongly learn and predict instances (from 73.8% to 47,6% of correct classifications)

classifications correct (percent)	Kappa Statistic (percent)
73.8	73.8
90.3	90.3
88.6	88.6
47.3	47.3
47.599999999999994	47.599999999999994

Figure 51: GNB - MOA Balanced prediction accuracy

7 CONCLUSION

As described in the previous paragraphs, we carried out several tests with different transformations of the dataset and models, below is a summary of the cases:

- 1: unbalanced dataset transformed with the mean imputation technique.
- 2: balanced dataset transformed with the mean imputation technique.

In the first case, the study carried out with the Scikit classifiers and the one carried out with the MOA classifier provided rather satisfactory results, as the accuracy value was higher than 80%. In the second case, an interesting phenomenon occurred when we balanced the dataset we obtained a worse results than the previous ones. In fact, both with MOA and with Scikit-learn classifiers the

accuracy value has changed, getting worse with a resulting accuracy of 47%.

In summary:

- The accuracy of the KNN has improved, going from 85% to 86%
- The accuracy of the LR has getting worse, going from 83% to 72%
- GNB accuracy has getting worse, going from 81% to 70%
- GNB accuracy with MOA has getting worse, going from 94% to 47%

We came to the conclusion that the worsening of the accuracy was caused by the dataset transformation through the data imputation, in particular by replacement the average for the NaN values and promote the propagation of the error during the computation of the classifiers, excepting for the KNN classifier that improve the accuracy because of the replacement of the NaN values lead it to better learn the characteristics of the instances and classify them through the k-neighbour. With an unbalanced dataset, MOA provided the best results with 94% accuracy, while batch learning model reach 85% of accuracy. Therefore, the batch learning classifiers performed decrease their performance with a modified dataset through mean imputation, however we still have a reasonable accuracy.

MOA greatly get worse its performance and this phenomenon could be attributed to a better prediction of online learning models with raw and untreated datasets.

Some considerations. The Extrasensory dataset presents a satisfactory amount of data, allowing us to achieve all our objectives set for our study, with satisfactory results in the identification of static activities or movements. This work could be a useful resource for future studies, expanded and used for other contexts. Eventually it can be exploited to further study the lifestyle of individuals, proposing new solutions such as the creation of control and monitoring applications integrated with IoT and embedded solutions. Certainly a satisfying goal, but in order to draw more accurate conclusions, we think that we need a study done on a greater number of individuals, with different habits and different age groups.

SMOTE	CLASSIFIER	ACCURACY	PRECISION	RECALL	F1-SCORE	CLASS
Before	KNN	85%	70%	49%	58%	0
			88%	45%	91%	1
After		86%	81%	94%	87%	0
			93%	78%	84%	1
Before	LR	83%	73%	25%	37%	0
			83%	98%	90%	1
After		72%	76%	65%	70%	0
			70%	80%	74%	1
Before	GNB	81%	54%	48%	51%	0
			87%	90%	88%	1
After		70%	82%	51%	63%	0
			64%	89%	75%	1

Figure 52: Summary accuracy

SMOTE	CLASSIFIER	ACCURACY
Before	GNB with MOA	94,6%
After		47,6%

Figure 53: Summary accuracy

REFERENCES

[1] Manhyung Han, La The Vinh, Young-Koo Lee, and Sungyoung Lee. 09-2012. Comprehensive Context Recognizer Based on Multimodal Sensors in a Smartphone. 12588-12605 (09-2012).

[2] Jani Mantyjarvi, Johan Himberg, and Tapio Seppanen. 2001. Recognizing human motion with multiple acceleration sensors. In Systems, Man, and Cybernetics. Vol.2 (2001).

[3] Hamed Pirsiavash and Deva Ramanan. 06-2012. Detecting Activities of Daily Living in First-person Camera Views. 12884640 (06-2012).

[4] Valentin Radu, Catherine Tong, Sourav Bhattacharya, Nicholas D. Lane, Cecilia Mascolo, Mahesh K. Marina, and Fahim Kawsar. 01-2018. Multimodal Deep Learning for Activity and Context Recognition. Vol.1, Issue 4 (01-2018), 1 – 27.

[5] Christie Napa Scollon, Chu-Kim Prieto, and Ed Diener. 2009. Experience sampling: promises and pitfalls, strength and weaknesses. (2009).

[6] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. 04-2018. ExtraSensory App: Data Collection In-the-Wild with Rich User Interface to Self-Report Behavior. 554 (04-2018), 1 – 12.

A ONLINE RESOURCES

- `The ExtraSensory Dataset`: http://extrasensory.ucsd.edu
- `Extrasensory Dataset`: http://extrasensory.ucsd.edu/data/primary_data_files/ExtraSensory.per_uuid_features_labels.zip
- `Extrasensory Tutorial`: http://extrasensory.ucsd.edu/intro2extrasensory/intro2extrasensory.html