Stress Detection based on Behavioral Context Recognition

Roberto Calabrese
Department of Computer Science
University of Salerno
Italy
r.calabrese21@studenti.unisa.it

Antonio De Luca
Department of Computer Science
University of Salerno
Italy
a.deluca72@studenti.unisa.it

INTRODUCTION

Nowadays our smartphones are an integral part of our daily routine. We bring them everywhere with us and they constantly collect data about our surrounding environment through sensors. Behavioral context recognition is about detecting the environment in which the user is located and the actions that the user is exercising. In this study we took into consideration a dataset which contains data from several different users about their human context. The users have been observed for a limited period during which they have installed an app on their smartphones (iPhone/Android) and smartwatches which has listed their actions during their daily activities. These samples were inserted in a dataset for each user.

The Internet of things (IoT) describes all the physical objects that are embedded with sensors, processing activities, software, and other technologies, and that connect and exchange data with other devices and systems over the internet or other communication networks. During our study the Internet of things was represented by the 60 users' smartphones and smartwatches that collected data. Behavioral context recognition can be quite tricky, together with the Internet of things it can be a real challenge. It is about detecting the actions that a user is doing normally during their day and the places that they are in at that moment. This can be hard with an Internet of Things scenario because you cannot know with certainty if the smartphone or smartwatch is correct about a certain action or location, or if the user is influenced by some factors so they cannot perform their actions as they normally do.

The goal of this study is to identify the features that better predict a user who appears to be in a relaxed state from another one who appears to be in a stressed state. This can be useful in future domain research to detect users' lifestyle, looking for stressed people to propose some actions and behavior to prevent it and to improve their health.

The classification has been performed with two different models of batch learning (Logistic Regression and Gaussian Naive Bayes) and MOA in relation to the online learning. Further ahead, we will examine the state of the art of the problem, then we will inspect the dataset in more detail, as well as our approach for the prediction, justifying all our decisions and our intentions.

STATE OF THE ART

The heartbeat of a person can be very important to predict whether that person is physically healthy or not, knowing that person's age. Based on the heartbeat it is also possible to predict whether a person is in stress or not. According to the American Institute of Stress[1]:

- About 33 percent of people report feeling extreme stress
- 77 percent of people experience stress that affects their physical health
- 73 percent of people have stress that impacts their mental health
- 48 percent of people have trouble sleeping because of stress

These percentages let us understand that stress is a serious problem that affects most of the global population. A research study conducted by Pandey [1] used the heartbeat rate as one of the parameters to detect the stress beforehand. This study was done to inform the person about their unhealthy lifestyle and alarm them before worse health conditions happen. They used Internet of things (IoT) together with Machine Learning (ML) to alarm the person when they are at risk. In our study, the heartbeat rate of the users wasn't available, so we used other parameters.

Work should be a source of wellbeing, in the sense of enhancing personal development. Instead, it is usually one of the main causes of stress. It is obvious that healthy workers perform better, but when work becomes the cause of stress, it may lead

[1] https://www.therecoveryvillage.com/mental-health/stress/related/stress-statistics/

to serious illnesses. Another study by Muaremi et al. [2] leveraged the features derived from smartphones and other devices to assess the stress experience of people. Some of the features that they used were from audio and the battery level of the device, which we used as well.

During a study by Bauer et al. [3], they managed to describe initial results from an ongoing project to use mobile phone sensors to detect stress related situations. The questions that they address is whether differences between stressful and non-stressful periods can be detected in information readily available on a smartphone such as location traces, Bluetooth devices seen during the day and phone call patterns.

Hong Lu et al. [4] leveraged the power of smartphone microphones. Changes in the speech production process is one of many physiological changes that happen during stress. Microphones, embedded in mobile phones and carried ubiquitously by people, provide the opportunity to continuously and non-invasively monitor stress in real-life situations.

Kostopoulos et al. [5] collected data from people's daily phone usage gathering information about the sleeping pattern, the social interaction, and the physical activity of the user to detect stress.

Wang et al. [6] leveraged the power of smartphones, embedded with a rich set of sensors, which can capture people's context, such as movement, sound, location and so on, to access people's behavior.

We can notice that there was a lot of studies on behavioral context recognition together with stress detection. Our focus is to use machine learning classification models on two different mind states: *relaxed* and *stressed*, to understand the levels of stress of ordinary people and try to improve their lives.

3 DATA DESCRIPTION AND FIRST ANALYSIS

For our study we used the ExtraSensory dataset[2], which contains data from 60 users, each identified with a universally unique identifier (UUID). For every user there are thousands of examples, typically taken in intervals of 1 minute. Every example contains measurements from sensors. The sensors data was taken from the user's personal smartphone and from a smartwatch provided by the researchers (even though only 56 out of the 60 users agreed to wear the smartwatch). Most examples also have context labels self-reported by the user with the ExtraSensory application for collecting data in-the-wild, describing people's behavioral context, for example: lying down, computer work, running, watching TV, etc. Every minute the application records a 20 second window of sensor measurements from the smartphone and smartwatch (if available), collecting data from the accelerometer, magnetometer, location, low-frequency measurements, and phone-state sensors, as well as accelerometer and compass from the smartwatch. The data collection performed on real life scenarios was accompanied by a 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 was loaded online on a Google Drive folder to preprocess the data so that it is more suitable for modelling. The use of this method helps to parse the CSV files from the dataset, to obtain all the data from one user or from every user. As said before, we selected a subset of labels and divide them in two different groups, *relaxed and stressed*, as shown in Figure 1:

```
relaxed_labels = ['Lying down', 'Sleeping', 'With friends', 'Bicycling',
                  'Exercise', 'Grooming', 'Bathing - shower', 'Shopping',
                  'Running', 'Outside', 'At a restaurant', 'Strolling',
                  'At a party', 'At the beach', 'At the gym']

stressed_labels = ['Computer work', 'Lab work', 'In a meeting', 'In class',
                   'At main workplace', 'On a bus', 'Drive - I\'m the driver',
                   'Watching TV', 'Surfing the internet']
```

Figure 1: Labels subsets

Labels were chosen and split accordingly to the correlation to the main classes such as for *relaxed* class have been selected labels like *Bicycling, Exercising, Sleeping* etc. For *stressed* class have been selected labels like *In a meeting, Lab work, Driving a bus* etc.

The first part of the study was the collection of a single user's data chosen randomly, calculating how much time he spent performing each activity in the two groups of labels, as shown in Figure 2:

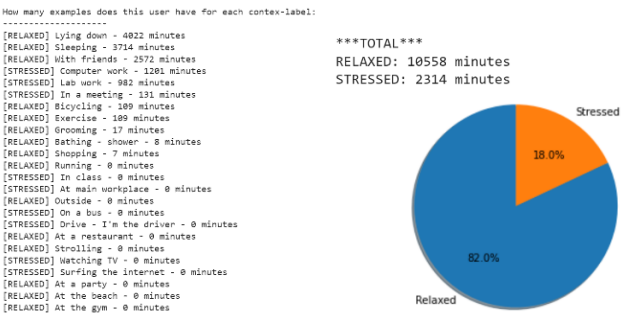

Figure 2: *relaxed* and *stressed* pie chart for a single user

[2] Extrasensory Dataset: http://extrasensory.ucsd.edu/

From the pie chart we can clearly see that this user spends more time in a relaxed state than in a stressed one. We developed this first analysis to all the 60 users. The result was the same: from the total data, we can see that all the users spend more time in a relaxed state than in a stressed one, as shown in Figure 3.

```
***TOTAL***
RELAXED: 251409 minutes
STRESSED: 129632 minutes
```

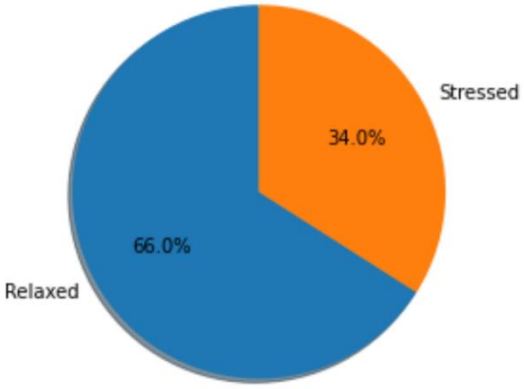

Figure 3: *relaxed* and *stressed* pie chart for all users

Our goal is to detect and differentiate the *relaxed* states from the *stressed* ones, for this reason we calculated the overall time spent doing each *relaxed* activity and each *stressed* activity.

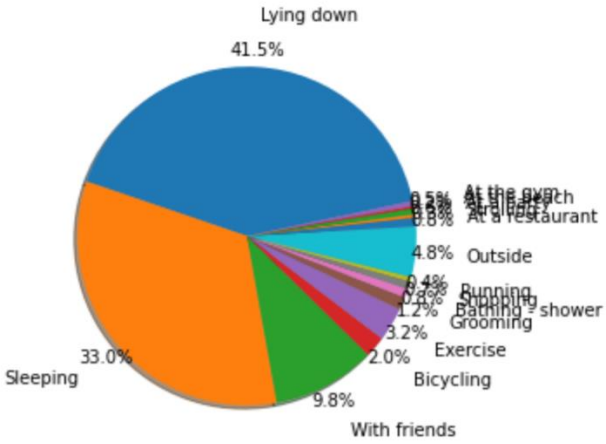

Figure 4: *relaxed* labels chart exploration

In the Figure 4, we immediately see that among the *relaxed* labels, *Lying down* and *Sleeping* have a very high percentage, followed by *With friends* and all the others that have a smaller percentage.

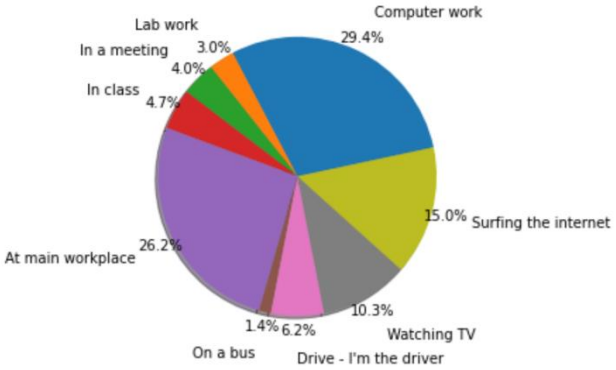

Figure 5: *stressed* labels chart exploration

In the Figure 5, the *stressed* labels, *Computer work* has a higher percentage with respect to the others, followed by *At main workplace* and *Surfing the internet*. Ultimately, we can say that in the overall dataset, data referring to *relaxed* behaviors is prevalent with respect to the *stressed* ones. At this stage of the study, we need to make additional analysis, this means that we have to find a way to detect the best fitting features that surely differentiate a *relaxed* behavior from a *stressed* one. According the research papers that we found, there are a set of features that influence this aspect.

4 FURTHER ANALYSIS AND FEATURES SELECTION

4.1 Single user study

The ExtraSensory dataset has features provided by ten types of sensors. Phone-state sensors, unlike the other sensors, have discrete categories such as battery state and WiFi availability. The other sensors have many features extracted from the same source of time-sequential raw data: for example, related to the acceleration, the dataset provides the mean, the standard deviation, the percentiles and so on. The features are harder to interpret than the labels. They were determined from the raw measurements from the several sensors on the smartphone or smartwatch. Let's start looking at the feature names, we can notice that the prefix of each feature suggests the sensor where it comes from. To simplify the features names, we used a function to parse them and provide the code-name of the sensor they belong to. We used the code-names as they are called 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.

```
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
18) Acc         raw_acc:3d:mean_y
19) Acc         raw_acc:3d:mean_z
20) Acc         raw_acc:3d:std_x
21) Acc         raw_acc:3d:std_y
22) Acc         raw_acc:3d:std_z
23) Acc         raw_acc:3d:ro_xy
24) Acc         raw_acc:3d:ro_xz
25) Acc         raw_acc:3d:ro_yz
26) Gyro        proc_gyro:magnitude_stats:mean
27) Gyro        proc_gyro:magnitude_stats:std
28) Gyro        proc_gyro:magnitude_stats:moment3
29) Gyro        proc_gyro:magnitude_stats:moment4
30) Gyro        proc_gyro:magnitude_stats:percentile25
31) Gyro        proc_gyro:magnitude_stats:percentile50
32) Gyro        proc_gyro:magnitude_stats:percentile75
33) Gyro        proc_gyro:magnitude_stats:value_entropy
34) Gyro        proc_gyro:magnitude_stats:time_entropy
```

Figure 6: Some of the ExtraSensory features

Now, let's consider some features that we consider relevant for our study because they can be useful to detect different behaviors. We will show they vary overtime, relatively a definite type of activity. We create a function that takes two labels and the feature's index that we want to process (the feature index is taken from the list of features shown above).

We took into consideration the following features, according to the researches described in the state of art:

1. location latitude range
2. location longitude range
3. audio properties: maximum absolute value
4. low-frequency measurements: battery level
5. low-frequency measurements: screen brightness

We chose the two most frequent labels for each group, that are *Lying down* for *relaxed* and *Computer work* for *stressed*. We compared the data labeled as *Lying down* and the data labeled as *Computer work* for a single user, considering the distribution value of the location latitude range.

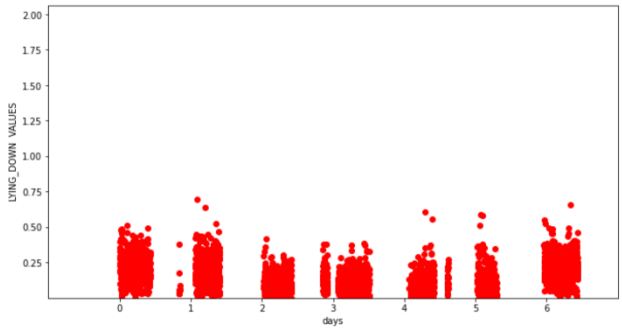

Figure 7: Lying down latitude distribution

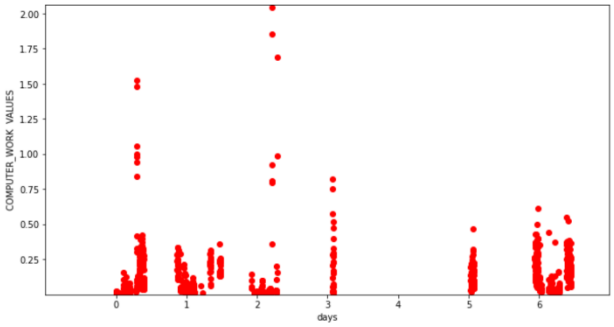

Figure 8: Computer work latitude distribution

From the two graphs (Figure 7 and 8) we can say that the values are more constant in *Lying down* than *Computer work*. The values appear more compact in *Lying down*, maybe this could be caused by the fact that that user usually lies down at the same place every day (for example their home). Instead, for *Computer work* we notice that the values are more scattered, perhaps this user does computer work at different places.

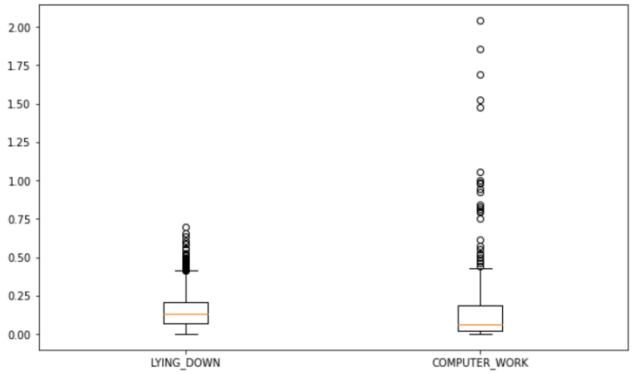

Figure 9: Lying down vs Computer work latitude comparison

We can see that in the Figure 9 the values are different between *Lying down* and *Computer work*. From the boxplot of the *Lying down* distribution the values are concentrated at the median, at 50% percentile. We also notice the presence of outliers above the concentration. Perhaps these outliers represent events like exercising in a lying down position, explaining the behavior of these values. For *Computer work* distribution the values are concentrated in the range of 25% and 75% of percentile; the outliers are more scattered with respect to the *Lying down* ones. At this point, let's compare the values of the location longitude range between the two labels.

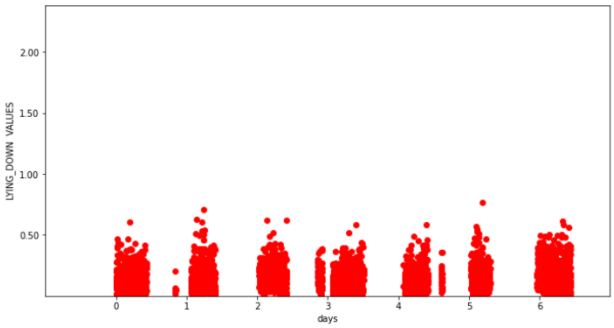

Figure 10: Lying down longitude distribution

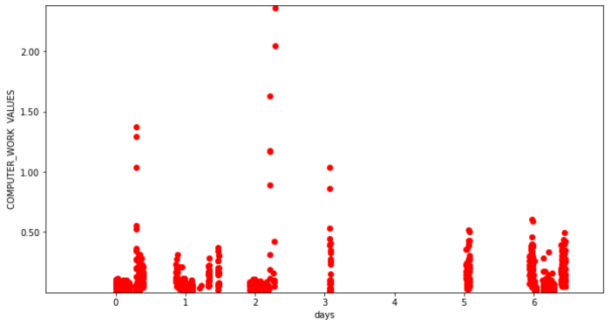

Figure 11: Computer work longitude distribution

As before, in these new graphs (Figure 10, 11), the values of *Lying down* label are more than the *Computer work* values, which are more scattered. Same as before, this could be caused by the fact that the user usually lies down at home and does computer work at different places.

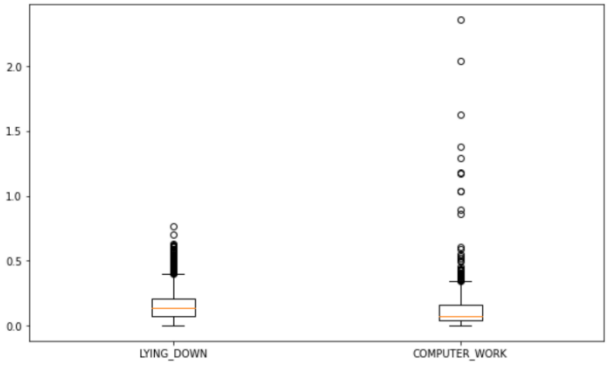

Figure 12: Lying down and Computer work longitude comparison

The boxplot (Figure 12) is also similar to the latitude one. This is probably because latitude and longitude are two features highly connected between them. The analysis continues considering another feature, which is the audio property called maximum absolute audio value. In this case we didn't obtain any data for this user: to have a meaningful analysis, we will later show data analyzed from different users.

Now let's see what happens with the battery indicator of the user's smartphone: a feature that in our opinion can be important since it can be useful to identify a *Lying down* behavior from a *Computer work* one. We can justify this with the fact that when the user is at the computer, they are probably not using their phone. At home they probably just charged their phone, or they are charging it while they are adding the labeling activity on the Extrasensory app. The battery level during *Lying down* events can be higher with respect to the battery level during *Computer work* events.

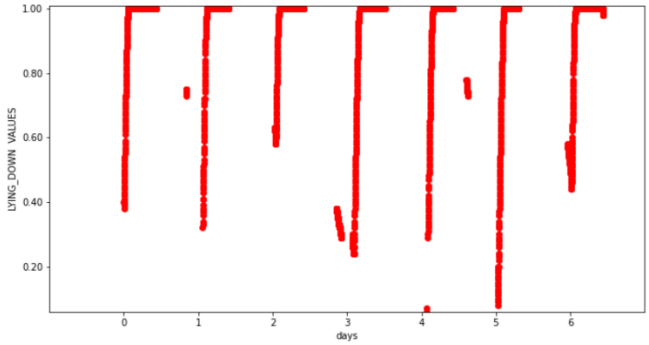

Figure 13: Lying down batter level distribution

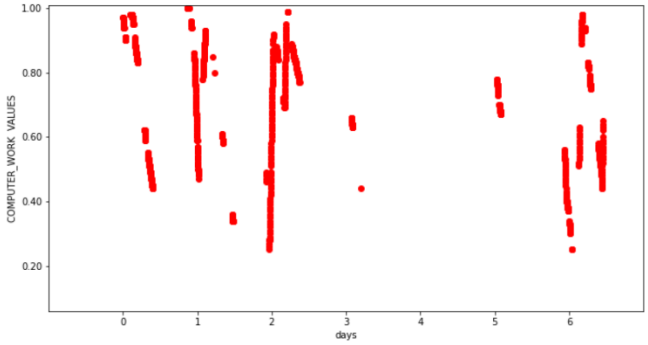

Figure 14: Computer work battery level distribution

As we can see (Figure 13, 14) there are more values for *Lying down* instead of *Computer work*. The *Lying down* values seem to appear always in the same positions, probably because this user is charging their smartphone at night, so the battery level values are constant throughout the day. Instead, the *Computer work* values are more scattered, perhaps because their smartphone is used more.

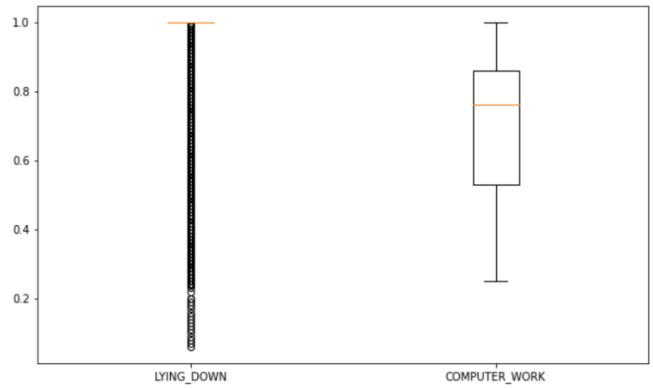

Figure 15: Lying down and Computer work battery level comparison

The boxplots (Figure 15) appear to have a lot of outliers for the *Lying down* events. This is probably because the battery level is a feature that continuously changes over time.

Finally, let's see what happens with the brightness of the smartphone.

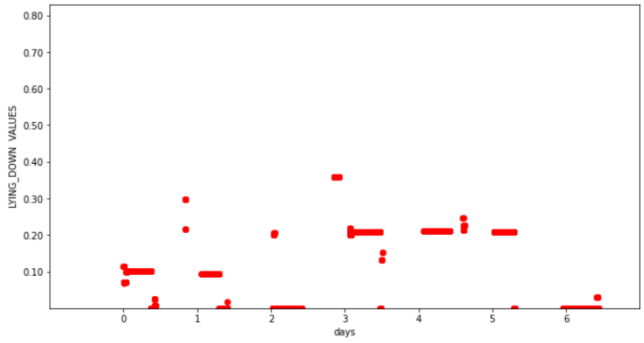

Figure 16: Lying down brightness distribution

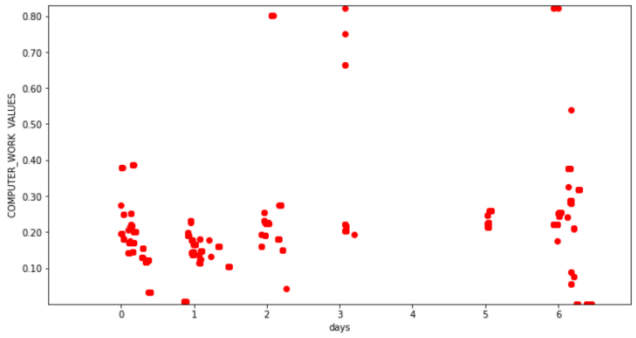

Figure 17: Computer work brightness distribution

As always, as we can see in the graphs (Figure 16, 17), the *Lying down* values are more concentrated with respect to the *Computer work* ones. This is probably because this user lowers the brightness to relax, in fact a brightness that is too high doesn't let a person relax properly.

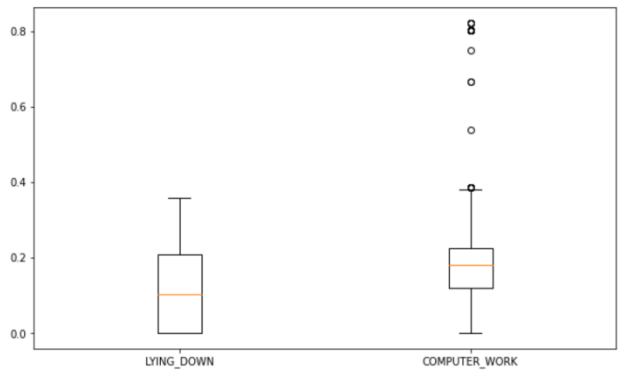

Figure 18: Lying down and Computer work brightness comparison

As we can see in the boxplots (Figure 18) the values appear more compact for *Lying down* instead of *Computer work*.

4.2 All Users Study

At this point, we are going to take into consideration all the users to achieve a more accurate visualization of data. We defined a function that extracts data related to a specific feature and compare all the behavior of all users performing the transformation to have two most important labels that we encountered: *Lying down* and *Computer work*. The next Figures (19 to 26) show the result acquired by analyzing the latitude and longitude range of all users using histograms instead of boxplots in order to have better visualization.

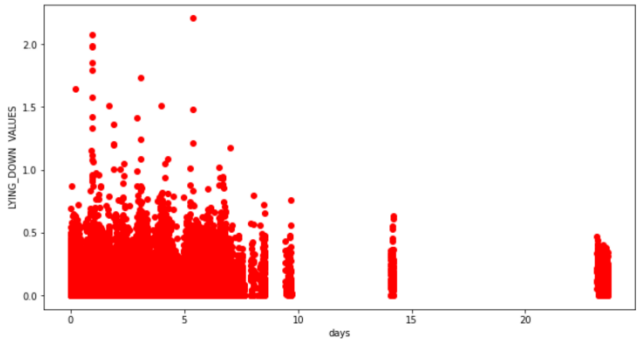

Figure 19: Lying down latitude distribution (all users)

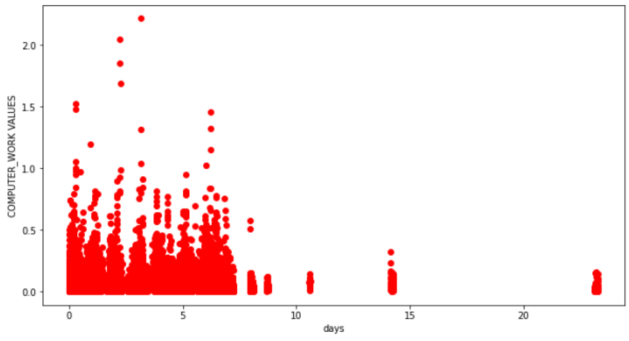

Figure 20: Computer work latitude distribution (all users)

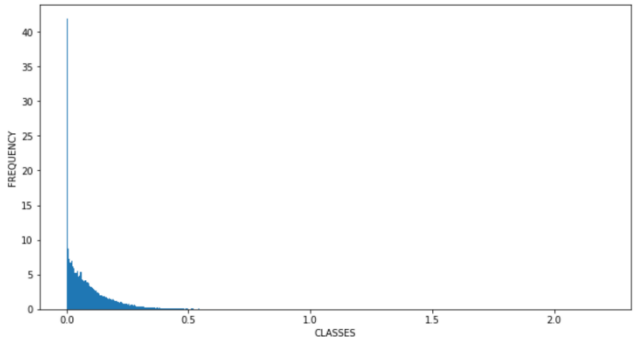

Figure 21: Lying down latitude histogram (all users)

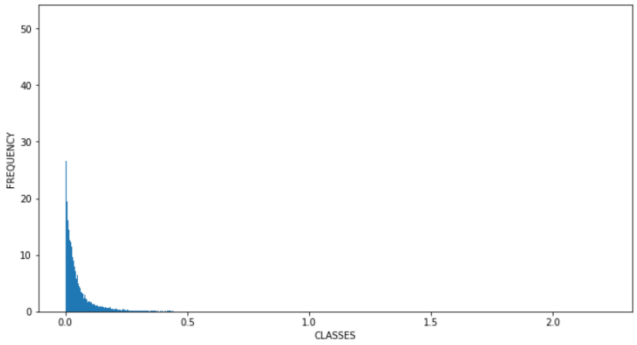

Figure 22: Computer work latitude histogram (all users)

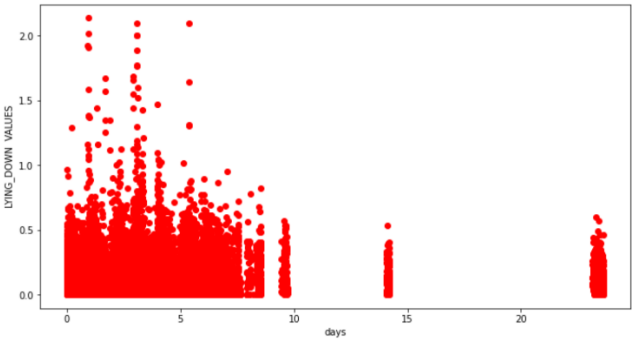

Figure 23: Lying down longitude distribution (all users)

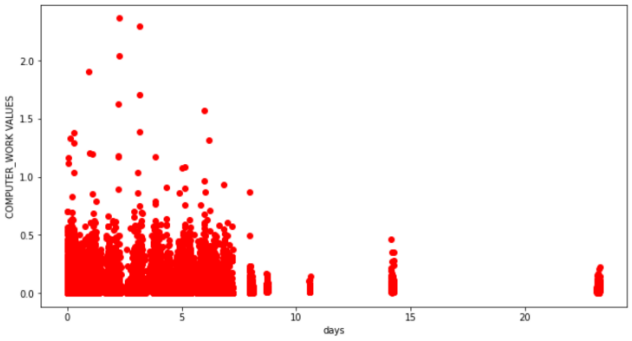

Figure 24: Computer work longitude distribution (all users)

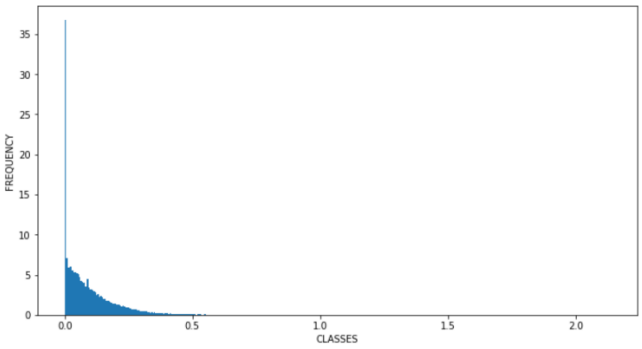

Figure 25: Lying down longitude histogram (all users)

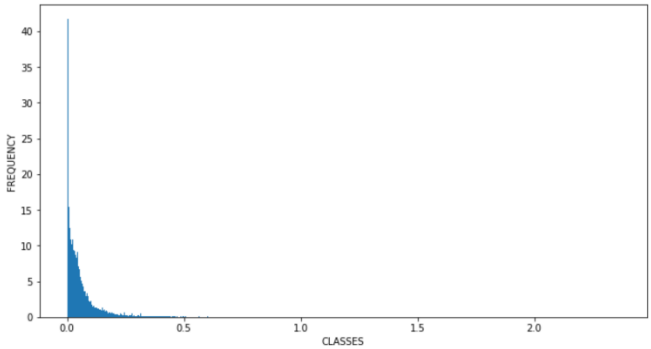

Figure 26: Computer work longitude histogram (all users)

Comparing the result obtained from a random single user and all the users, we can confirm that the data is generally more numerous for the *Lying down.* 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 *Lying down* compared to *Computer work*), while in the last few days the data becomes less and less precise, because not updated anymore and losing the consistency.

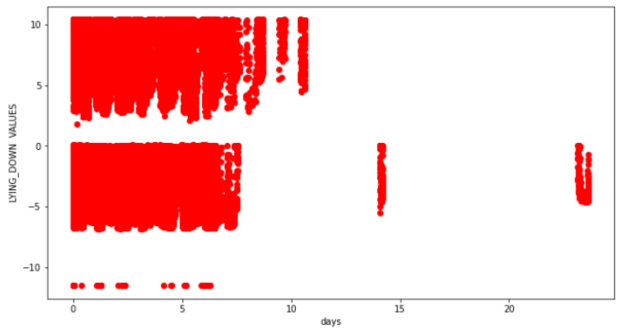

Figure 27: Lying down audio maximum absolute value distribution (all users)

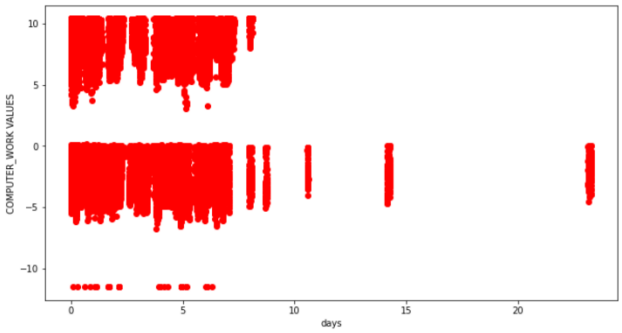

Figure 28: Computer work audio maximum absolute value distribution (all users)

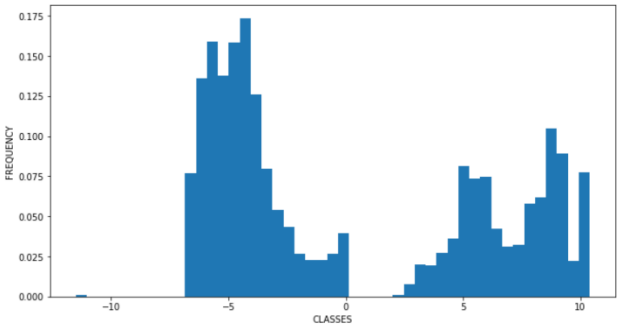

Figure 29: Lying down audio histogram (all users)

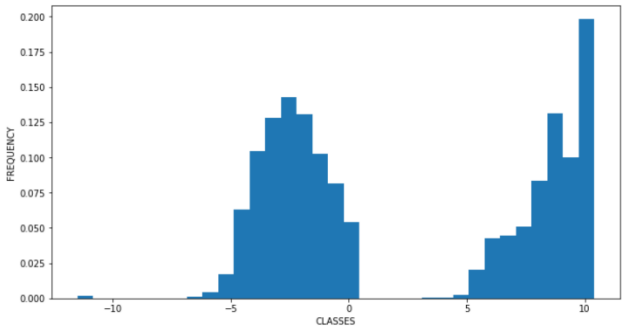

Figure 30: Computer work audio histogram (all users)

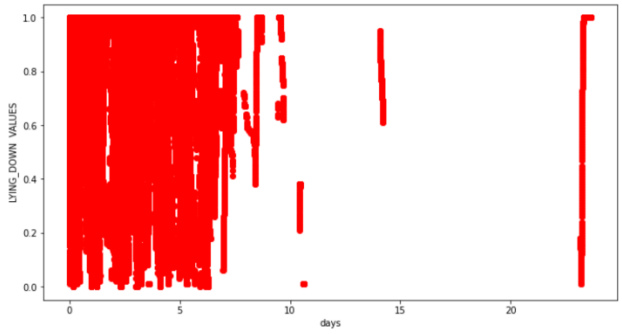

Figure 31: Lying down battery level distribution (all users)

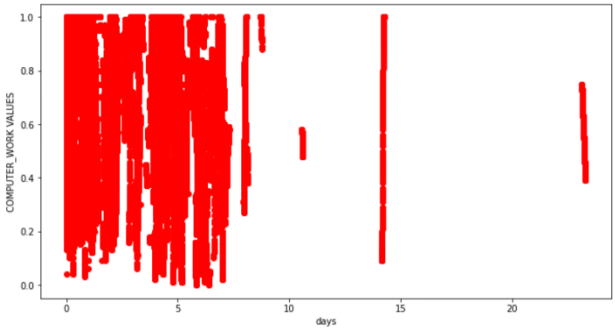

Figure 32: Computer work battery level distribution (all users)

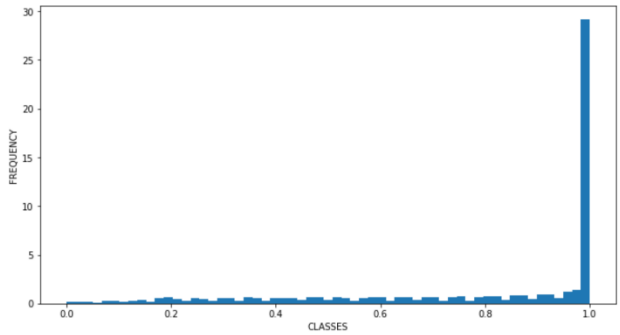

Figure 33: Lying down battery level histogram (all users)

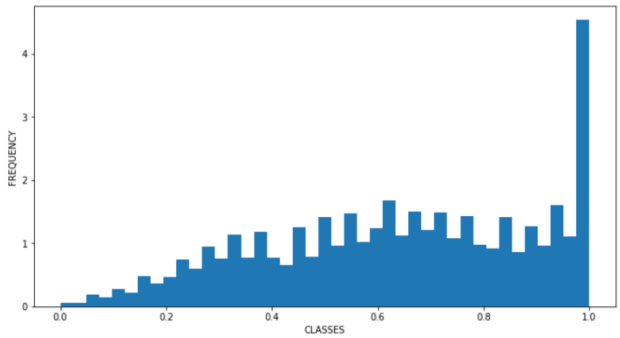

Figure 34: Computer work batter level histogram (all users)

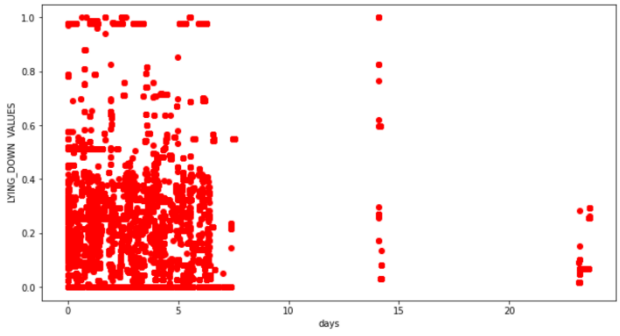

Figure 35: Lying down brightness distribution (all users)

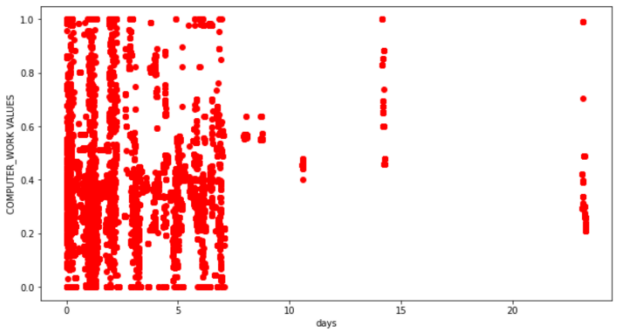

Figure 36: Computer work brightness distribution (all users)

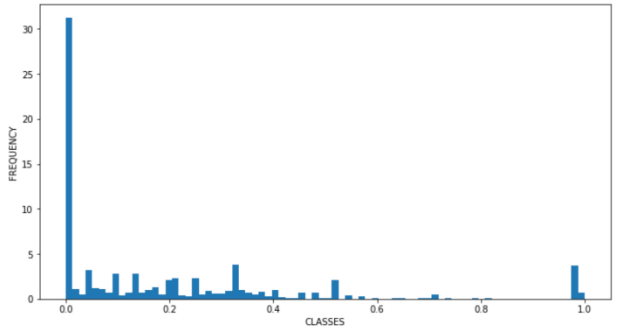

Figure 37: Lying down brightness histogram (all users)

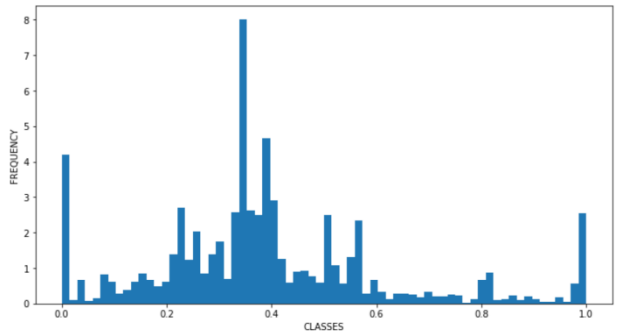

Figure 38: Computer work brightness histogram (all users)

5 DATA CLEANING AND PREPROCESSING

At first sight, we noticed that many cells had NaN values, so we proceeded with the deletion of all the columns that have less than 10% of not NaN values. With an additional investigation, a great number of NaN values was still present in the dataset, for this reason another removal of feature columns was executed related to the columns that had a number of NaN values greater than 50%. At this point we modified the dataset again with the addition of the target column called 'CLASS' to have a binary classification represented in this way: 1 for *relaxed* behaviors, 0 for *stressed* behaviors. The ExtraSensory dataset, as said before, has different columns labeled with *Sitting*, *Lying down* and all the other labels, assuming the value 1.0 if the user was doing that particular activity, 0 if they weren't. To populate our target column 'CLASS', we iterated over the rows of the dataset: if among the columns labeled with *relaxed* labels there was at least an occurrence of 1.0, we assigned 1 to 'CLASS', otherwise, we assigned 0 to 'CLASS'. At this point, we eliminated all the columns labeled with *relaxed* labels, *stressed* 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'. We chose to consider only the properties related to the 5 features analyzed in detail in the previous paragraph: so, if the location had been analyzed, all the properties related to the latter such as the latitude and longitude range, altitude etc. have been included in the dataset. The same reasoning was applied for the audio and low frequency measurements properties. All the other features related to the raw acceleration, proc_gyro, raw_magnet, the watch_heading, discrete, app_state and so on have been removed. Finally, we applied data imputation techniques, which consist in calculating a statistical value for each column (such as the mean value) and replacing all NaN values with this statistic: for this operation we used the *SimpleImputer* library of *itshape scikit-learn*.

5.1 Data Balancing - SMOTE

The distribution of the data example is another important aspect to take into consideration; 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. As follows the plots before and after balancing the dataset with SMOTE:

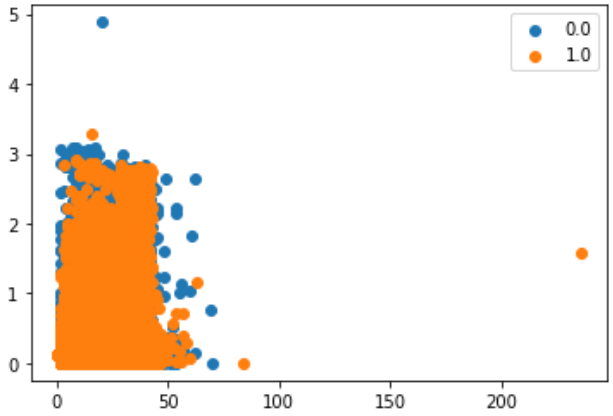

Figure 39: Unbalanced dataset

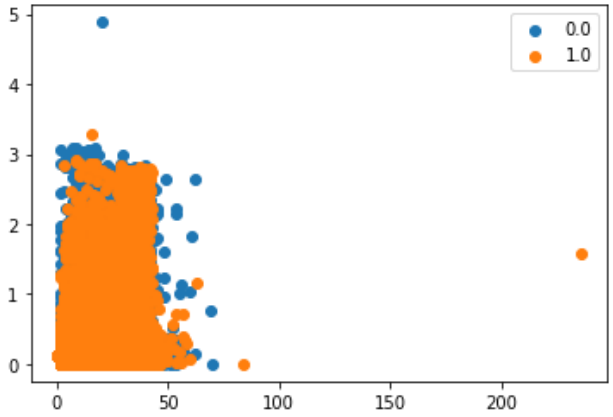

Figure 40: Balanced dataset using SMOTE

6 COMPARISON CLASSIFICATION MODEL

We decided to use two of the most used machine learning models in the context of ExtraSensory dataset: Logistic Regression and Gaussian Naive Bayes. These models are supervised learning models. Once the dataset is prepared, we use the sklearn library '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 the second part is to 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 gives as a strong

evidence of the two split 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 2 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 two parallel classification tasks on the dataset; these results acted as a baseline for any future run. At this point, a first comparison among the two classifiers was made, in order to analyze 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 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.

With the unbalanced dataset, the results are as follows:

```
Best: 0.662793 using {'C': 3.0, 'penalty': 'l2', 'solver': 'sag'}
accuracy:  0.6394232158509685
              precision    recall  f1-score   support

         0.0       0.67      0.70      0.68     58057
         1.0       0.60      0.56      0.58     45760

    accuracy                           0.64    103817
   macro avg       0.63      0.63      0.63    103817
weighted avg       0.64      0.64      0.64    103817
```

Figure 41: LR Unbalanced Classification Accuracy

With the balanced dataset, here are the results:

```
Best: 0.638178 using {'C': 7.0, 'penalty': 'l2', 'solver': 'sag'}
accuracy:  0.5822646634119507
              precision    recall  f1-score   support

         0.0       0.66      0.35      0.46     58346
         1.0       0.56      0.81      0.66     57998

    accuracy                           0.58    116344
   macro avg       0.61      0.58      0.56    116344
weighted avg       0.61      0.58      0.56    116344
```

Figure 42: LR Balanced Classification Accuracy

6.3 Naive Bayes (MOA vs Sklearn)

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 multiclass 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.

With the unbalanced dataset, the results are as follows:

```
Best: 0.597555 using {'var_smoothing': 6.579332246575682e-07}
accuracy:  0.44077559551903833
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00     58057
         1.0       0.44      1.00      0.61     45760

    accuracy                           0.44    103817
   macro avg       0.22      0.50      0.31    103817
weighted avg       0.19      0.44      0.27    103817
```

Figure 42: GNB Unbalanced Classification Accuracy

With the balanced dataset, here are the results:

```
Best: 0.614212 using {'var_smoothing': 1e-06}
accuracy:  0.49850443512342707
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00     58346
         1.0       0.50      1.00      0.67     57998

    accuracy                           0.50    116344
   macro avg       0.25      0.50      0.33    116344
weighted avg       0.25      0.50      0.33    116344
```

Figure 49: GNB Balanced Classification Accuracy

6.5 Naive Bayes - MOA

As said before, we decided to address the problem also by utilizing online machine learning models, so we loaded our dataset (converted into ARFF format) on MOA, setting the parameters as follow:

- instance limit: 100.000.000
- timeLimit: -1
- sampleFrequency: 100.000
- memCheckFrequency: 100.000

Now let's discuss about the results that we obtained with the unbalanced dataset. In Figure 50, we notice that over the time the learner improves its performance, obtaining increasingly correct predictions, reaching an accuracy of 80.0%.

classifications correct (perce...	Kappa Statistic (perce...
36.8	36.8
23.799999999999997	23.79999999999...
9.0	9.0
80.0	80.0

Figure 50: GNB, MOA Unbalanced prediction accuracy

Now let's consider the balanced data on MOA (Figure 51): the learner starts predicting instances wrongly but, as before, it improves the accuracy over time, even though it reaches a lower accuracy than before (73.9%).

classifications correct (perce...	Kappa Statistic (perce...
36.8	36.8
75.7	75.7
23.799999999999997	23.79999999999...
73.9	73.9

Figure 51: GNB, MOA Balanced prediction accuracy

7 RESULTS

In conclusion, we can summarize all the tests that we performed in two categories:

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

In the first case, we obtained a low accuracy (44%) for the Naïve Bayes with scikit, a better accuracy (64%) for the Logistic Regression with scikit and a good accuracy for Naïve Bayes with MOA classifier (80%).
In the second case, we obtained a medium accuracy (50%) for the Naïve Bayes with scikit, a better accuracy (58%) for the Logistic Regression with scikit and a quite good accuracy for the Naïve Bayes with MOA classifier (73.9%).
The results are not what we expected. We see a small improvement only for the Naïve Bayes with scikit (from 44% to 50%), while the other classifiers produce an unexpectedly lower accuracy going from an unbalanced to a balanced dataset.
The cause of these results can be attributed to a low number of labels considered for the two groups (*relaxed* and *stressed*). Perhaps the selection of more labels for each group could result in more data that the learner could use to perform the classification, resulting in better performance metrics. Unfortunately, we didn't choose more labels because we didn't consider them part of any of the two groups. With an unbalanced dataset MOA classifier got the best accuracy with 80%, while the best batch model reached an accuracy of 64%. With a balanced dataset MOA obtained the best results again.

In conclusion, we managed to make a differentiation between *relaxed* and *stressed* activities. Future works could use our study to analyze people's behaviors and lower their stress levels. The ExtraSensory dataset is certainly a complete dataset to perform data analytics, but perhaps an even larger dataset, with people who have various personal habits could give even better results.

SMOTE	CLASSIFIER	ACCURACY
Unbalanced	Logistic Regression	64%
Balanced		58%
Unbalanced	Naïve Bayes	44%
Balanced		50%

Figure 52: scikit classifiers accuracy

SMOTE	CLASSIFIER	ACCURACY
Unbalanced	Naïve Bayes with MOA	80%
Balanced		73.9%

Figure 53: MOA classifer accuracy

REFERENCES

[1] Machine Learning and IoT for Prediction and Detection of Stress, Mr. Purnendu Shekhar Pandey

[2] Towards Measuring Stress with Smartphones and Wearable Devices During Workday and Sleep, Amir Muaremi, Bert Arnrich, Gerhard Tröster

[3] Can smartphones detect stress-related changes in the behaviour of individuals?, Gerald Bauer, Paul Lukowicz

[4] StressSense: Detecting stress in unconstrained acoustic environments using smartphones, Hong Lu, Denise Frauendorfer, Mashfiqui Rabbi, Marianne Schmid Mast

[5] Stress Detection Using Smart Phone Data, Panagiotis Kostopoulos, Athanasios Kyritsis, Michel Deriaz, Dimitri Konstantas

[6] Assessing Mental Stress Based on Smartphone Sensing Data: An Empirical Study, Feng Wang, Yasha Wang, Jiangtao Wang, Haoyi Xiong