uc3m | Universidad Carlos III de Madrid

Bachelor's Degree in Telecommunication Technologies

Engineering

2017-2018

Bachelor Thesis

"Development of a bodyguard application in Android"

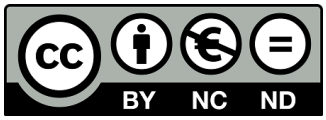
Daniel Marcos Mazón

Tutor

Mario Muñoz Organero

Leganés

03 - 07 - 2018



ABSTRACT

In this thesis it is going to be explained the process to develop a bodyguard application using Android. This application should be able of properly detecting whether the user is suffering a physical aggression or not and, if so, send an email alert to an emergency contact previously established. It will be also tried to extend the classification to additional movement patterns.

For that, a first process involving the collection of the training samples will be performed. Once a proper dataset is obtained, different classifiers such as kNN, Multilayer Perceptron or Support Vector Machines will be explored to decide which one fits better to the activity recognition and, after that, a feature reduction process is applied to reduce the number of attributes in the classification with the purpose of making the application more efficient with respect to the number of calculations. Finally, an optimization of the selected classifiers comes to obtain the highest accuracy possible.

In the end, it is obtained an Android application that properly differentiates between aggression and non-aggression movements and capable of sending the email alert when needed in a low amount of time. With respect to the classification of additional movement patterns, it was not achieve a high precission for all of the classifiers considered due to inaccuracy problems in the training data that will be explored in future works.

Key words: Weka, Supervised Learning, Multilayer Perceptron, kNN, Activity Recognition

ACKNOWLEDGMENTS

First of all, I would like to thank my tutor, Mario Muñoz, for trusting me with this project as well as for all the guidance received from him during the whole process.

Special thanks to those people that took part in the project and helped me gathering the necessary data. Without them this project would not have been possible.

Thanks to my family for all the support and trust given along this years. I know I will always be able to count on you.

Finally, thanks to all the people I have met in the way that have helped me to continue working and achieve my goals.

Thanks to all of you.

GENERAL INDEX

FIGURE INDEX

TABLE INDEX

1. INTRODUCTION

1.1. Problem statement

In recent years, there has been a rising interest in the information enclosed in all the data produced as it is a key element in the development and improvement of different activities such as marketing strategies, a company can make use of people interests in order to focus on a certain market, or medical diagnosis, data collected from previous diseases can help recognizing them easily in the future. This has been possible thanks to the advancements in machine learning which hava allowed to perform better and deeper analysis on the data, gathering useful information and making it understandable.

In this thesis project, knowing that nowadays it is estimated that around 62.9 % of the global population has a mobile device [1], it is intended to develop a mobile application that, making use of the data gathered from the user's movement activity and different machine learning techniques, is capable of properly distinguish whether the user is suffering a physical aggression or not in real time and, if so, send an alert to an emergency contact.

With this aim, a machine learning tool called Weka will be used in the design and testing of different classification algorithms to find which one fits better when solving the classification problem. Matlab will also be used as an auxiliary tool for calculations in the feature extraction process. Later on, once a proper model has been obtained, its functionality will be added to the mobile application by means of the programming language Java.

As additional objectives the following ones were thought:

- As for detecting an aggression it is necessary to analyze the movement patterns performed by the user, it will be tried to extend this analysis to a wider range of movements such as walking or running among others, instead of focusing only in aggression patterns, so in a future the application could be extended to more fun-

ctionalities.

- Whenever an aggression is detected, the application will automatically send a notification via email to an emergency contact previously selected so the user can get help as soon as possible.

The motivation found to start the project was introducing a new approach in the area of physical activity recognition focused in the personal security. It is thought that this thesis may be of use in future works as it introduces a base analysis in the movement classification problem from which develop more specific applications. Apart from that, the final application designed may add real value to people as it can be used with a clear beneficial purpose in a daily basis.

On the other hand, in chapters 4 and 5 the regulatory framework as well as the socio-economic impact are explained. In the regulatory framework different regulations, such as licenses and data protection, that apply to the development of the appication are analyzed. In the socio-economic impact it is analyzed the budget of this thesis and how the application can be taken advantage of economically by making an study of the possible revenues. Also it is estated the possible impact of the program in the society.

1.2. State of the art

In this section, an analysis of previous works related with the task carried out on this thesis will be made in order to establish an starting point in the development of the project and to obtain a clear understanding of the subject. Two approaches are considered: previous works in the classification of movement patterns using machine learning algorithms and similar existing applications in the market.

1.2.1. Movement patterns classification

The first part of the analysis will be conducted through different previous works related with the study of physical activity recognition so a first insight of the existing approaches for this kind of classification is obtained. All of them have in common the search for a set

of features to compose the dataset as well as a proper classification algorithm that could yield good results when differentiating between the movement patterns. It is important to notice that these works has the only purpose of recognizing movement and not future applications of it.

In [2], it is stated the process followed in order to be able to detect daily physical activities such as walking, running or sitting among others. It starts collecting the necessary data from the accelerometer, placed in the waist area, to train the model using a set of subjects with a wide range of physical characteristics so the model is as general as possible. The most characteristic part of this work is the classification model generation, as it does not make use of a classical machine learning algorithm but of the Hidden Markov Models, which produces an output depending on the given input and a different set of states, each one having a probability distribution related with the outputs. This set of states are trained from an initial state distribution, initialized at the beginning of the process.

With respect to the obtained results after the training process, it is achieved a classification accuracy of 94.8 %, being the classes with the worst performance sitting and falling. This work is concluded arguing that a good model has been obtained due to its stability and efficiency.

Turning into [3] and [4], it is possible to observe a different approach to the classification of movement patterns. Again, both works have a similar procedure with the one previously analyze as they involve many different subjects to obtain a generalize model as well as make use of one accelerometer positioned near the waist.

The first main difference is that now the raw data obtained from the accelerometer is processed so a set of features for each sample is obtained and introduce in the design of the classification models. Both works make use of an statistical approach to characterize the movements using attributes such as the mean, the standard deviation or the correlation of each one of the axis of the accelerometer. Remark that [4] makes use of the Fast Fourier Transform in order to obtain the energy concealed and use it as another feature for the

classification.

Another difference is that, instead of testing the performance of only one algorithm, a search for the best accuracy is conducted through various models such as a Multilayer Perceptron, Naive Bayes, Logistic Regression, J48 tree, kNN or SVM. They both concluded that a sufficiently good model has been obtained as they achieve accuracies of around 90 % in [3] and around 95 % in [4] for the different models considered.

Knowing this, it is considered that in this thesis the second approach will be used as it is the best way to find out which model adapts better to the problem and has more flexibility as it considers various algorithms and not just one. With respect to the data collection, as all of them uses the same procedure with good results, it is concluded that the same method will be used.

It was also taken into account works related with the topic of feature reduction, as it is known that very large datasets are very cost computationally as well as may drive to an overfitted model. Different works were studied so it could be seen the effect of various feature selection algorithms over the selected classification models.

In [5], it is presented the influence of Principal Component Analysis (PCA) over already tested algorithms such as kNN or decision tree C4.5. After training the models with normal data and with PCA data, it is concluded that PCA may have a positive or negative effect depending on the algorithm and the dataset used. For example, in C4.5 decision tree, the performance was worsened after applying PCA but in kNN it was improved. One solution given for C4.5 algorithm is to use PCA in conjunction with he original dataset, but then the dataset is not reduced but increased.

Another method is shown in [6]. Here it is presented the Information Gain algorithm, which selects features with respect to the information they have from the class. This time, the algorithm is tested against other feature selection methods, such as filter methods and wrapper methods, in different classifiers and it is concluded that, if a proper selection of

attributes is performed, Information Gain could achieve better results than the others.

Finally, in the Weka documentation [7], it is possible to find an interesting feature selection method called Correlation Based Subset Evaluation(CBSE), which analyze different subsets of features that are low correlated between them but highly correlated with the class, thus producing a dataset with low dependency among its variables but with high information about the class.

As a conclusion, due to the fact that the methods considered base their performance depending on the selected classifier, it is decided that a whole section will be devoted to analyze their effect over the models considered at that point.

1.2.2. Mobile applications

In this section they are going to be considered different technologies that nowadays allows the development of mobile applications as well as existing applications similar to the one that is to be developed. The advantages and disadvantaged of each one will be considered with the aim of obtaining the best view of how the application should be designed in order to perform well and be competitive.

Regarding the technology use in the development of the application, different operating systems for smartphones, such as Android, iOS and Windows, along with their corresponding programming languages (Java, C, Swift) can be found, so it is important to drive a deep analysis to see which one fits better to the established objectives.

In figure 1.1, it is shown the market share distribution for the most popular operating systems from 2009 to 2016. It can be seen how along the years the presence of different operating systems has been reduced just to two of them, with a clear majority of Android over iOS. This superiority is due principally to the low prices of Android devices compared with those using iOS as well as the high compatibility of Android applications along the vast majority of devices in the market while iOS is only supported in Apple products.

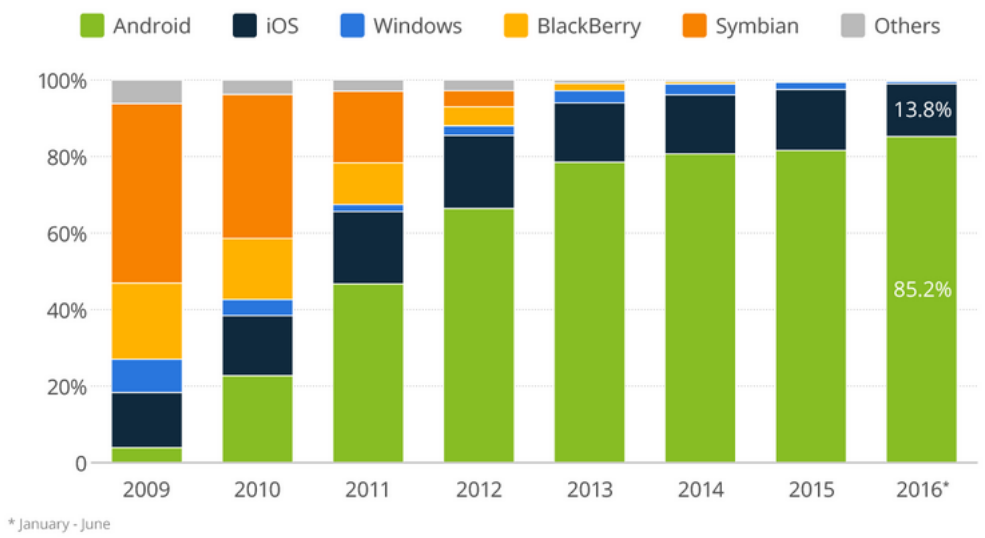


Fig. 1.1. Market share distribution for different operating systems[8]

Along with the previous conclusions, it is useful to focus in the programming language used by the operating systems. In table 1.1, it is shown the TIOBE ranking[1] at May 2018[9]. It can be seen that the most popular programming language nowadays is Java closely followed by C, which implies that selecting Java as the programming language for the development of the application may give advantages as the market seems to tend towards it.

Position	**Programming language**	**Rating %**
1	Java	16.280
2	C	14.000
3	C++	7.668
4	Python	5.192
5	C#	4.402
6	Visual Basic .NET	4.124

TABLE 1.1. TIOBE INDEX FOR PROGRAMMING LANGUAGES.

With respect to similar existing applications, after an exhaustive search through the main platforms of applications distribution, the next ones were found:

[1]The TIOBE ranking indicates the popularity of different programming languages based on engineers all around the world and applications that makes use of them.

- **Care App:** this application uses constant pop-up notifications that the user should deactivate in order to check that everything is alright. In case the user does not disables the notification, a trusted contact will receive an alert indicating that something may be happening. During all this process, the application is recording audio, video, location and route and storing it in the cloud in case it would be needed.

- **Companion:** this application allows the user to select a trusted contact with which share the route the user is about to follow. At any moment, the contact can check if the user is following the established path or, if the application detects something has happened, receive an alert.

- **GPS Bodyguard:** this application sends an alert with the user's GPS location in case there is no feedback from him/her, if the panic button is pressed or if an action area is abandoned.

- **Movement tracker:** this application allows to track all the activities done by the user as well as to locate the position at which these activities were performed.

It is easy to notice that all the applications described have common characteristics with the application to be developed but none of them can be fully compared. Those applications regarding security are based on the interaction of the user and third parties in order to guarantee the security, which is not always the better approach as the user may not able to interact with the device. Anyway, the features of these applications could be a good starting point for the development of the application at hand.

2. METHODOLOGY

In this chapter, the methods followed for the development of the application will be explained, as well as contrasted with other techniques, ranging from the designing of the data gathering and the different classification algorithms to the development of the final application with all its features and processes, in order to show why certain decisions where made and how they impact on the final outcome.

2.1. Data acquisition

The raw data used to characterize the different movement patterns was obtained from the mobile device sensors, specifically the linear accelerometer and the gyroscope, via an application developed for this only purpose.

The linear accelerometer allows to measure the acceleration applied in each one of the axis without taking into account the gravity force; this is an important point because if the gravity effect is not eliminated, the orientation of the device will have a large effect in the measurements, which may yield inaccuracies in the classification. An alternative to the use of the linear accelerometer would have been the use of the normal accelerometer but keeping the device always in the same position while recording the movements so the gravity force has the same effect over each axis at all times. With respect to the gyroscope, it gives information about the angular velocity as well as the orientation of the device.

Both linear accelerometer and gyroscope measurements are given using a 3-axis coordinate system (each axis having a unique value, yielding six total values to define the movement) set up in a relative position with respect to the mobile device (see Figure 2.1).

Regarding the application used, its steps can be seen in the flowchart in Figure 2.2. After the initialization, a file should be created in order to store all the measurements. Once we start measuring data, this step will not stop until it is said so and each measurement

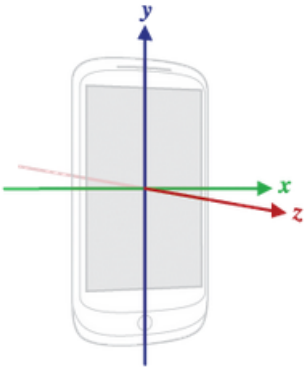


Fig. 2.1. Coordinate system for linear accelerometer and gyroscope.[10]

will be written to the output file in the following format:

AccX	AccY	AccZ	GyrX	GyrY	GyrZ
0.5252	-1.1927	-1.3984	-1.3191	0.0247	0.0807
-0.2235	-1.9247	-2.4195	-1.2392	-0.0838	-0.0167
-0.9559	-2.0781	-1.7091	-0.9590	-0.1760	-0.1110
...	...	...	...	...	...

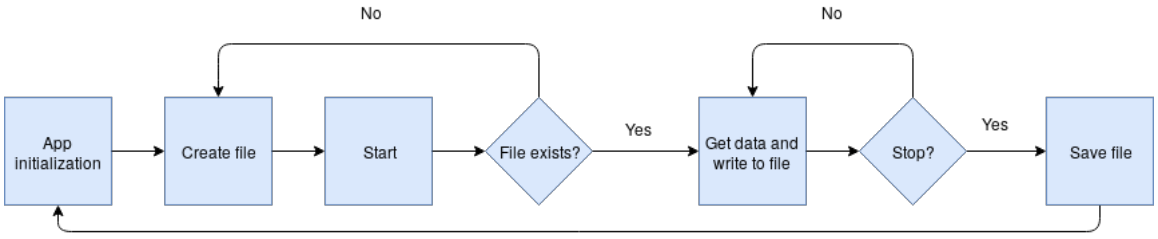


Fig. 2.2. Flowchart of the acquisition application

The gathering of data was performed using a Xiaomi Redmi 4X (see Annex A for detailed information), which was placed in a pocket near the waist as that is a usual position to carry the mobile device. Five different subjects took part in the data collection; it was tried to gather people with different physical characteristics in order to give diversity to the study. A collection of this characteristics is shown in Table 2.1.

With respect to the temporal length of the samples, as the aim of this work is to reach real time predictions, the sensor measurements were taken at a rate of 50 Hz, that is, one measurement both from accelerometer and gyroscope every 20 ms. As the triaxial data has

Subject	Gender	Height (cm)	Weight (kg)	Age
1	Male	192	73.5	23
2	Male	182	72	22
3	Female	165	48	21
4	Male	189	95	19
5	Male	182	64	22

TABLE 2.1. PHYSICAL CHARACTERISTICS OF THE INDIVIDUALS

no useful meaning by itself for the classification problem, samples of 1.5 and 3 seconds were produce in order to properly characterize the movement pattern they represent. Later on, both accuracies will be tested to see which one achieves better results in the predictions.

The different movement patterns considered for this application were walking (*Walking*), running (*Running*), going upstairs (*WalkUp*) and going downstairs (*WalkDown*) for non-aggression patterns, and lying on the floor[2] (*Floor*) and fighting (*AgrStand*) for physical aggression patterns. These different patterns are distributed almost equally between the total number of samples as seen in Figure 2.3 so, at the time of training the classification algorithms, there is no class with a higher weight.

2.2. Feature and dataset generation

Once all movement samples are conveniently stored and labeled according to their class, it is possible to start the preprocessing of the data in order to make it understandable for the classification algorithms. To achieve this, it was decided to design a dataset containing a collection of different features as well as a class label for each one of the samples.

In Figure 2.4, it can be seen a representation of the sensor data behavior along time

[2]Meaning to be suffering an aggression while on the floor as well as lying still.

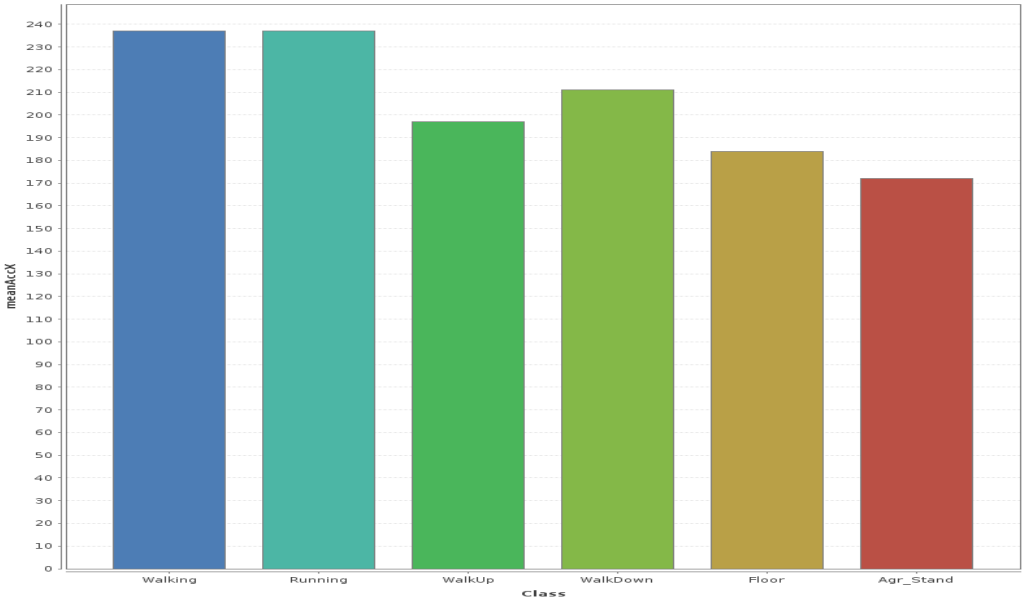


Fig. 2.3. Histogram for the movement patterns distribution

for the considered patterns. It is easy to notice remarkable differences between the movements; with respect to dynamic usual patterns, *Walking* is easily differentiated as it has a smooth behavior while *Running* ,*WalkUp* and *WalkDown* show abrupt changes. These last three patterns can be distinguished between them by its periodicity as well as the values taken from the sensors, which are higher for the *Running* pattern. With respect to the aggression patterns, *Floor* is differentiated from other patterns due to its negligible variations around a constant value[3] while *AgrStand* shows a similar periodicity to *Running* and*WalkDown* but with higher and faster values. After this analysis and taking into account the conclusions from [3] and [4], it was decided to use a statistical approach in the characterization of the movement pattern.

[3]This constant value is 0 as in a static movement such as been lying on the floor there is minimal acceleration applied to the device. Also the variations represented around this constant value are due to the precision of the sensors used.

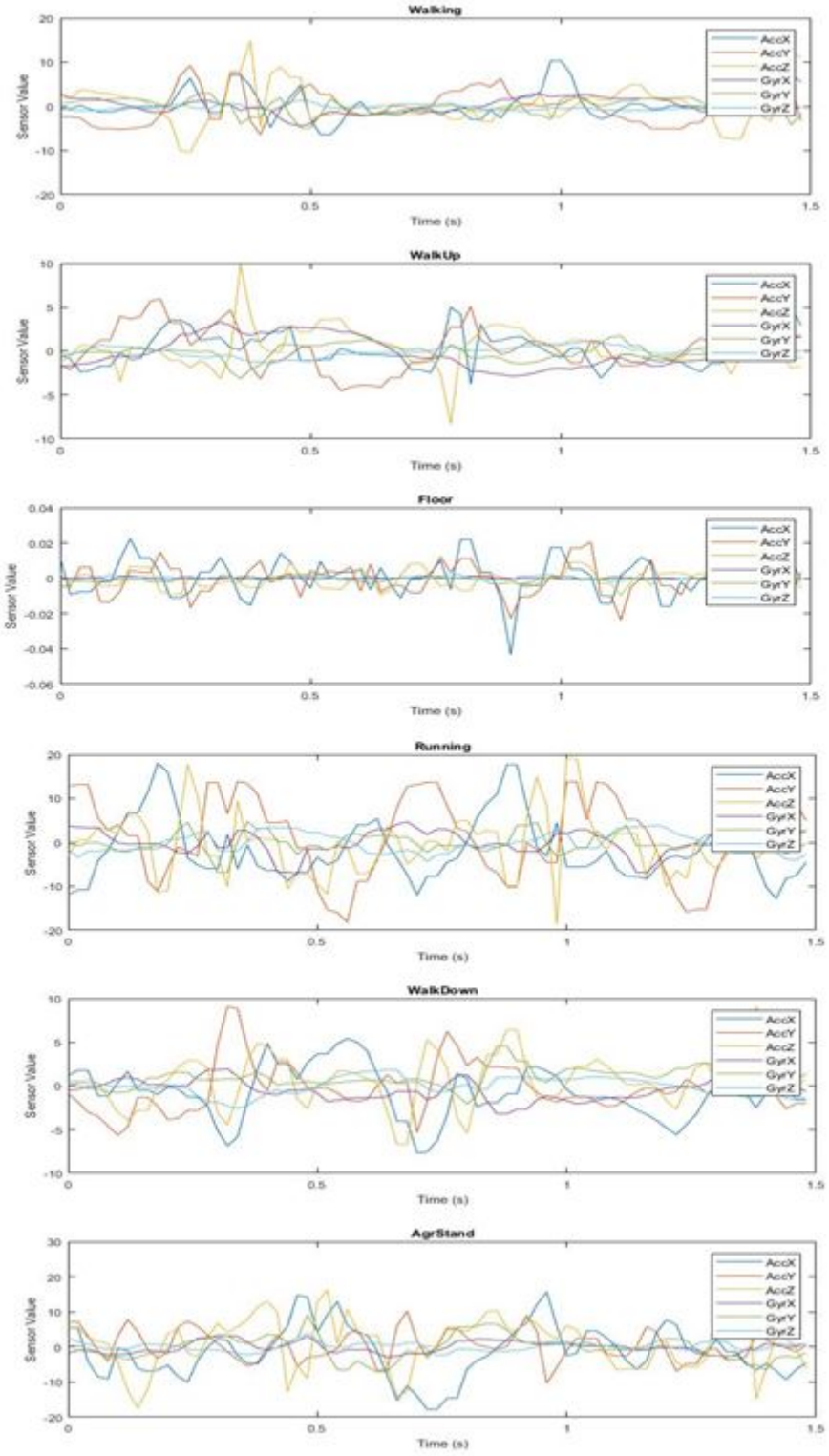


Fig. 2.4. Sensor data representation over time

So, for the dataset generation, it was tried to explain the data considering each axis individually as well as taking into account the distribution of the values and how related are the axis to each other. This was achieved by means of the following features:

Mean value for each axis: computed as the sum of all the values of that axis divided by the total number of values.

$$E(X) = \frac{1}{k}\sum_{n=1}^{k} x_i$$

Standard deviation: a measure to express how spread the data is and calculated as the squared root of the variance.

$$Std = \sqrt{Var(X)} = \sqrt{E(X^2) - E^2(X)}$$

Root mean square: to characterize each axis with no effect from negative measurements. It is given by the following formula:

$$RMS = \sqrt{\frac{1}{k}\sum_{n=1}^{k} x_i^2}$$

Average acceleration(AA): computed as the average of the squared root of the sum of the squared values of each accelerometer axis.

$$AA = \frac{1}{k}\sum_{i=1}^{k}\left\{\sqrt{AccX_i^2 + AccY_i^2 + AccZ_i^2}\right\}$$

Average angular velocity(AAV): computed as the average of the squared root of the sum of the squared values of each gyroscope axis.

$$AAV = \frac{1}{k}\sum_{i=1}^{k}\left\{\sqrt{GyrX_i^2 + GyrY_i^2 + GyrZ_i^2}\right\}$$

Absolute mean difference(AMD): for each axis, the average of the difference of each value and the mean is computed.

$$AMD = \frac{1}{k}\sum_{i=1}^{k}(x_i - E(X))$$

Binned distribution: for each axis, the maximum and minimum values are computed and the whole range of values is divided in five bins. After, a percentage of how many values are inside each of the bins is compute to determine a distribution of the values along the range.

Correlation: express how related is one axis to another. It is computed for each pair of axis.

$$Corr = \frac{Cov(X,Y)}{Std(X)Std(Y)}$$

This set of statistical attributes yields a total of 62 features, that along with the class label, produces a dataset with 63 parameters for each one of the samples. For optimizing the features calculations, a Matlab script was used in order to process all stored samples with their corresponding label and transform them into the dataset that would be used in future steps of the analysis.

2.3. First approach to classification

In this section, different algorithms are briefly evaluated with the aim of selecting which ones fit better to the classification problem and arguing if 1.5 seconds measurements are large enough to properly characterize the movement. For this purpose, the software Weka, developed by the University of Waikato and designed as a machine learning tool, is to be used.

In this first analysis, algorithms such as J48 tree[11], IBk[12], Multilayer Perceptron[13], SMO[14] and Naive Bayes[15] are going to be evaluated. All these algorithms

were tested for both 1.5 and 3 seconds with default parameters as established by Weka and evaluated using 10-fold cross validation. The results can be seen in Table 2.2.

	J48 (%)	**IBk (%)**	**MLP (%)**	**SMO (%)**	**Naive Bayes (%)**
1.5 (s)	91.4378	96.3615	95.7189	95.3958	94.2640
3 (s)	92.9254	***96.5764***	95.9489	94.4412	95.1402

TABLE 2.2. ACHIEVED ACCURACY FOR EACH OF THE CLASSIFIERS AND DATASETS

As it can be observed, the best accuracy is obtained for IBk classifier with the 3 seconds measurements dataset. In fact, better results are achieved when the 3 seconds dataset is used. The reason is that in 3 seconds there is more information about the movement and so it is easier for the classifier to distinguish between classes.

Despite of this better performance, it was decided to implement models trained with the 1.5 seconds dataset because the objective is detecting aggression in real time and the classification must be made as fast as possible. It is noticeable that the best results are obtained using IBk and MLP classifiers so a more in depth analysis will be made. In the following tables, additional data for these two classifiers from the previous test can be seen.

According to the TP Rate[4], the FP Rate[5] and the Precision[6] values presented in Table 2.3, it is possible to observe that both classifiers present good results as they achieve a precision of 96.4 % and 95.7 %, which implies a low error rate. Specifically, from de TP Rate it can be concluded that both algorithms are able to classify an actual aggresion as such for the majority of the times. Moreover, taking a look at Tables 2.4 and 2.5, it can be noticed how the number of samples correctly classified are higher than the number of wrongly classified.

[4]The true-positive rate shows the proportion of samples classified to its true class.[16]

[5]The false-positive rate indicates how many samples are classified as a specific class when they do not actually belong to it.[16]

[6]Precision establish how well the algorithm is classifying for each class. [16]

	IBk			MLP		
Class	*TP Rate*	*FP Rate*	*Precision*	*TP Rate*	*FP Rate*	*Precision*
Walking	0.983	0.004	0.983	0.966	0.014	0.942
Running	0.979	0.003	0.987	0.975	0.001	0.996
WalkingUp	0.929	0.012	0.938	0.944	0.014	0.925
WalkingDown	0.967	0.020	0.907	0.943	0.012	0.943
LyingOnFloor	0.989	0.003	0.984	1.000	0.004	0.979
AggresionStanding	0.924	0.002	0.988	0.907	0.007	0.957
Weighted Avg.	0.964	0.007	0.964	0.957	0.009	0.957

TABLE 2.3. PERFORMANCE OF IBK AND MLP CLASSIFIERS FOR EACH CLASS

classified as –>	**a**	**b**	**c**	**d**	**e**	**f**
a = Walking	**233**	0	1	3	0	0
b = Running	1	**232**	1	3	0	0
c = WalkUp	0	1	**183**	11	1	1
d = WalkDown	2	2	3	**204**	0	0
e = Floor	0	0	1	0	**182**	1
f = AgrStand	1	0	6	4	2	**159**

TABLE 2.4. CONFUSION MATRIX FOR IBK CLASSIFIER

In the same line, given that the objective of this project is to decide whether the user is suffering a physical aggresion or not rather than detecting which kind of movement is performing, it can be assumed that this decision may be reduced to a binary classification, where the decisions are Agresion and NotAgresion. If this conclusion is applied, the results obtained are shown in tables 2.6 and 2.7.

In this last approach accuracies of 98.86 % and 98.30 % are achieved for IBk and MLP respectively, which yields a system capable, at least theoretically, of detecting physical aggression in real time.

classified as –>	**a**	**b**	**c**	**d**	**e**	**f**
a = Walking	**229**	0	2	2	0	4
b = Running	1	**231**	4	1	0	0
c = WalkUp	2	0	**186**	5	1	3
d = WalkDown	5	1	6	**199**	0	0
e = Floor	0	0	0	0	**184**	0
f = AgrStand	6	0	3	4	3	**156**

TABLE 2.5. CONFUSION MATRIX FOR MLP CLASSIFIER

classified as –>	**a**	**b**
a = Agresion	**880**	2
b = NotAgresion	12	**344**

TABLE 2.6. CONFUSION MATRIX FOR IBK BINARY CLASSIFICATION PROBLEM

2.4. Dimensionality reduction

After selecting two possible classification algorithms, a dimensionality reduction is to be performed in order to reduce the number of features considered. This will result in a better performance with respect to computation time as the number of calculations needed to characterize the movement will be lower. Also it may be possible that the results obtained with the classifiers improved, as dimensionality reduction could imply a reduction in the redundancy that the original dataset may have.

Three different processes will be analyzed: Principal Component Analysis (PCA), which makes use of feature transformation, Information Gain Feature Selection and Correlation Base Subset Evaluation. These three models will also be evaluated using the classification algorithms previously selected.

classified as –>	**a**	**b**
a = Agresion	**874**	8
b = NotAgresion	13	**343**

TABLE 2.7. CONFUSION MATRIX FOR MLP BINARY CLASSIFICATION PROBLEM

2.4.1. Principal Component Analysis

According to [5], Principal Component Analysis is a multivariate statistical technique that allows to transform a given dataset of dimensions $m \times n$ into a completely different dataset of dimensions $m \times n'$, where $n > n'$. This new dataset is composed of attributes called *principal components* that are calculated as a linear combination of the original features. The weights of these linear combinations are extracted from the eigenvalues of the correlation matrix. In order to measure the performance of this procedure, and knowing that the principal components are directly related to the explained variance of the original dataset, IBk and MLP algorithms will be evaluated using different reduced datasets according to the number of new features and the explained variance.

In figure 2.6, it can be seen the influence of PCA over the performance of IBk and MLP. In the x-axis is represented the number of principal components used, as well as the proportion of the explained variance achieved for that number of principal components, and in the y-axis it is shown the accuracy achieved. The maximum accuracy, with a value of 94.6688 % for IBk and 94.5886 % for MLP, is achieved for a total number of 21 principal components explaining the 85 % of the variance, which implies a reduction of more than half the original features without crippling too much the performance of the classifiers[7].

It could be concluded then that it should be used PCA with 85 % of the variance for the reduction of the original dataset as it yields a more than enough reduction of variables while keeping the performance of the classifiers (In Annex B it can be seen the 21 selected

[7]Without PCA, the accuracies achieved for IBk and MLP were 95.5616 % and 94.7897 % respectively (see table 2.2).

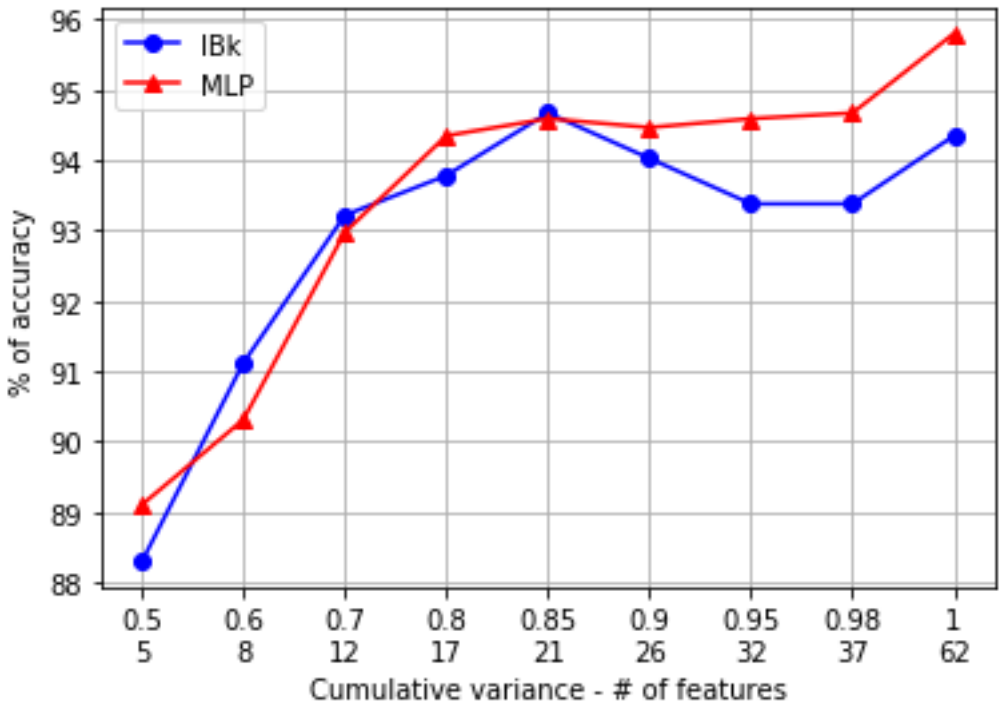


Fig. 2.5. Accuracy of IBk and MLP with respect to the number of features and the explained variance.

principal components along with its linear combination of original features). Despite of keeping a decent accuracy after reducing the dataset dimensionality, this method does not take into account the relationship of each original feature with the class, which makes the reduced dataset more difficult to interpret in relation to the problem at hand.

2.4.2. Information Gain Feature Selection

Information Gain Feature Selection (IGFS) is a method that selects the subset of original features that contain more information about the classes to be predicted. This is a powerful method as it allows to reduce the number of features to only those containing useful information for the classification. More in depth, as stated by [17], information gain is computed as the difference between the entropy of a given feature f_i and the entropy of that same feature after observing C:

$$IG(f_i, C) = H(f_i) - H(f_i|C)$$

In this analysis, a threshold will be used in order to select which features contain enough

information to be selected. As in the previous section, different thresholds will be tested along with the selected classifiers in order to obtain the optimum subset of features. In figure 2.4 it can be seen the achieved accuracy for each of the algorithms with respect to the selected information gain threshold and the number of resultant features. It is possible to observe how the accuracy of both classifiers decreases as the threshold is increased due to the fact that there is a smaller number of features so highly related with the class. For the problem treated, the optimum number of features is 24[8] as the accuracy achieves its highest value both for IBk and MLP at this point, having only features whose information gain is higher than 0.75. See Annex C to observe the 24 selected features.

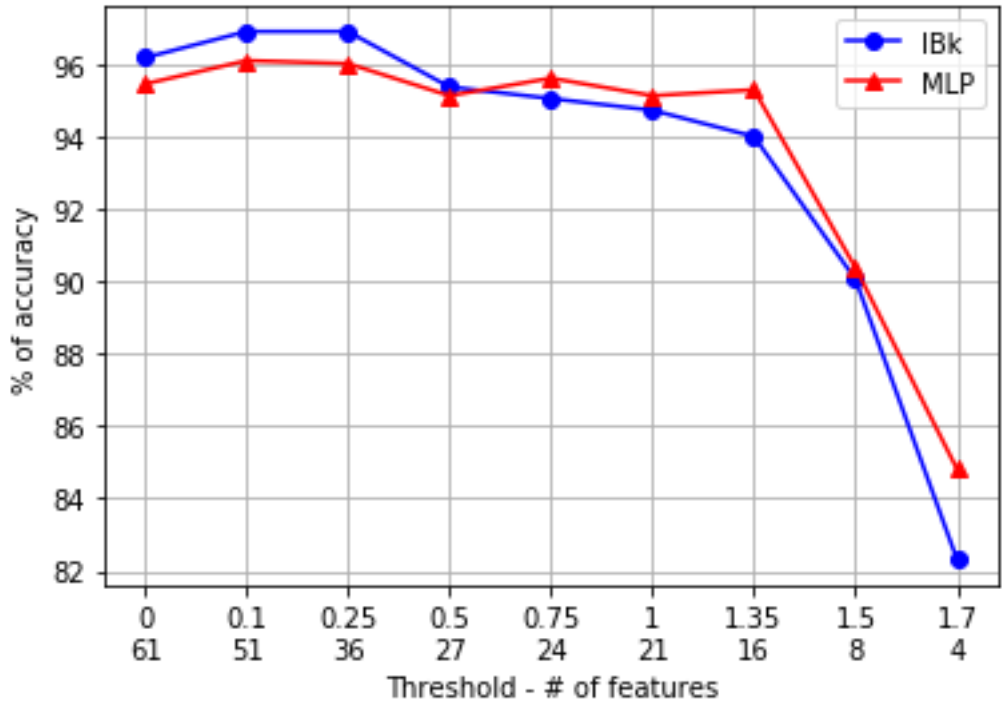


Fig. 2.6. Accuracy of IBk and MLP with respect to number of features and information gain threshold.

Despite of being able to increase the accuracy of IBk and MLP to 95.0727 % and 95.6381 %, it has to be taken into account that this method, although it measures the relation of each feature with the class, does not consider any possible relationship between features, which may produced a reduced dataset with redundant data.

[8] Actually for 24 features the accuracy for IBk differs from the maximum accuracy at 27 features in 0.3 %. Despite of this, 24 features are selected as the feature reduction is higher and the difference in accuracies is negligible.

2.4.3. Correlation Based Subset Evaluation

In order to solve the problems had in previous sections such as difficult interpretation of the resultant dataset in PCA or redundant data in IGFS, Weka has a feature selection method called Correlation Based Subset Evaluation (CBSE). This method evaluates different subsets, each one composed of low correlated features that highly correlates with the class. With this it is assured that the final selection of features will be highly related with the class without having redundant data.

Weka has a tool that automatically evaluates different subsets and computes the optimum subset of features. The selected features can be seen in table 2.8. After creating the dataset with the 33 selected variables, IBk and MLP classifiers are evaluated obtaining an accuracy of 96.6882 % for IBk and 96.1228 % for MLP.

Selected features		
meanAccX	meanAccY	meanAccZ
meanGyrX	meanGyrY	rmsAccX
rmsAccY	rmsGyrX	AvgAcc
AngVel	stdAccY	stdAccZ
stdGyrY	AbsDiffAccY	AbsDiffAccZ
AbsDiffGyrY	AbsDiffGyrZ	Bin3AccX
Bin2AccY	Bin5AccY	Bin3AccZ
Bin1GyrX	Bin3GyrX	Bin5GyrX
Bin2GyrY	Bin1GyrZ	Bin3GyrZ
CorrAccXY	CorrAccXZ	CorrAccZY
CorrGyrXY	CorrGyrXZ	CorrGyrZY

TABLE 2.8. SELECTED FEATURES USING CBSE.

2.4.4. Conclusion

After evaluating the reduced datasets provided by the different dimensionality reduction methods, it was concluded that CBSE should be use instead of PCA or IGFS. Despite that the number of reduced features is higher in CBSE than in PCA or IGFS, it presents better results in the accuracy obtained both for IBk and MLP as well as it is guaranteed that these reduced features will keep lower correlation level between each other while maintaining a higher dependency with the class.

2.5. Classifiers optimization

In this section, using the two algorithms previously selected, a parameter optimization will be performed in both of them in order to produce the best results possible in the classification.

2.5.1. K nearest neighbors

The first selected algorithm is a K nearest neighbors algorithm, which in Weka is implemented following an instance based algorithm [12]. kNN is a non-parametric algorithm that uses distance measurements to find the closest neighbors to a given sample and, using a majority vote, find the class to which the given samples belongs to. In order to perform this process, the kNN algorithm has different parameters which can be tuned to achieve a better performance. The main parameters are the number of neighbors and the distance calculation method.

The number of neighbors is usually represented by the letter k and it represents how many neighbors should be computed for the classification of a given sample. Obtaining the optimum number of neighbors is crucial because having a small k may imply producing and overfitted algorithm, as it would not select enough neighbors to produce a proper classification, and having a high k could make the algorithm to not adjust well to reality.

With respect to the distance calculation method, there are different approaches but

the most commonly used, and the one used in this thesis, is the euclidean distance. This distance is used to obtained the k nearest neighbors, those having the smallest distance to the sample, and it could be weighted or not depending on the approach. This weight allows the algorithm to give the closest neighbors more importance, as the sample is most likely to belong to that neighbor's class. The euclidean distance is computed as follows:

$$d(x^{(i)}, x^*) = ||x^{(i)} - x^*||^2$$

In order to optimize the IBk algorithm, different trials were performed changing the parameters previously explained with the aim of obtaining the highest accuracy and the lowest Root Mean Square Error(RMSE):

- Using IBk with euclidean distance and distance weighting. 2.7

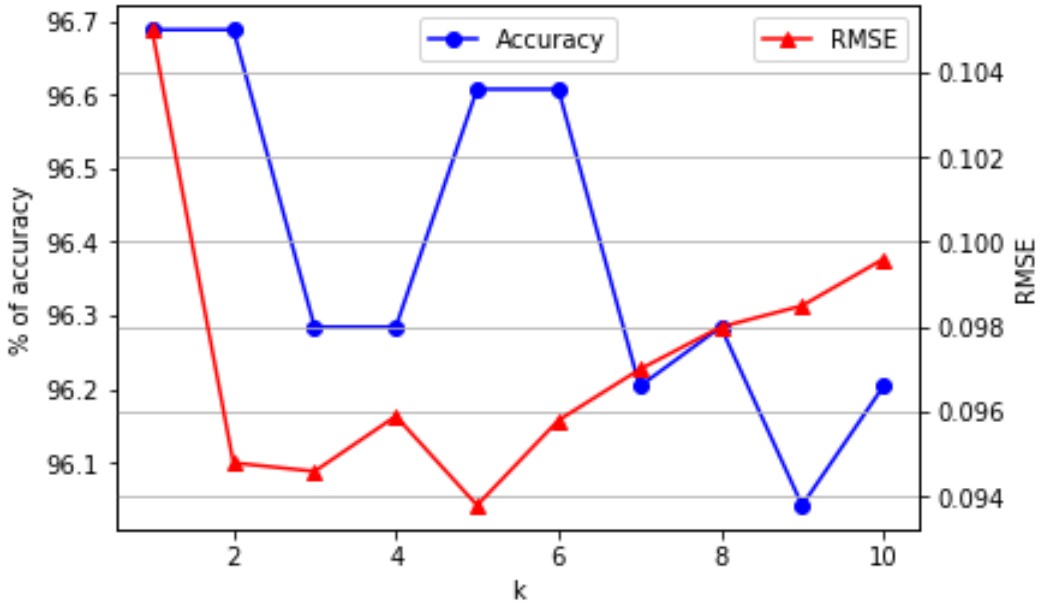


Fig. 2.7. Accuracy and RMSE of IBk with respect to the number of neighbors using distance weighting.

- Using IBk with euclidean distance and no distance weighting. 2.8

As it can be seen in figures 2.7 and 2.8, the best results in accuracy are obtained for $k = 1$, but it was considered not a good result because, as it was stated before, having a low value for k could imply that the model is overfitted. In fact, for $k = 1$ the RMSE gets its highest

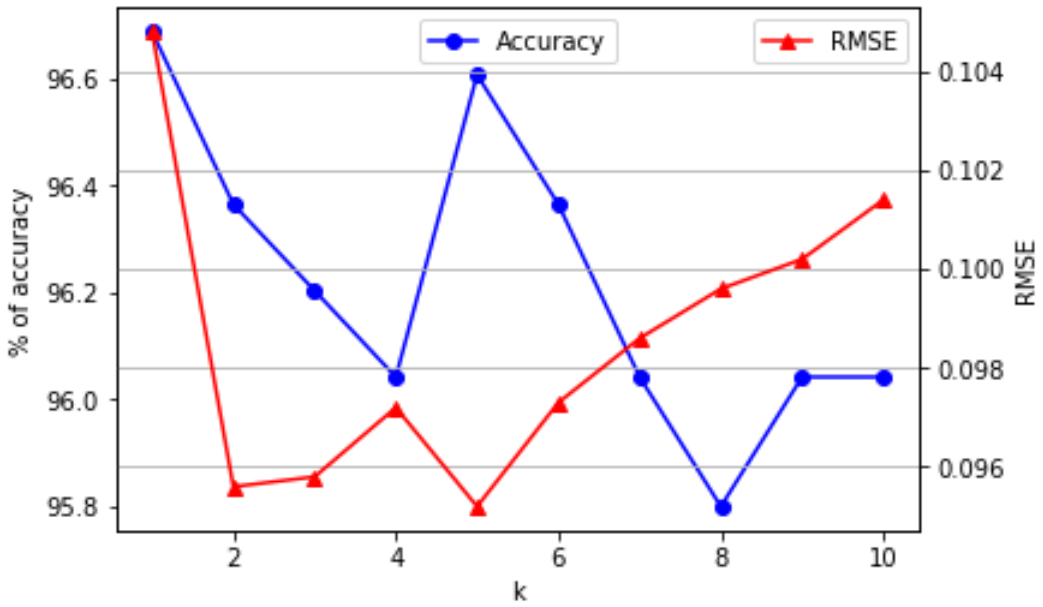


Fig. 2.8. Accuracy and RMSE of IBk with respect to the number of neighbors without distance weighting.

value. For that reason, the next maximum was taken and it is found for $k = 5$ in both cases. At this point, it is also reached the minimum for the RMSE which is a clear indicator that the optimum parameters for this model have been found. It is noticeable also how the performance gets worse for bigger values of k, due to the model not being able to fit to a single class, so there is no need to keep trying new values as it is unlikely for the model to improve its performance.

With respect to the differences observed between weighted and no weighted distance, for the optimum k, there is almost no difference on the values obtained, being the accuracy and RMSE for the weighted case 96.6074 % and 0.0938 respectively, and 96.6074 % and 0.0952 for the non-weighted case. As it can be seen they only differ in the RMSE by 0.0014, which is practically negligible. Despite of this similarity, it was decided to use weighted distances as it gives priority to closest neighbors.

2.5.2. Multilayer Perceptron

A multilayer perceptron is a type of neural network that complies with the following properties [13]:

- Each neuron or node in the network includes a differentiable activation function.
- The network is composed of a set of *n* hidden layers.
- It is a fully connected network, that is, each neuron in a layer is connected to all the neurons in a previous layer.
- The network progresses in a forward direction, in a layer after layer base and from left to right.
- For updating the weights and biases of each neuron, the back-propagation algorithm is used.
- The activation function for each neuron is a sigmoid function.

The final structure of a multilayer perceptron could be seen in figure 2.9.

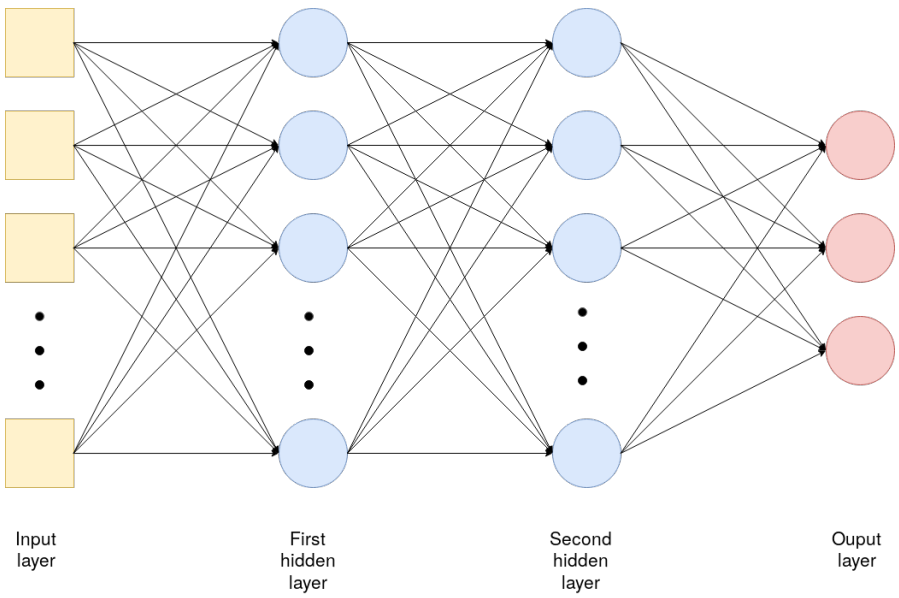


Fig. 2.9. Multilayer perceptron structure.

One of the main parameters of MLP is the learning rate, which indicates the variability of the weights of the neurons when adjusting them. Setting this parameter to its optimum value allows to train the network more efficiently. If the learning rate is too high, the algorithm will not able be to find a global minima and will be stacked at a local minima while setting it to a low value will allow to find the global minima but with a large training time,

which is not optimum. In this optimization process, thanks to Weka, the learning rate will be decreased automatically all along the training process so it adapts efficiently at any moment.

Another important parameter is the number of epochs, which indicates the number of times the whole dataset is passed through the network in the training process. Finding the optimum number of epochs will lead to avoid underfitting as well as overfitting. It will also affect to the training time.

For the optimization of the neural network, the number of hidden layers, the number of epochs and the learning rate (automatically with Weka) are going to be tuned in order to achieve the maximum accuracy in the classification and the minimum RMSE. A result of this process can be seen in figure 2.10.

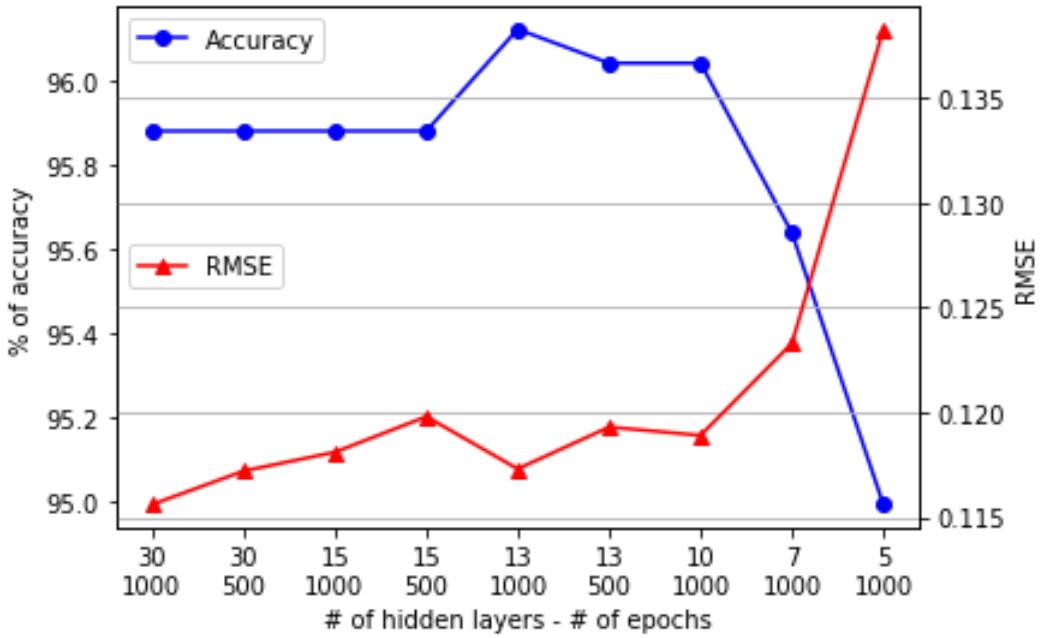


Fig. 2.10. Accuracy and RMSE of MLP with respect to the number of hidden layers and the number of epochs.

As it can be observed in figure 2.10, the impact of the number of epochs is visible in the RMSE and it could be said that it has a negligible effect as for the same number of hidden layers it changes a maximum of 0.117. On the other hand, varying the number of

hidden layers can make a difference as it can be appreciated when changing from 15 to 13 layers; the accuracy changes from 95.8805 % to 96.1228 % while the RMSE drops from 0.1181 to 0.1173.

It could be concluded that the optimum model is the one conformed of 13 hidden layer plus input and output layers, a decreasing learning rate starting on 0.3 and 1000 epochs.

2.6. Mobile application development

In this section the process for the development of the final application is going to be explained as well as how the different parts composing it have been structured and planned. The program was written using Java because, as it is mention in section 1.2.2, it is one of the most important programming languages nowadays but mainly because Weka is based also in Java, which allows to import all the classification models previously designed to the application just adding Weka as a library to the project.

At the end, the program should be able to detect whether the user is suffering a physical aggression and if so send a notification via email to a selected contact. Moreover, it should be able to detect additional movement patterns such as walking or running.

2.6.1. Global functioning

In figure 2.11, it is shown a flowchart explaining the working steps of the application. After initialization, the program will start taking measurements until it has saved a total of 75, which amounts for a total of 1.5 seconds as one measurement is taken every 20 ms. Once a complete sample has been saved, it is passed to the feature extractor and then it is classified to decide if the taken sample corresponds to an aggression or not. At this point, the sample measurements are deleted so it can accept new ones again. If the classification yields false[9], the whole process is restarted. If the result is true, it will stop collecting samples and will send the alert notification.

[9]Being false that no aggression has been detected and true the opposite.

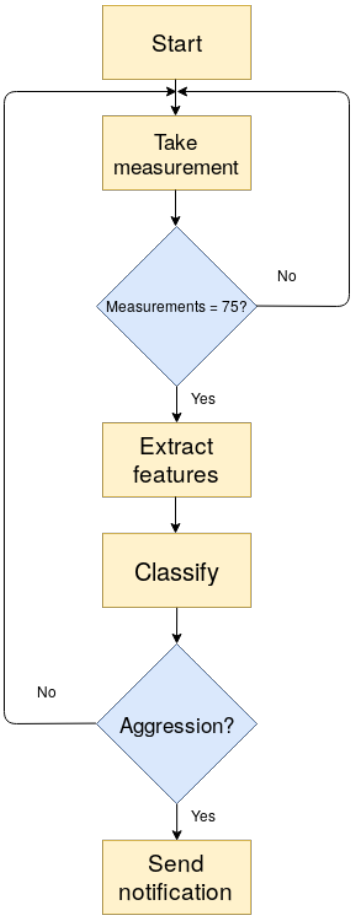


Fig. 2.11. Global functioning of the application.

Regarding some technical aspects of the program, the main problem was that the application stopped whenever the screen turned off or the application passed to a second plane. This is a problem as the idea is to keep the program working for a long time and keeping the screen on during all of it is very inefficient. To solve this problem WakeLocks were used as they allow to prevent the application from stopping when it is not in first plane. After some trials, it was shown that acquiring a WakeLock was very inefficient as they are designed for very specific moment, and keeping a program running for a long time makes use of an excessive amount of battery.

Apart from that, whenever the process of classification was started, the screen froze due to the device having to do a high number of computations. In order to solve this problem, it was considered the use of threads. This allows to execute part of the functionality in the background so the main thread does not get congested. After some trials it was not only shown that the application worked without freezing at all but also that the thread

allows to keep the program working when the screen is off or in a second plane.

On the other hand, it was initially considered that the collection of samples and the classification would be in different threads so while a sample is being classified the application can continue gathering a new sample at the same time so no information about the movement is lost. But after testing the classification process it was seen that it only takes 4 milliseconds to perform the whole process of the classification, including the extraction of features. This time is only a 0.2667 % of the 1.5 seconds that takes a whole sample, so it was concluded that, for the reason of making the program easier, the processes of taking data and classifying could be consecutive, that is, one cannot start until the other has finished.

With respect to the classification process, Weka allows to save a trained model so it can be used later. Using this, the MLP and IBk models studied before were added to the Android project to be in charge of the classification process. Both of them worked as expected.

2.6.2. Classes distribution and functions

The structure of the application is composed of different classes, each one of them having a set of attributes and functions that allow them to perform a unique role in the global performance. Its main characteristics are explained in subsequent lines.

- **Main.java:** this class is in charge of unifying all the processes describe in section 2.6.1 to follow the flowchart in figure 2.11. Additionally, it also controls the initialization of the application as well as all of its lifecycle states (see Annex D for more information on Android activity's lifecycle). It also starts the graphical user interface whenever the program is opened and loads the model for the classifier algorithm in memory for a later use.

  This class is also in charge of the process controlling the operation of the sensors and the collection of the measurements. As it can be appreciated in figure 2.12,

changes in the accelerometer and gyroscope values are monitored and every 20 milliseconds these new values are stored with the aim of constructing a new sample. Remark that these new sensor values are only stored only if the sample does not have 75 measurements, which corresponds to 1.5 seconds.

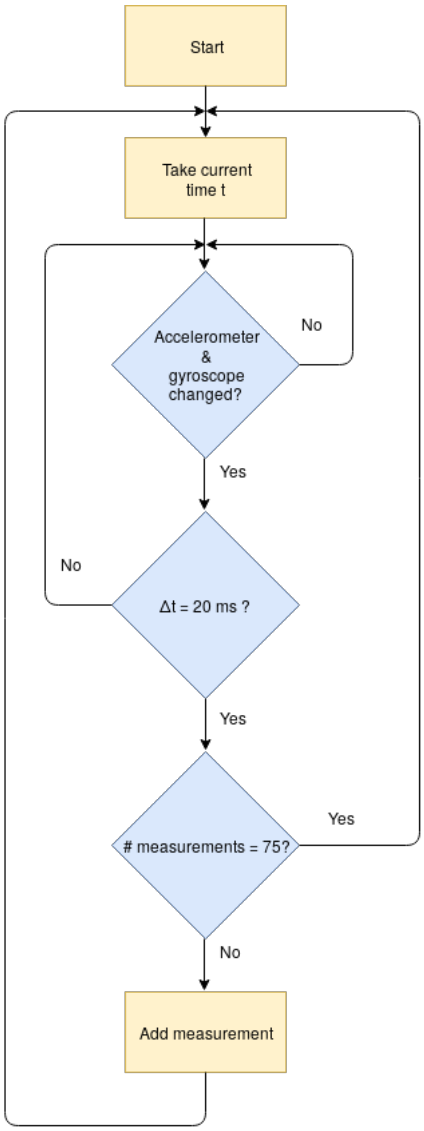


Fig. 2.12. Flowchart for the operation of the accelerometer and gyroscope sensors.

The last important functionality of this class is the process regarding the decision taking, that is, deciding if the user is under an aggression or not. For that, a buffer with six positions is filled with the classification outputs. Once the buffer is full, it checks whether the number of classifications corresponding to an aggression are bigger or equal to 4 and if it yields true, start the process of sending the notification.

In the case the output is false, it will repeat the process adding a new sample and deleting the oldest one each iteration so the buffer keeps full at all times. To sum up, an aggression will be detected if the user is suffering it for at least 6 out of 9 seconds. A graphical explanation of this process can be seen in figure 2.13.

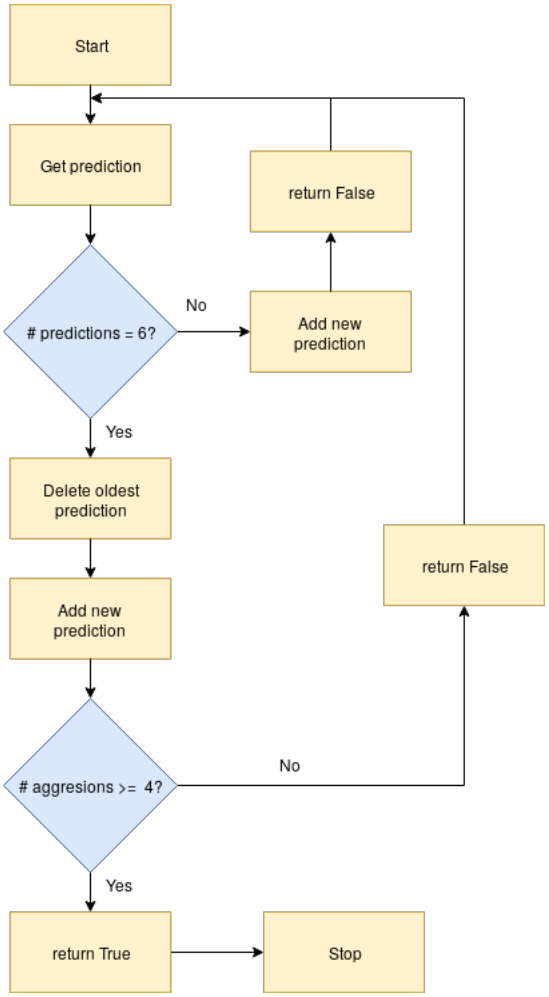


Fig. 2.13. Flowchart for the operation of the aggression decision process.

- **Measurement.java:** this class is in charge of saving each one of the measurements with its corresponding axis values.
- **Sample.java:** this class has as attributes the different features that compose the dataset that is passed to the classifiers and as functions the necessary processes to compute the features from a given sample. These features are calculated in a sequential order calling the corresponding methods when necessary.
- **Classifier:** this class is in charge of controlling the classifier. First of all, it makes possible the transformation of the input features into instances, which is the variable

type accepted by Weka's classifiers and contains the name of each feature as well as its value along with the class label. Once the corresponding instance is obtained, it is passed to the method *classifyInstance()* that returns the predicted class.

- **SendAlert.java:** this class is in charge of the email notifications. It makes use of the AsynTask functionality to, once an aggression has been detected, send an email alert to a previously established emergency contact without the user's interaction. For that, with the help of the auxiliary classes *Mail.java* and *SMTPAuthenticator.java*, the application authenticates the user with the given email address and password and sends the alert to the destination address whenever necessary using SMTP[10].

2.6.3. Graphical user interface

For the graphical user interface, it was decided to designed it as simple as possible so the application is easy to understand and fast to use. With that aim, it was constructed using only the necessary elements to enable each one of the functions without putting too much attention into obtaining a visually attracting interface.

In figure 2.14, it is shown a representation of the final interface as would be seen by the user. At the top of the screen there is a welcoming message displaying the name of the application and right after three text fields where the user should enter its personal email address and password as well the email address that will be used as the emergency contact. The *"SAVE EMAIL"* button allows to store this informatios into memory.

With respect to the initialization, after pressing the *"START"* button the application will begin collecting data from the user's movement and make classifications. It will not stop until the *"STOP"* button is pressed; at this point the model stops making classifications and the sensors are turned off.

At the bottom of the screen is constantly displayed a message indicating whether the user is under a physical aggression or not.

[10]SMTP stands for Standard Mail Transfer Protocol.

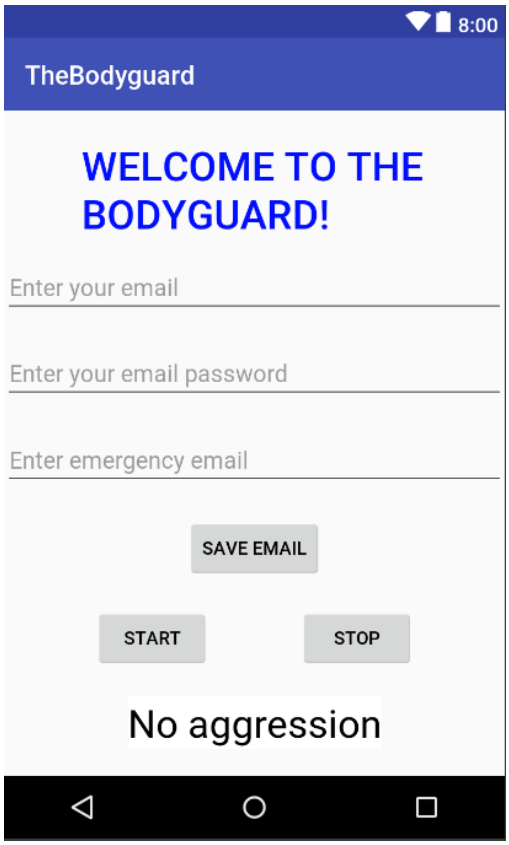


Fig. 2.14. Graphical user interface of the application.

2.6.4. Efficiency

Regarding the execution times, it will be analyze if the different processes running complies with the specified times. As can be seen in figure 2.15, the initialization process takes around 0.267 seconds, being 0.107 seconds of them corresponding to loading the MLP model into memory. It is considered that this time is sufficiently small in order to obtain a good user experience.

With respect to the classification processes, in table 2.9 and figure 2.16 it can be seen an itemization of the different processes involving the classification with their corresponding execution times. Each classification round takes 52 milliseconds approximately and in the case an aggression is detected, 3.03 seconds would be added to send the notification. The process *Alert sending* starts right when the aggression is detected and finish when the email is sent, and it involves processes such as the user authentication and the email composing.

```
06-14 11:44:18.207 10525-10525/com.example.daniel.thebodyguard D/AccessibilityManager: current package=com.example.daniel.thebodyguard,
accessibility manager mIsFinalEnabled=false, mOptimizeEnabled=true, mIsUiAutomationEnabled=false, mIsInterestedPackage=false
06-14 11:44:18.273 10525-10525/com.example.daniel.thebodyguard D/Classifier: Loading model
06-14 11:44:18.380 10525-10525/com.example.daniel.thebodyguard D/Classifier: Model loaded
06-14 11:44:18.464 10525-10545/com.example.daniel.thebodyguard I/Adreno: QUALCOMM build                   : dd15ef5, Ic280a69317
                                                                         Build Date                     : 05/09/17
                                                                         OpenGL ES Shader Compiler Version: XE031.09.00.04
                                                                         Local Branch                   :
                                                                         Remote Branch                  : quic/gfx-adreno.lnx.1.0.r5-rel
                                                                         Remote Branch                  : NONE
                                                                         Reconstruct Branch             : NOTHING
06-14 11:44:18.474 10525-10545/com.example.daniel.thebodyguard I/OpenGLRenderer: Initialized EGL, version 1.4
06-14 11:44:18.474 10525-10545/com.example.daniel.thebodyguard D/OpenGLRenderer: Swap behavior 1
```

Fig. 2.15. Initialization logs.

Task	Execution time (s)
Feature extraction	0.038
Sample classification	0.014
Variable reset	≈ 0
Alert sending	3.03

TABLE 2.9. EXECUTION TIME FOR CLASSIFICATION TASKS

```
06-14 11:37:40.494 9916-10080/com.example.daniel.thebodyguard D/BodyguardApp: Start feature extraction
06-14 11:37:40.532 9916-10080/com.example.daniel.thebodyguard D/BodyguardApp: Feature extraction done
06-14 11:37:40.532 9916-10080/com.example.daniel.thebodyguard D/BodyguardApp: Start classification
06-14 11:37:40.546 9916-10080/com.example.daniel.thebodyguard D/BodyguardApp: Classification done
06-14 11:37:40.546 9916-10080/com.example.daniel.thebodyguard D/BodyguardApp: WalkUp false
06-14 11:37:40.546 9916-10080/com.example.daniel.thebodyguard D/BodyguardApp: Reset of variables done
06-14 11:37:42.053 9916-10080/com.example.daniel.thebodyguard D/BodyguardApp: Start feature extraction
```

Fig. 2.16. Classification logs.

Also in figure 2.16, focusing on the last two lines of the logs, it can be seen that the time between the reset of variables, which corresponds to the end of the classification process, and the start of a new feature extraction is approximately 1.507 seconds, which corresponds to a whole new sample.

Regarding the usage of battery, CPU and memory, in figures 2.17 and 2.18 it is shown how the application makes use of the available resources along time. It uses around 17 %

of CPU at all times and approximately 28.68 MB of memory, which allows the application to run smoothly and without freezing and the mobile device not to overheat. With respect to battery usage, one hour of continued use drains around 5 % of the battery.

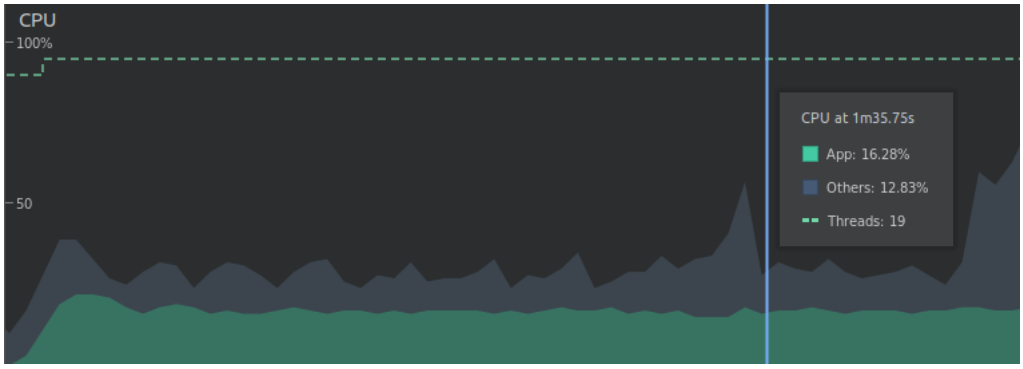


Fig. 2.17. CPU usage.

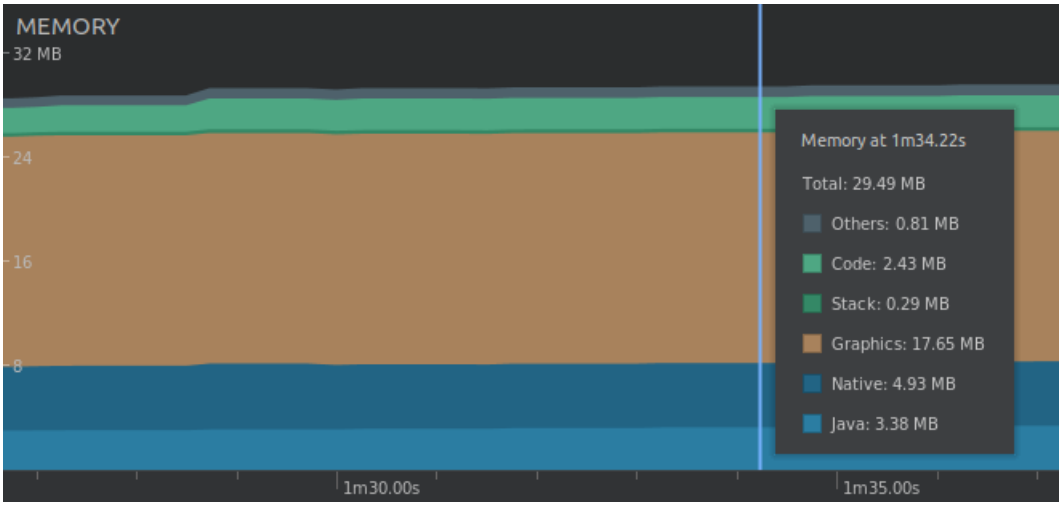


Fig. 2.18. Memory usage.

3. REGULATORY FRAMEWORK

In this section, it will be analyzed an aspect of the application related with the user's data protection and security so it complies with the regulatory laws.

According to the General Data Protection Regulation (GDPR), a company that has access to its user's data has to implement different process that guarantee the protection of this data as well as the user is properly informed about the usage that is being made with his/her information.

For that, as the user is giving the email account private information, at the initialization of the application, a message will be shown informing the user that none of his/her data will be used for any other application or provided to third-parties but only used to send the alert notification via email.

Additionally, in case the application is improved to be use in conjunction with an online server, different processes will be implemented in order to constantly monitor the application looking for security breaches that could violate the user's data protection. In case a security breach appears, all user will be notified in 72 hours at maximum as stated by the GDPR.

With respect to the software used, both Weka and Android Studio are Open Source projects so there is no limitation for their use in this project. Regarding Matlab usage, a license obtained through the University Carlos III was used.

4. SOCIO-ECONOMIC IMPACT

Regarding the budget for the development of the application, in table 4.1 are presented the main expenses.

Concept	**Price (€)**
Salaries	45180
Internet connection	288
Computers	3000
Training data collection	250
Total	48718

TABLE 4.1. EXPENSES FOR THE DEVELOPMENT OF THE APPLICATION.

The salaries involved the individual salaries of two engineers receiving 3765€/month[11] for 6 months. The training data collection refers to what was paid to the five subjects that participate in the data collection at the beginning of the project.

With respect to the economic impact, it is expected that the application could be launched to the public as soon it is ready. Its incomes are planned to come from the following sources:

- **Adds version:** in this version, the incomes will come from the user's clicks in the adds displayed. It is estimated that each click will produce 0.02€. This version will be free to download.
- **No-Adds version:** the incomes in this version will come from a downloading price, which will be set at 6€.

[11] Approximate salary for a telecommunications engineer as stated by COIT. Source: `https://www.coit.es/noticias/pleno-empleo-para-los-ingenieros-de-telecomunicacion`

- **Premium version:** any user will be able to upgrade to the premium version at any moment. It will have a cost of 8€/month and will provide a security service that will attend immediately each user if he/she suffers an aggression. This security service will produce an expenses of 480000€/year as there will be needed 20 people with salaries of 2000€/month.

In table 4.2, the estimated incomes for a year for each of the versions are shown. These estimations are calculated with 10000 active users.

Version	**Revenues (€/year)**
Adds version	73000
No-Adds version	60000
Premium version	480000

TABLE 4.2. INCOMES OF THE APPLICATION IN THE MARKET

In the case that online servers are implemented, to the expenses described in table 4.1 will be added the cost of maintaining the servers as well as the data storage, which will vary depending on the company providing this service.

Regarding the social impact, it is expected that the application will have a high influence as it will allow people to feel more protected. The generalized use of the application could also imply a fastest intervention by part of the police and emergency services, which in the end could imply an increment in the safety as well as a reduction in the criminality.

5. RESULTS

For the obtainment of the results, different trials have been performed for both the MLP and IBk classifiers. For that, an auxiliary functionality has been added to the program that allows to record the classifications made for each instant of time so later these classifications can be compared with the real movement that was being performed.

Two different approaches are shown. First, results related with the classification taking into account all the movement patterns considered. Second, results related with the binary classification problem, presented in section 2.3, which only focus on whether the user is suffering and aggression or not.

Finally, some results regarding the usage of the application will be presented. It will be shown the ability to send the alert notification at the right moment.

5.1. Multilayer Perceptron

In table 5.1, it can be seen the confusion matrix for the test trials. From here, it is obtained an accuracy of 67.22 %.

classified as –>	**a**	**b**	**c**	**d**	**e**	**f**
a = Walking	**18**	0	14	1	0	2
b = Running	8	**19**	0	0	0	5
c = WalkUp	0	0	**18**	8	5	0
d = WalkDown	0	0	6	**19**	4	0
e = Floor	0	0	0	0	**26**	0
f = AgrStand	0	0	6	0	0	**21**

TABLE 5.1. CONFUSION MATRIX FOR TEST MLP CLASSIFIER

In table 5.2, it is presented the confusion matrix for the binary problem. From it, the

accuracy obtained is 87.78 %.

classified as –>	**a**	**b**
a = Agresion	**111**	16
b = NotAgresion	6	**47**

TABLE 5.2. CONFUSION MATRIX FOR TEST MLP BINARY CLASSIFIER

5.2. K Nearest Neighbors

In table 5.3, it can be seen the confusion matrix for the test trials. From here, it is obtained an accuracy of 91.11 %.

classified as –>	**a**	**b**	**c**	**d**	**e**	**f**
a = Walking	**32**	0	2	0	0	1
b = Running	3	**28**	0	0	0	1
c = WalkUp	0	0	**26**	4	1	0
d = WalkDown	0	0	2	**27**	0	0
e = Floor	0	0	0	0	**26**	0
f = AgrStand	2	0	6	0	0	**25**

TABLE 5.3. CONFUSION MATRIX FOR TEST MLP CLASSIFIER

In table 5.4, it is presented the confusion matrix for the binary problem. From it, the accuracy obtained is 97.22 %.

classified as –>	**a**	**b**
a = Agresion	**124**	3
b = NotAgresion	2	**51**

TABLE 5.4. CONFUSION MATRIX FOR TEST MLP BINARY CLASSIFIER

5.3. Android application

Here are presented the results for the performance of the application at the time of sending the alert notification. They are shown in the form of a confusion matrix where *SendAlert* stands for the need of sending an alert as the user is under an aggression and *NoSendAlert* refers that there is no need of sending the alert. It is achieved an accuracy of 98.058 %.

classified as –>	**a**	**b**
a = SendAlert	**37**	0
b = NoSendAlert	2	**64**

TABLE 5.5. CONFUSION MATRIX FOR THE ALERT NOTIFICATIONS

6. CONCLUSIONS

The main objective of this thesis was the development of an Android application capable of detecting, using the accelerometer and gyroscope sensors available in a mobile device, whether the user is under a physical aggression or not. Taking a look at the results presented in chapter 5, more specifically tables 5.2 and 5.4, it is concluded that this objective has been fulfilled as accuracies of 87.78 % and 97.22 % are achieved for MLP and IBk respectively.

In fact, due to the buffer implemented in the application by which the alert is sent only when 4 out of 6 classifications corresponds to an aggression, the false positives are eliminated and thus the performance is improved to 98.058 % as seen in table 5.5.

Regarding the classification with respect to the different movement patterns, according to the results shown in tables 5.1 and 5.3, it is possible to observe that the performance has worsened drastically for the MLP model, which obtains an accuracy of only 67.22 % and shows poor differentiation between classes with similar movement patterns such as *Walking* and *WalkUp*, while IBk just falls to 91.11 %. This situation may be given for different reasons: one could be that the network has not been fully optimized and it is overfitted, so it cannot generalize correctly; another one could be that the network is too dependent of the device position at the time of gathering samples due to the use of the linear accelerometer, which makes an estimation of the gravity force applied and consequently could yield some error.

Either way, IBk model seems to work fine as it is capable of properly differentiate between classes properly so it could be concluded that this objective was also achieved.

With respect to the final application, as shown in table 5.5, it is capable of efficiently send an alert notification whenever an aggression is detected without taking into account false positives. Throughout the final trial it could also be observed that the application

worked according to the measured execution times stated in section 2.6.4 and without freezing at all.

To sum up, it has been developed an Android application capable of detecting whether the user is suffering a physical aggression or not with a sufficiently high accuracy and fast enough to be viable to use. What is more, it has been demonstrated that it could also be used to monitor alternative movement patterns.

7. FUTURE WORK

In this chapter, some future lines of work that could help to improve the results obtained in the thesis will be stated:

- In chapter 6 it is said that the problem for the MLP model to not be working properly could be the high dependency of the samples with the position of the device. To solve it, the training data collection will be performed in a wider range of subjects so it is more general as well as the total number of samples will be increased. Apart from that, the use of the linear accelerometer will be deprecated in favor of a manual approximation of the force of gravity over each axis so the estimation error is reduced to the minimum.

- It will be studied the possibility of increasing the number of movement patterns so the program could be used with additional applications such as physical activity monitoring.

- Regarding the features composing the dataset, they will be studied more in depth to discover new relationships between them and the classes so they can be reduced as much as possible, which will imply again an improvement in the classification process.

- Study a functionality through which the user can record its own movement patterns in the first use of the application so the model can be retrained with the aim of individualizing the classification algorithm. This could be achieved by establishing a communication with an online server in charge of training the model.

- With respect to the email process, it will be tried to increase the security involve in all of them by establishing an initial login in charge of synchronizing automatically with the user's Google account so he/she does not have to enter the password each time the application is initiated.

BIBLIOGRAFÍA

[1] Statista. (). Number of mobile phone users worldwide from 2013 to 2019, [Online]. Available at: https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/. (accessed: 09.06.2018).

[2] J. Wang et al., "Recognizing Human Daily Activities From Accelerometer Signal", *Procedia Engineering*, n.º 15, pp. 1780-1786, 2011.

[3] J. R. Kwapisz, G. M. Weiss y S. A. Moore, "Activity Recognition using Cell Phone Accelerometers", Fordham University, 2010.

[4] N. Ravi et al., "Activity Recognition from Accelerometer Data", Rutgers University, 2005.

[5] H. Abdi y L. J. Williams, "Principal component analysis", *Wiley interdisciplinary reviews: computational statistics*, vol. 2, n.º 4, pp. 433-459, 2010.

[6] L. Li et al., "Multi-label Feature Selection via Information Gain", en *Advanced Data Mining and Applications*, X. Luo, J. X. Yu y Z. Li, eds., Cham: Springer International Publishing, 2014, pp. 345-355.

[7] U. of Waikato. (). Weka documentation, [Online]. Available at: http://weka.sourceforge.net/doc.stable/overview-summary.html. (accessed: 07.02.2018).

[8] J. Dunn. (). There's no hope of anyone catching up to Android and iOS, [Online]. Available at: http://uk.businessinsider.com/smartphone-market-share-android-ios-windows-blackberry-2016-8?IR=T. (accessed: 09.10.2017).

[9] TIOBE. (). TIOBE index, [Online]. Available at: https://www.tiobe.com/tiobe-index/. (accessed: 09.10.2017).

[10] A. Developers. (). Sensors Overview, [Online]. Available at: https://developer.android.com/guide/topics/sensors/sensors_overview.html. (accessed: 01.09.2017).

[11] R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann Publishers, 1993.

[12] D. W. Aha, D. Kibler y M. K. Albert, “Instance-based learning algorithms”, *Machine Learning*, 1991.

[13] S. Haykin, *Neural Networks and Learning Machines*. Pearson, 2009.

[14] J. Platt, “Fast Training of Support Vector Machines using Sequential Minimal Optimization”, en *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges y A. Smola, eds., MIT Press, 1998.

[15] G. H. John y P. Langley, “Estimating Continuous Distributions in Bayesian Classifiers”, en *Eleventh Conference on Uncertainty in Artificial Intelligence*, San Mateo: Morgan Kaufmann, 1995, pp. 338-345.

[16] N. Japkowicz y M. Shah, *Evaluatin Learning Algorithms. A classification Perspectiva*. Cambridge University Press, 2011.

[17] J. Tang, S. alelyani y H. Liu, “Feature Selection for Classification: A Review”, King Khalid University, 2014.

[18] A. Developers. (). Understanding the activity lifecycle, [Online]. Available at: `https://developer.android.com/guide/components/activities/activity-lifecycle`. (accessed: 15.06.2018).

ANNEX A: TECHNICAL SPECIFICATIONS

Xiaomi Redmi 4X

Android Version	7.1.2 N2G47H
CPU	Snapdragon 435 octa-core, 1.4 GHz max
RAM	3.00 GB
Battery	4100 mAh
Dimensions	139,24 mm x 69,96 mm x 8,65 mm, 146 g

Acer Aspire V3-572G-77S8

OS	Ubuntu 17.10
Processing unit	Intel Core i7-4510U 2.0 GHz
RAM	16 GB DDR3L SDRAM

Software

Software	**Version**
Android Studio	3.0.1
Weka	3.9.2
Matlab	R2018a 9.4.0.813654

ANNEX B: PCA TRANSFORMATIONS

PC	Linear combination of original features
1	$0{,}219 rmsAccZ + 0{,}219 stdAccZ + 0{,}219 AbsDiffAccZ + 0{,}219 AvgAcc$
2	$0{,}333 CorrGyrXY + 0{,}285 meanAccZ + 0{,}241 Bin3AccY - 0{,}221 CorrGyrZY.$
3	$0{,}319 Bin5GyrX - 0{,}242 meanAccZ + 0{,}239 Bin2AccX + 0{,}209 Bin1AccX$
4	$0{,}336 Bin4AccZ - 0{,}301 Bin2AccZ + 0{,}278 Bin5AccZ - 0{,}26 Bin1AccZ$
5	$-0{,}354 Bin4AccZ - 0{,}335 Bin4GyrY + 0{,}299 Bin2AccZ - 0{,}274 Bin5AccZ$
6	$0{,}364 Bin2GyrY - 0{,}335 Bin4GyrY - 0{,}302 Bin5GyrY - 0{,}265 Bin2AccY$
7	$-0{,}312 Bin1GyrX + 0{,}301 Bin3AccZ - 0{,}3 Bin4AccY - 0{,}29 Bin2GyrX$
8	$-0{,}469 Bin4GyrZ + 0{,}319 Bin1GyrZ + 0{,}306 Bin2GyrZ - 0{,}275 Bin4GyrX$
9	$0{,}388 CorrGyrXZ + 0{,}317 CorrGyrZY + 0{,}291 meanGyrZ - 0{,}267 CorrAccXZ$
10	$0{,}336 Bin1AccZ + 0{,}306 meanGyrZ - 0{,}289 Bin3GyrY + 0{,}266 Bin2AccZ$
11	$0{,}468 Bin3GyrX - 0{,}363 Bin3AccY + 0{,}28 meanGyrY - 0{,}279 Bin1GyrY$
12	$-0{,}378 Bin1GyrZ + 0{,}33 Bin3GyrY - 0{,}322 meanGyrY + 0{,}301 Bin3GyrZ$
13	$0{,}445 Bin3AccZ - 0{,}356 Bin3AccX - 0{,}33 Bin1AccZ + 0{,}247 CorrAccXZ$
14	$-0{,}522 Bin3GyrZ + 0{,}455 Bin3GyrY + 0{,}299 Bin2GyrZ + 0{,}24 Bin1GyrZ$
15	$-0{,}595 meanGyrX - 0{,}286 CorrGyrXZ + 0{,}247 meanGyrY - 0{,}226 CorrGyrZY$
16	$0{,}506 Bin3AccX - 0{,}474 Bin1AccX - 0{,}245 Bin3AccY + 0{,}217 Bin5GyrZ$
17	$0{,}378 CorrAccXZ - 0{,}328 Bin5AccX - 0{,}291 Bin4AccX - 0{,}29 meanGyrY$
18	$-0{,}447 Bin3GyrX + 0{,}314 CorrAccZY - 0{,}272 Bin3AccY + 0{,}268 CorrAccXY$
19	$0{,}476 meanAccY + 0{,}37 Bin3AccY - 0{,}316 Bin4AccY + 0{,}288 Bin5GyrZ$
20	$-0{,}498 Bin2GyrZ + 0{,}472 Bin1GyrZ - 0{,}222 Bin5GyrZ + 0{,}212 Bin4GyrZ$
21	$-0{,}431 Bin1GyrX + 0{,}355 Bin2GyrX - 0{,}282 Bin1AccY + 0{,}27 meanAccX$

ANNEX C: INFORMATION GAIN SELECTED FEATURES

Position	Information Gain	Feature
1	1.798	AvgAcc
2	1.749	stdAccY
3	1.749	rmsAccY
4	1.708	AbsDiffAccY
5	1.584	AngVel
6	1.562	AbsDiffAccZ
7	1.561	AbsDiffGyrY
8	1.529	stdGyrY
9	1.488	rmsAccZ
10	1.487	stdAccZ
11	1.401	rmsGyrY
12	1.395	rmsAccX
13	1.39	stdAccX
14	1.387	rmsGyrZ
15	1.383	AbsDiffGyrZ
16	1.381	stdGyrZ
17	1.329	stdGyrX
18	1.317	AbsDiffAccX
19	1.27	AbsDiffGyrX
20	1.268	rmsGyrX
21	1.16	meanAccZ
22	0.968	meanAccX
23	0.838	meanGyrY
24	0.812	meanAccY

ANNEX D: LIFECYCLE OF AN ANDROID ACTIVITY

An Android activity is a component of an application that is in charge of controlling a certain screen as well as all the interactions the user makes in that screen. To monitor this activity, there are different methods available that allow to control all of its life states [18].

- **onCreate():** receives a call whenever the application is started for the first time. All necessary variables and previously states, if saved, are initialized here.
- **onStart():** the activity is now visible for the user and ready to interact with him/her. The activity is at the top of the stack of activities.
- **onStop():** the activity is no longer visible for the user. The activity has been destroyed or another activity was restarted.
- **onDestroy():** this function is called right before the activity is destroyed. It could happened because the user finished the application or it was automatically destroyed to save memory space.