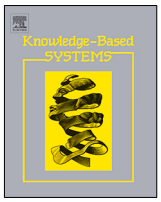
Knowledge-Based Systems 262 (2023) 110260

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



New machine learning approaches for real-life human activity recognition using smartphone sensor-based data

Daniel Garcia-Gonzalez *, Daniel Rivero, Enrique Fernandez-Blanco, Miguel R. Luaces

Department of Computer Science and Information Technologies, University of A Coruna, CITIC, 15071 A Coruna, Spain

ARTICLE INFO

Article history:
Received 7 April 2021
Received in revised form 4 June 2021
Accepted 2 January 2023
Available online 5 January 2023

Keywords:
HAR
Human activity recognition
Machine learning
Real life
Smartphones
Sensors

ABSTRACT

In recent years, mainly due to the application of smartphones in this area, research in human activity recognition (HAR) has shown a continuous and steady growth. Thanks to its wide range of sensors, its size, its ease of use, its low price and its applicability in many other fields, it is a highly attractive option for researchers. However, the vast majority of studies carried out so far focus on laboratory settings, outside of a real-life environment. In this work, unlike in other papers, progress was sought on the latter point. To do so, a dataset already published for this purpose was used. This dataset was collected using the sensors of the smartphones of different individuals in their daily life, with almost total freedom. To exploit these data, numerous experiments were carried out with various machine learning techniques and each of them with different hyperparameters. These experiments proved that, in this case, tree-based models, such as Random Forest, outperform the rest. The final result shows an enormous improvement in the accuracy of the best model found to date for this purpose, from 74.39% to 92.97%.

1. Introduction

Having become a hot research topic in recent years, human activity recognition (HAR) analyses series of sensor-collected data to identify the actions taken by a person [1–3]. These sensors can be used through wearable devices such as wristbands or, more recently, smartphones. Both cases offer a broad set of sensors that can be used relatively easy, with excellent accuracy and a small size that favours its portability. In addition, this area has many application possibilities in various fields such as health, fitness or even home automation [4–8]. All this, together with the recent application of smartphones in HAR and its global use, make this field a highly attractive option for research [9,10].

There are several research challenges within this field. Firstly, there is the challenge of correctly processing the vast amounts of data that these devices collect, while controlling the temporality of such data. Also, although significant advances have been made [11,12], the relation between these data and most human movements is still not known precisely, making the task even more difficult. Besides, most of the studies carried out to date were done in a laboratory environment, with highly controlled movements and specific placements of the device that collects the data [13,14]. That is interesting in order to see what the approximate relation between the information collected and the action studied is. However, the excellent results seen in these works may not be as good when they are applied outside that highly-controlled environment. That is because, in a daily life environment, people will use and carry the data collection device differently, outside of what was previously examined. In this way, the orientation and placement of the device could vary greatly, even when performing the same action. Also, each person may have many physical peculiarities that could considerably influence the final result as well. In fact, the personalization of AI models in HAR for large numbers of people is something that it is being researched since almost a decade [15–18]. For these reasons, the transfer of this knowledge to real-life remains to be seen.

In this paper, looking to close the gap with the real-life application, a dataset gathered for this purpose was used [19]. For that goal, the dataset was collected using the sensors of several individuals' smartphones, with almost total freedom. In this way, a comparative study between the last results obtained and the current ones is presented. To do so, numerous machine learning algorithms frequently used in HAR were implemented, in search of the best combination between algorithm and hyperparameters. In the same manner, a comparison of the results obtained by using the data taken with all the sensors, as well as with the absence of the gyroscope, is also presented to observe which case behaves better. In this way, we will be able to get even closer to that real-life ideal that is currently being pursued.

* Corresponding author.
E-mail addresses: d.garcia2@udc.es (D. Garcia-Gonzalez), daniel.rivero@udc.es (D. Rivero), enrique.fernandez@udc.es (E. Fernandez-Blanco), miguel.luaces@udc.es (M.R. Luaces).

https://doi.org/10.1016/j.knosys.2023.110260

Thus, the main contributions of this paper are the following:

- A comparison of the main machine learning algorithms applied in the HAR field, using a dataset taken in a real-life environment, unlike in other studies.
- The addition of the Extreme Gradient Boosting (XGB) algorithm to the comparison, not very explored until today in this knowledge area.
- A study of the best model configurations for long-themed activities (such as driving or jogging), based on the indicated dataset, with changes and additions to the last feature set used.
- A review of the gyroscope's real influence to the final results.
- The improvement of the current approaches in this field, oriented towards real life.

The remaining sections of this paper are organized as follows: Section 2 shows some related works on HAR, Section 3 gives a thorough explanation of how the data was prepared, as well as a brief description of every algorithm and metric used, Section 4 presents and discuss the experimental results obtained on the models we propose, and, finally, Section 5 contains the conclusions and future work lines.

2. Related work

Human activity recognition (HAR) has been studied extensively in recent years and, over the last decade, the continuous flow of works has brought a steady pace of advances. Most of these works were carried out using datasets such as those provided in [20,21], which are two of the most widely used ones in HAR. Both datasets offer a large amount of information about different actions to be exploited, using smartphone sensors such as the accelerometer and the gyroscope. However, for both cases, these data were taken in a laboratory environment. That means that the smartphone was placed in a particular position and the actions performed were highly controlled. An example would be [22], where a comparison is made between different machine learning algorithms, namely, Convolutional Neural Networks (CNN), Random Forest (RF), K-Nearest Neighbours (KNN) and also a feature selection method, Principal Component Analysis (PCA). Among all of them, CNN was the best by far, for which they also contributed different architectures, with several combinations of hyperparameters and the result of each one of them. Besides, they also concluded that with rather large time windows the results did not improve. Likewise, in [23], another CNN model was also proposed for this problem, with slightly better performance. Alternatively, other works such as [12] provided techniques based on deep learning, such as the Deep Belief Network (DBN). Here, after a feature selection process, they also obtain pretty good results, even better than those of the models based in the Support Vector Machine (SVM) algorithm, which proved to be the best to use for the HAR problem. Conversely, research was also done on the selection of features for different machine learning algorithms widely used in HAR. The results of works such as [24,25] showed that the frequency-based parameters are more feasible since they were the ones that showed better results.

However, not all the work relied solely on the accelerometer and gyroscope for its research. Some studies such as [26,27] show high-grade results with the addition of other sensors such as the GPS or the magnetometer. In fact, in these works, they studied long-themed activities such as walking or jogging, which shows the potential of these sensors for this type of actions. On the other hand, in [28], an online SVM model is proposed for nine different smartphone orientations, although all of them are based on leaving the mobile phone in a backpack. They also made a comparison with other methods typically used in HAR, such as KNN, Decision Tree (DT) and Naïve Bayes (NB). All these methods, together with other techniques such as SVM, CNN, Random Forest (RF) and Gradient Boosting (GB), proved to be fully valid in HAR for a reasonable amount of data. At the same time, it also indicates that the application of deep learning techniques could be a very up-and-coming line of research for the HAR field, as some of the best results in practice seem to be obtained with this type of methods.

In the same vein, more recently, research in HAR is focusing more on the application of purely deep learning techniques. One of the first works to apply these techniques was [29], in which a comparison of different architectures for a deep CNN model with other methods widely used in the literature, such as SVM or Multilayer Perceptron (MLP), is presented. Moreover, currently, besides the deep CNN models, much research is being done with models that implement the Long Short-term Memory (LSTM) technique. The main advantage that these implementations have is that they can include information from the past in their training, as well as not needing a previous feature extraction period. However, as a disadvantage, they need a large amount of data to obtain reliable results, as well as requiring an adequate stop criterion to avoid overfitting and underfitting. Some examples of the application of this technique are the works of [30,31], in which excellent results were obtained. Specifically, in [30], a modification of this method was carried out, called Bi-LSTM (bidirectional LSTM), which also manages to learn from the future, throwing accuracies of around 95%. On the other hand, other works have been in charge of comparing in depth the two most used deep learning methods, CNN and LSTM and their variants, in search of the most suitable model for HAR [32,33]. The results show that both techniques have very similar potentials and performances, being probably two of the best options to use for short-themed activities such as sitting or standing. Nonetheless, it seems that some studies suggest that the application of CNNs over LSTMs is favoured, due to its higher speed and its straightforward application [34].

However, despite all the progress mentioned above, all the works share the same problem. That one is no other than the dependency on precise use guidelines for the device to obtain good results. While there are some works such as [35,36] that have addressed this problem, they cannot be considered feasible for real life. In these cases, they obtained good results by transforming the phone's coordinate system to the Earth's coordinate system. In addition, in the case of [35], different models of smartphones were also used, without an apparent drop in the eventual accuracy. In any case, when changing the orientation of the smartphones, the final performance does decrease. Finally, they also do not address the problem of placing the smartphone in different places, like a backpack, and not just in a trouser pocket.

Nonetheless, a very recent work does address this problem rightly [19]. There, a dataset focusing on the application of HAR techniques in real life is presented, which will be used in this paper as well. Also, they did a series of experiments with machine learning techniques, but they are very elementary and could be highly improved. Therefore, this work has as main aim to advance in the resolution of these problems of lack of realism and applicability in real life of all the models developed up to date in HAR. While it is true that the advances made so far are very promising, if those advances were taken into a real-life environment, more than probably they would show a grave detriment on their performance. Hence, with the development of new models, comparing them to each other and building on a much more realistic dataset, it is hoped to surpass the results obtained so far in this transition to a more credible environment.

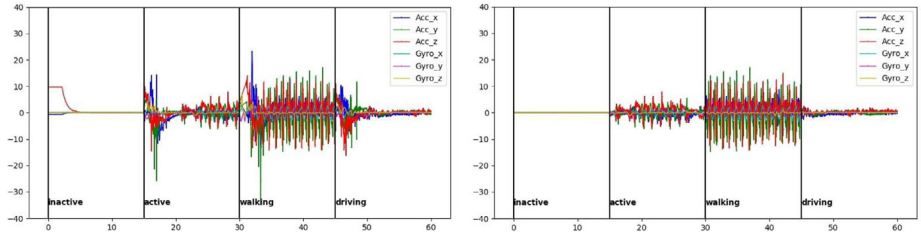


Fig. 1. Examples of data taken by the accelerometer and gyroscope of a specific individual's smartphone, during the first 15 seconds of each session, for each of the specified activities, being: (a) Raw data. (b) Data after having been preprocessed.

3. Experimental setup

This section contains a description of all the resources and methods that were used to carry out this work. Firstly, in Section 3.1, the data preprocessing guidelines are presented, as well as the chosen features for the machine learning models. Then, Section 3.2 gives a brief description of each artificial intelligence algorithm used, as well as introducing their most crucial hyperparameters.

3.1. Data preparation and feature extraction

To carry out this project, the dataset published in [19] was used. In that work, the authors gathered information from four different sensors: accelerometer, gyroscope, magnetometer and GPS. Likewise, it also offers datasets in which the gyroscope does not exist, or neither the gyroscope nor the magnetometer exist simultaneously. The last best results came from the case where the gyroscope data were missing. For this reason, in addition to studying all the sensors, it was decided to study this option as well. In this way, a detailed comparison can be made, in search of the most representative set and the real influence of the gyroscope on the final result. Regarding the activities performed, they were four, as said in that work:

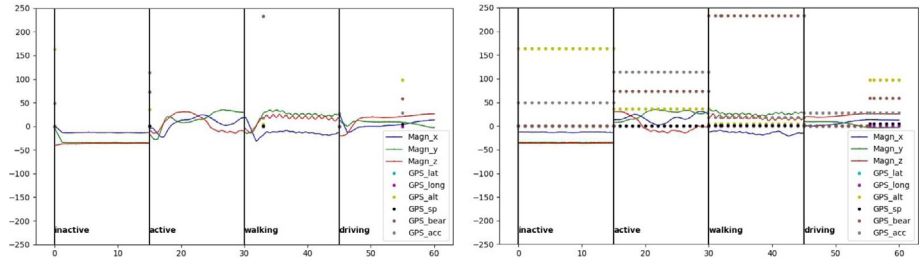
- Inactive: the individual does not have the smartphone on him.
- Active: any action that involves moving, such as cooking or brushing your teeth, but not moving anywhere in particular.
- Walking: any trip made without the use of vehicles, such as walking or running.
- Driving: every kind of journey made utilizing an engine-powered transport, without the need to be the person driving it.

As for the data preparation, it obeyed the same steps as in the original proposal, so the following actions were carried out:

- All outliers in the GPS data that exceeded 0.2 in latitude and longitude increments or 500 in altitude increments were ignored.
- The first and last five seconds of each data collection session were excluded to prevent the model from learning the movements of before and after the activity (putting in or taking the smartphone out of the pocket, for example). Each session corresponds to a whole activity recording, since a person starts an action until they stop it.
- Due to the lack of GPS data in many sessions, they were replicated so that there was always one record per second. For this purpose, if the difference between one observation and another was greater than one second, the last measurement was repeated, with a different timestamp. Similarly, for the same reason, any data session that did not have at least one GPS observation was disregarded.
- Any data session that had long gaps ($>$ 5 seconds) between the accelerometer, gyroscope or magnetometer sensor observations was also ignored.

In order to represent more clearly the arrangement of these data, Figs. 1 and 2 show examples of the data provided by a specific user for each of the indicated activities. In both figures, the image on the left (a) shows the first 15 s of each activity before any preprocessing, while the one on the right (b) shows the result of such preprocessing. The selection of this time interval allows for a straightforward interpretation of the values on a considerable figure size. In order to improve the readability, the figures were divided due to the high variability shown by the average of the values for each sensor. Fig. 1 shows the accelerometer and gyroscope data, while Fig. 2 shows the magnetometer and GPS data. The data for the accelerometer, gyroscope and magnetometer are shown separately for each of their three axes (“Acc”, “Gyro”, and “Magn” on the figures). As for the GPS acronyms, these correspond, on the one hand, to the latitude, longitude and altitude increments between each sample (“GPS_lat”, “GPS_long” and “GPS_alt”, respectively). On the other hand, the speed, bearing and accuracy of each of its measurements correspond to “GPS_sp”, “GPS_bear” and “GPS_acc”, respectively. Note that each GPS measurement is plotted as a single point instead of a continuous signal to highlight its differential behaviour regarding the other sensors. Additionally, each figure contains a series of vertical bars, every 15 s, to delimit the different activities in the same plot. It should be highlighted that these are not continuous signals from one activity to another. Each action is given by different sessions, from the initial moment of data collection until the 15 s have elapsed. As can be seen on the (b) figures, all GPS data is replicated, and the first few seconds of each session are cut out, for the reasons indicated above. Moreover, some clear trends can be observed for each sensor, depending on the activity performed.

Once data were loaded and preprocessed as previously described, the features were extracted. That extraction was based on the application of a sliding window from 20 to 90 s, in increments of 10, with the maximum possible overlap (one second less than each full window size), to have as many samples as possible. Thus, as an example shown in the original work for a window size of 20 s and the dataset containing all the sensors, Table 1 shows the number of available patterns and their distribution by every activity studied. On the other hand, in each of those windows, the features shown in Table 2 were calculated. As can be seen, there is a primary set already proposed in [19] (“Primary set” column) that will be used as a basis for comparison. Similarly, another series of features are also shown (“Proposed additions” column), which were added to the primary set. In this way, it will be possible to examine the differences in performance between the initial set and the one formed by that set plus the proposed collection. Each of the sub-columns that can be observed in this table refers to the sensors to which these features were practised. In the case



(a) (b)

Fig. 2. Examples of data taken by the magnetometer and GPS of a specific individual's smartphone, during the first 15 seconds of each session, for each of the specified activities, being: (a) Raw data. (b) Data after having been preprocessed.

Table 1
Number of patterns available and their distribution by every activity studied, for a window size of 20 seconds and the dataset containing all the sensors.

Activity				
Inactive	**Active**	**Walking**	**Driving**	**Overall**
214,130 (43%)	140,060 (28%)	83,376 (17%)	61,710 (12%)	499,276

Table 2
Feature set used.

Features			
Primary set	Proposed additions		
General	General	Not for GPS	Only for GPS
Mean		Signal magnitude area	
Variance	Energy	Number of zero crosses	
Median absolute deviation	Number of observations	Number of local maxima	Total distance travelled
Maximum	Maximum time gap	Number of local minima	
Minimum	Minimum time gap	Total positive time	
Interquartile range		Total negative time	

of "Not for GPS", they relate to the features that did not make sense to be used for GPS. That is because for the GPS the values are much more separated in time (approximately one value every 10 s) and always remain on the positive side of the signal, as these are absolute increments between observations. Besides, for the same reason, there are not three accurate axes like the X, Y and Z used in the other sensors, so it was also not attainable to use the signal magnitude area feature (SMA). Thus, a specific feature for this sensor was applied, "total distance travelled", which, from the increments of the latitude and longitude values, approximates the distance travelled using a Pythagorean theorem. The resulting value corresponds to the sum of all the hypotenuses, i.e. all the distances calculated in each of the observations.

Regarding the implemented features, it was decided to apply some of those included in the proposed collection, such as "number of zero crosses", "number of local maxima/minima" and "total positive/negative time". This is due to having seen their reliable performance in [37], in which a comparison of different features for HAR was made. The first of those listed refers to the number of times the signal changes from positive to negative or vice versa. All of them were considered attractive given the variability in the activities to be studied. For example, in a case of inactivity, these values should be much lower than those that could be found in a situation where the individual is walking or running. On the other hand, "energy" and "signal magnitude area", are two very common calculations in signals and the HAR field, so it was decided to include them as well. As for "number of observations", "maximum/minimum time gap" and "total distance travelled", these were features that were thought that could be favourable given the peculiarities of this dataset, in order to take advantage of the fact that there are some gaps between the data, mainly in the case of GPS.

3.2. Classification algorithms

Within the scope of HAR, numerous machine learning algorithms can be applied. In this case, it was decided to use the following ones: Support Vector Machine (SVM), Decision Tree (DT), Multilayer Perceptron (MLP), Naïve Bayes (NB), K-Nearest Neighbour (KNN), Random Forest (RF) and Extreme Gradient Boosting (XGB). The selection of these algorithms is due, in most cases, to the fact that they were the most used and with the best results within this field [22,28,29], as seen in the Related Work section. Only the case of XGB would be a novelty, as it has not been seen so much in this area. Anyhow, its addition was considered attractive to the list due to its high popularity in recent years and its outstanding results in many machine learning competitions [38]. Moreover, every algorithm mentioned above was implemented in Python, using the Scikit-learn library [39], as well as the XGBoost one [40] for its own case.

3.2.1. Support Vector Machine

Support Vector Machines (SVM) are machine learning models often used in binary classification problems [41]. This type of models searches for the hyperplane which maximizes the margins between two previously specified and labelled classes. To make this hyperplane non-linear, functions called kernels are used, which are one of the most crucial hyperparameters in SVM. These functions transform non-linear spaces into linear spaces, by changing the dimension in which they are plotted, making

possible the application of this linear approach. Depending on the kernel used (linear, polynomial or radial basis function), the hyperparameters to be applied change. The only fundamental hyperparameter that occurs in any kernel is C, which defines the number of errors that can be accepted by the model, as well as the width of the margins of the resulting hyperplane. In the same way, other fundamental hyperparameters also influence to a great extent the definition of such hyperplane. One of them is gamma (not applicable in linear kernels, among others), which determines the curves that the hyperplane can take, making them more accentuated or softer, depending on the patterns that are introduced into the model. Similarly, for polynomial kernels, the degree of the polynomial broadly affects the curvature that such a hyperplane can take. In fact, for example, if the degree is equal to 1, the result will be equivalent to that of a linear kernel (one straight line).

Usually, SVMs are employed to solve binary classification problems, but they can also be used for multi-class ones. To perform these tasks, it is necessary to choose a *one-vs-one* or *one-vs-all* strategy. In the first case, the classes are modelled in pairs, performing several binary classifications until a final result is obtained. Conversely, in the second case, the models are formed by confronting each class with the rest independently, creating a specific classifier for each situation. In this paper, the one-vs-all approach will be the one implemented, since it is the most used one in the literature [20,42]. In this way, the final result returned will be the average of all the classifiers created in the process.

3.2.2. Decision Tree

Decision Trees (DT) are one of the closest models to human thought, representing knowledge through trees. To do this, they generate a series of rules or questions that they use to predict and classify the data entered. There are numerous tree creation algorithms, including ID3 [43], C4.5 [44] or CART [45]. In this paper, the latter one will be used, as there is a widely accepted version, which is available and on which no modifications have been made for the purpose of comparison. To carry out this creation process, the algorithm follows a series of steps:

1. It starts by looking for the attribute that best defines each of the classes and places it at the top of the tree. This attribute is also known as the root node. To determine the order in which the attributes are evaluated, it uses statistical measures such as information gain. This metric calculates the expected reduction of uncertainty, which is obtained from the division of the dataset into a given attribute.
2. The algorithm then generates a criterion by which it separates the data, depending on the probability distribution of each of the classes in the tree.
3. Finally, it forms branches that split the datasets into subsets known as internal nodes. To evaluate these divisions, the algorithm uses the Gini Index, which provides a score of how good the resulting subsets are. The smaller this value is, the better the division.

Once these steps have been performed, the algorithm repeats the first and second steps until it reaches, in each branch, a leaf node, which is a subset of data that cannot be further divided.

3.2.3. Multilayer Perceptron

The Multilayer Perceptron (MLP) is one of the most widely used neural models nowadays, as well as being one of the first machine learning techniques to appear [46]. Unlike more traditional neuron networks, it can have more than one layer of neurons. For the simplest case, it would consist of three different layers, where the first one would be the input layer, followed by the hidden layer and ending with the output layer. The data are entered by the input layer, taking the predictions in the output layer. The hidden layers can be multiple, depending on how complex the model needs to be for the specified problem. Each layer is represented as follows:

$$y = f(W \times x + b) \tag{1}$$

The letter "f" would be the activation function, which is responsible for describing the input–output relations in a non-linear way. In this way, the model has more power to be more flexible in the description of arbitrary associations. On the other hand, "W" refers to the layer weights, which change as errors are found, by adding the learning rate, which can be constant or dynamic. Similarly, "x" would correspond to the input data vector of the previous network and "b" would be the bias vector, which is an additional set of weights with which to allow the layer to produce a series of output data. In order to carry out the training of the network, it is necessary to define a loss function. This loss will be high if the class predictions do not correspond to the ground truth, and will be low if they do. In this way, the layer weight values (W) would be added to this loss. The idea is that during the training of the model this loss value will be low. For this purpose, functions called optimizers are used, which look for the appropriate values of the weights with which to lower this value. In this paper, the Adaptive Moment Estimation (Adam) function will be used [47], as it is the more recommended one for large datasets. Moreover, to avoid overfitting, these algorithms use an alpha parameter that penalizes weights with large magnitudes.

3.2.4. Naïve Bayes

The Naïve Bayes (NB) classifiers are a collection of classification algorithms based on Bayes' Theorem [48]. This theorem expresses the conditional probability of an event A given B, from the conditional probability of B given A and the marginal probability of A. This definition is represented in the Bayes' Rule:

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B|A)\Pr(A) + \Pr(B|\neg A)\Pr(\neg A)} \tag{2}$$

Thus, it is not a single algorithm, but a family of algorithms that share a common principle, which is that in each pair of classified features, each one is independent of the other. The main differences between each of the algorithms of this family are based on the assumption they make regarding the distribution of $\Pr(B|A)$. The continuous values associated with each feature are assumed to follow a specific distribution, such as the Gaussian one, a given multinomial distribution or Bernoulli's multivariate event model, where the features introduced are independent booleans (binary variables) [49]. In this paper, this last assumption will be used, since the rest do not apply to our problem, or offered preliminary results far below what it is considered randomness (50% success rate). On the other hand, although the assumptions made by this kind of methods may seem very simple, the truth is that this kind of algorithms have worked well in many tasks. Moreover, it is an extremely fast classifier compared to other types of more sophisticated machine learning algorithms, so it is considered that it is worth trying.

3.2.5. K-Nearest Neighbour

The K-Nearest Neighbour (KNN) algorithm is supervised and instance-based, so it needs the data entered to be pre-labelled, as well as not being able to create a model explicitly [50,51]. Instead, it memorizes the training instances that are used as a basis for the prediction phase. The most crucial point in this algorithm is the selection of the "K" number, which represents the number of neighbours that are taken into account in the neighbourhood to

classify the previously specified groups. In this way, the algorithm follows a series of steps defined for each of the observations in the data:

1. The distances between the selected observation and all other observations in the dataset are calculated. This distance can be understood as a similarity measure between these elements. It is calculated using a predefined function, such as the Euclidean or the Manhattan distance.
2. Then, the closest K-elements are selected and a majority vote is taken among them. The dominant class will be the one deciding the final classification, depending also on the weights given to each of these classes.

One of the most prominent problems in KNN is the immense amount of memory and time required as the selected dataset grows. That is because they need to evaluate every observation in the data, so if the number of features and data is very high, the computational resources required for their training can be quite significant. Nevertheless, it is considered as an algorithm that can produce great results, being also easy to understand and to implement.

3.2.6. Random Forest

Models based on Random Forest's (RF) algorithm are among the most popular nowadays [52,53]. Through the creation of multiple decision trees from previously tagged data, they can produce very robust models. That is because, by having to create different trees, they can select the best possible solution in a much more general and flexible way, as well as also reducing overfitting by not having a single decision tree. Thus, the main work of the algorithm is divided into the following steps:

1. First, you start by selecting various subsets randomly over the given dataset.
2. Then, the algorithm will build decision trees for each of these examples, following the steps described in 3.2.2. The number of decision trees constructed will be given from the number of estimators hyperparameter, which is specified previously.
3. Once the trees have been created, the resulting prediction is obtained from each of them. At this point, a vote is also taken on each of these resulting values, where the dominant class will decide the final result.
4. Finally, the most voted class is selected as the final result of the prediction.

When making predictions with the model already created, this algorithm is usually much slower than the rest. That is because of having to average the outcomes of each of the trees that make up the final model. Even so, it is widely used today because it is capable of creating very robust models with a high-grade performance, being also faster to train than many of the other artificial intelligence algorithms used nowadays.

3.2.7. Extreme Gradient Boosting

Although Extreme Gradient Boosting (XGB) is not an algorithm by itself but a refined implementation of the Gradient Boosting algorithm [40], it is worth to be considered. The main reason is that this approach has won several competitions and has recurrently offered very competitive results in the related literature [38]. The implementation provides a more efficient and flexible method by parallelizing the tree boosting process. Concerning the Gradient Boosting Machine (GBM), it is an algorithm that seeks the production of a model through the formation of numerous "weak" prediction models, usually decision trees. For this purpose, decision trees are created in a stage-wise fashion, sequentially, following the same lines listed in 3.2.2. As in Random Forest, a number of estimators hyperparameter is used to determine the number of trees to be created. The idea is to seek the progressive improvement of the final model. To do this, a loss function is defined that evaluates the performance of the last tree created, and which, presumably, will progressively decrease as all the observations in the trees built are better classified. That results in a final model that is much more robust and easy to tune, as well as offering excellent results. However, it can be quite sensitive to overfitting and noise, so it is important to be careful when training it.

Table 3
Binary confusion matrix example.

		Model output	
		False	**True**
Ground truth	**False**	TN	FP
	True	FP	TP

3.3. Evaluation metrics

One of the most elementary and easily interpreted metrics is the confusion matrix. A confusion matrix is a table that facilitates the visualization of the performance of a classification model from a set of test data. A simple example would be the one showed in Table 3. From this, we can draw many widely used terms to evaluate these confusion matrices, such as: precision, recall, accuracy and F_1-score [54].

Precision and recall are the metrics used to measure the quality and quantity of the classifications made, respectively. Precision measures the number of true positives, divided by the total number of positive results returned. Concerning recall, it measures the number of true positives, divided by the number of correct results that should have been returned. Their formulas would be as follows:

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

Any other way, accuracy and F_1-score metrics are used to know the performance of a model in test. The first consists of the measurement of all correctly identified cases, while the second is based on the harmonic mean of the recall and precision metrics. Their formulas would be as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

However, in a multi-class problem such as the one in this paper, the way these metrics are computed changes. As a model example, it can be seen how these values would be calculated in Table 4, compared to the binary case of the previous example. Thus, to calculate the overall precision and recall of the whole model, the final value is obtained from various types of averaging, among which the following stand out: *micro* and *macro* [55]. The first one considers the total of TP, FN and FP to calculate the metric, so it is suitable for problems with mutually exclusive classes. As for macro, it returns the average of calculating the metric for each label, regardless of the proportion of each of them in the dataset. Concerning accuracy, it is usually calculated in the same way as in the latter case.

On the other hand, F_1-score has more ways of weighting its result to evaluate multi-class classification problems. In addition

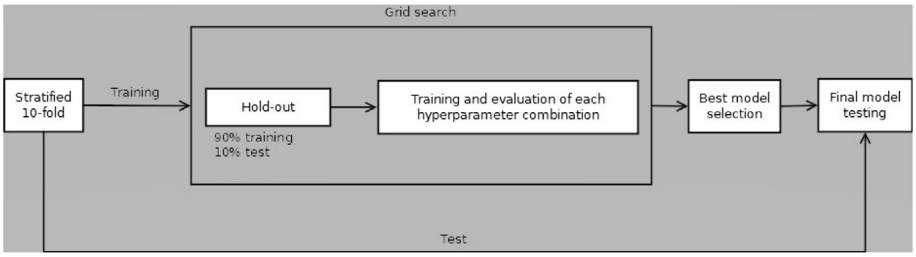


Fig. 3. Methodology followed for each algorithm and window size specified.

Table 4
TP, TN, FP and FN calculations for the "Class 1" class of a multi-class confusion matrix example.

		Model output		
		Class 1	Class 2	Class 3
Ground truth	**Class 1**	TP		FP
	Class 2 Class 3	FN		TN

to the two most commonly used ones discussed above, there is a variant of macro, *macro-weighted* (provided by the Scikit-learn library), which does take into account the data proportions by averaging the precisions and recalls of each of the classes involved.

Although accuracy is the most widely used measure globally, F_1-score is also closely linked to the correct classification of groups, but it is not as influenced by possible imbalances between classes in the datasets [56]. In fact, when this occurs, the accuracy could give an incorrect impression of the final results. In this paper, the F_1-score will be used as the initial assessment metric, specifically with the macro-weighted case of averaging, as the proportions of the data are quite inclined towards one of the classes involved. Anyhow, the final results will be shown mainly with the accuracy metric, since it is the most common one for comparison with the rest of the works.

3.4. Validation and optimization techniques

One of the most widely used model validation techniques in the field of machine learning is *cross-validation* [57]. Before entering the data in the model to be trained, the data is divided into training and test. This elementary division of data is also called *hold-out*. In this way, there is a set of data destined to train the model and another subset that serves to test its performance, with data a priori unknown to it. One of the most common ways of making this division is employing the *k-fold cross-validation* technique. Here, the aim is to partition the original data set into "k" equal sized subsamples. Then, one of these subsamples is selected as the validation set to test the model, being the rest of subsamples used to train it. This process is repeated "k" times until all subsets are selected as a test once. Finally, the results obtained by the model are averaged and the relevant performance evaluation metrics are calculated. In this paper, a variant of this technique will be used, called *stratified k-folding*. This alternative seeks that the proportion of each class in each of the subsets created is practically the same. In this way, the existing imbalance on the dataset and its possible influences on the model performance are avoided.

As for the optimization of the models to be used, there is a technique widely employed in the field called *grid search* [58]. A grid search is a process that consists of an exhaustive search for the best combination of hyperparameters introduced to a model using a particular algorithm. Here, each of the possibilities of the set of hyperparameters previously indicated is tested. It is a long and expensive process, but with which it is possible to know the best possible performance of the model to be used. In this paper, the F_1-score commented in 3.3 will be used as the evaluation metric for these combinations. Also, some of these combinations will be trained not once, but 50 times. This is because non-deterministic algorithms such as Multilayer Perceptron (MLP), Random Forest (RF) and Extreme Gradient Boosting (XGB) will be used. The nature of these types of algorithms means that there is always a small random component that affects their final result. Thus, by averaging the values obtained in each of these repetitions, it is possible to get a reliable outcome.

In the present paper, these techniques will be used together to find the most robust and optimal model for these problems, as shown in Fig. 3. For this purpose, an initial 10-fold will be applied to the dataset. Each of the training parts of each fold will be introduced later in the grid search. To validate the performance of each of these models, the data of the corresponding fold will be divided applying a hold-out. Hence, 90% of the data will be for training and the resulting 10% for testing. It was decided to apply this technique instead of the traditional k-fold cross-validation due to the high number of experiments to be performed in the grid search. In this way, we obtain the final results in a reasonable time, as we do not have to evaluate each of the folds. Besides, the high number of patterns available (around 450,000 for each fold) could lead to greater redundancy in the data if we applied another 10-fold. Finally, the best model selected by the grid search will be tested again, this time with the corresponding part for testing of the initial 10-fold. Thus, the models are tested with truly unseen data during their training, making the final result more realistic concerning their data generalization capabilities.

4. Results and discussion

Within this section, all the results of the experiments carried out will be displayed. First, in Section 4.1, all the data obtained will be represented, with their corresponding graphs and evaluation metrics. Then, in Section 4.2, the main observations and considerations of the results obtained will be discussed.

4.1. Results

After having prepared the data, applying the selected window sizes with the previously discussed features (Table 2), a series of experiments were conducted on them. As described before, the sliding window sizes ranged from 20 to 90 s, in increments of 10, with the maximum possible overlap (one second less than each full window size). To do this, the most applied machine learning algorithms in HAR were used, which are: Support Vector Machine (SVM), Decision Tree (DT), Multilayer Perceptron (MLP), Naïve Bayes (NB), K-Nearest Neighbour (KNN) and Random Forest (RF), with the addition of Extreme Gradient Boosting (XGB). In order to get the best possible results for each of them, it was decided to explore the best architecture for each of them beforehand. For

Table 5
Chosen hyperparameters to perform further grid search for each machine learning algorithm.

Algorithms	Hyperparameters
SVM	Kernel = {linear, RBF (Radial Basis Function), polynomial} C = {1, 10, 100, 1000, 10000} Gamma (not applicable for linear kernels) = {0.0001, 0.001, 0.01, 0.1, 1} Degree (only applicable for polynomial kernels) = {1, 2, 3, 4} Maximum iterations = 1000
DT	Maximum depth = {5, 8, 15, 30, None} Leaf minimum size = {1, 2, 4, 8, 16, 32, 64} Minimum size for node division = {2, 5, 10}
MLP	Hidden layers and units size = {(5,), (10,), (20,), (30,), (50,), (70,), (100,) (5, 5), (10, 10), (20, 20), (30, 30), (50, 50), (70, 70), (100, 100)} Activation function = {tanh (Hyperbolic Tangent), ReLU (Rectified Linear Unit)} Alpha = {0.0001, 0.05, 0.1}
NB	It has no specific hyperparameters. In our case, we used the model based on Bernoulli, since the rest, after some preliminary tests, were not applicable to this problem.
KNN	Number of neighbours = {3, 5, 7, 11, 15, 19, 24, 29, 34} Weights = {uniform, distance} Metrics = {Euclidean, Manhattan} Leafs size = {30, 50, 100}
RF	Number of estimators = {100, 250, 500, 1000} Maximum depth = {5, 12, 25, 50} Leaf minimum size = {1, 2, 4} Minimum size for node division = {2, 5, 10}
XGB	Number of estimators = {100, 300, 500, 800, 1200} Maximum depth = {5, 8, 15, 30, None} Minimum size for node division = {1, 3, 5}

Table 6
Accuracy results comparison between algorithms and sliding window sizes for the initial feature set and the complete dataset.

	Window size							
	20	30	40	50	60	70	80	90
SVM	69.28% ±15.10%	78.07% ±10.37%	81.30% ±8.79%	79.52% ±10.57%	78.84% ±8.54%	79.45% ±9.77%	81.23% ±8.53%	80.90% ±9.02%
DT	88.17% ±12.47%	85.79% ±16.40%	88.12% ±8.83%	86.71% ±13.47%	87.82% ±13.00%	86.17% ±14.37%	87.57% ±12.81%	89.91% ±7.15%
MLP	86.46% ±6.30%	86.80% ±6.02%	86.85% ±6.12%	86.59% ±6.61%	86.65% ±7.11%	86.39% ±7.69%	86.57% ±7.90%	85.47% ±8.65%
NB	78.11% ±6.93%	78.68% ±6.61%	79.09% ±6.73%	79.48% ±6.92%	79.66% ±6.95%	80.01% ±7.08%	80.09% ±7.08%	79.62% ±6.81%
KNN	85.68% ±7.20%	86.30% ±6.20%	86.83% ±6.34%	86.32% ±6.48%	86.56% ±6.50%	86.84% ±6.77%	86.99% ±6.57%	87.09% ±6.76%
RF	91.78% ±5.20%	92.27% ±5.58%	92.36% ±5.74%	**92.56%** ±**5.92%**	92.55% ±5.99%	92.29% ±5.86%	92.37% ±5.80%	92.28% ±6.50%
XGB	90.58% ±7.57%	90.47% ±9.09%	91.42% ±7.62%	91.36% ±7.76%	91.80% ±8.06%	92.23% ±7.30%	91.21% ±6.98%	91.30% ±7.09%

this reason, different values for the most crucial hyperparameters of each of them were selected, in search of the best possible combination between them. To choose the best final result, the next steps were followed, for each of the previously specified datasets:

1. The first step was to carry out a stratified 10-fold to have ten different datasets, with approximately the same pattern distribution for each class.
2. Then, with each of the previous folds, a grid search was carried out to find the best combination of hyperparameters of each algorithm. Because the dataset used has some unbalance towards the class of "inactive", the resulting predictions were evaluated using the F_1-score metric, offering a value that should better represent the performance of the model. On the other hand, concerning the hyperparameters used for each of the selected machine learning algorithms, they were all arranged in Table 5. The intention was to have a wide range of hyperparameters, to know the best possible option. Therefore, attempts were made to increase the number of possibilities in those that were more influential in the final result. However, the overall complexity of each training increases with higher amounts of hyperparameters, so it was necessary to be conservative in general. Any other hyperparameters not listed there were chosen and set to the default value, after some preliminary experiments and confirmation of good performance. In this way, the number of experiments is adequate to find the optimal model in a reasonable time.
3. For the MLP and RF algorithms, given their non-deterministic nature, the previous step was repeated 50 times, for each of the ten initial folds. In this way, we avoid that their random behaviour affects the final result, averaging all the obtained ones. On the other hand, although the XGB algorithm also should have this non-deterministic nature, for the Python package used in this work the results are always the same, so it was not necessary to perform these repetitions.
4. Once the grid search was completed, the results were evaluated and the best combination of hyperparameters for each of the folds was selected. Then, each of the best models selected was tested with truly unseen data from the initial 10-fold.

Table 7
Accuracy results comparison between algorithms and sliding window sizes for the proposed feature set and the complete dataset.

	Window size							
	20	30	40	50	60	70	80	90
SVM	80.77% ±12.64%	82.00% ±13.45%	82.15% ±14.12%	83.38% ±10.85%	83.59% ±11.84%	85.12% ±11.55%	86.56% ±11.30%	85.98% ±11.18%
DT	89.99% ±6.13%	89.92% ±6.62%	87.95% ±10.18%	88.27% ±11.01%	87.68% ±12.17%	86.94% ±14.31%	89.63% ±8.38%	88.26% ±10.26%
MLP	83.76% ±10.37%	84.00% ±10.49%	84.24% ±10.38%	84.60% ±10.22%	84.73% ±10.32%	84.32% ±10.79%	84.96% ±10.37%	84.43% ±10.09%
NB	81.69% ±7.20%	82.21% ±7.16%	82.49% ±7.16%	82.77% ±7.32%	82.86% ±7.60%	83.06% ±7.67%	83.27% ±7.78%	82.72% ±7.68%
KNN	87.62% ±7.37%	88.27% ±7.02%	88.72% ±6.99%	88.99% ±6.96%	88.76% ±7.94%	88.80% ±8.01%	89.02% ±8.00%	88.03% ±7.94%
RF	91.71% ±5.47%	92.08% ±5.49%	92.26% ±5.68%	92.51% ±5.86%	92.73% ±5.98%	92.77% ±6.16%	**92.97% ±6.23%**	92.61% ±6.60%
XGB	88.32% ±12.29%	88.99% ±11.66%	88.87% ±12.61%	88.78% ±14.26%	89.72% ±11.54%	89.55% ±12.54%	90.38% ±9.64%	91.15% ±7.01%

Once the steps indicated in the previous paragraph have been carried out, for each of the algorithms and sliding window sizes used, the results shown in the Tables 6 and 7 were obtained. The value below each accuracy result refers to its standard deviation. These values correspond to the initial and proposed sets of features, respectively. As can be seen, the tree-based models, DT, RF and XGB, work considerably better than the rest, since they are the only ones capable of approaching and even surpassing the 90% of success. The cases of RF and XGB stand out even more because they are less variable than DT. However, as it is logical, the computational complexity of these algorithms is much higher than in the case of DT. On the other hand, the size of the windows is not as influential as it was thought at first, except for SVM, where the fluctuations are quite broad, especially in the step of 20 to 30 s. For that reason, although the best value is obtained with a 80-second window, the model might perform similarly with a window such as the 20-second one, since the difference in accuracy is pretty low in most cases. In this way, they would be easier to apply to a real-life environment, as they can be used more finely with the activities that are being carried out at each moment. As for the differences between each of the sets of features, the truth is that these were much smaller than expected. In fact, after performing a T-test between each pair of values of both groups, significant differences were only found in one case. This only case corresponds to the SVM algorithm, specifically for the 30-second window size. For the rest, the p-values were all above 0.1, which means that the results are statistically similar. However, some clear improvements are visible for the proposed set, as in the cases of SVM, NB and KNN. On the contrary, for MLP and XGB, it seems that the results are slightly worse. The rest of the algorithms remain with very similar values to each other, especially in the case of RF. Additionally, the test results between each pair of results in Tables 6 and 7 seem to indicate that the new features do not add information that would significantly increase accuracy, not making it possible to reach a reliable conclusion. Moreover, in the same way, in Fig. 4 the F_1-scores resulting from each of the algorithms and window sizes used are shown, for each of the sets of features applied. Both metrics are displayed to, on the one hand, show the numerical accuracy results in tables as a comparison with other works. On the other, it is also possible to show the differences between each case in a much more visual way through the F_1-score graphs.

On the other hand, the results obtained for the dataset that did not include the gyro, both with the old and the new features, are also shown in Tables 8 and 9, respectively. As in the previous case, a T-test was also carried out to check the differences between the sets of features introduced in the models. Once again, the only case where significant differences can be seen is with SVM and a window size of 30 s. Therefore, the results are, in general, statistically similar, so we cannot claim that they are different. In any case, the observations from before are repeated, with improvements in SVM, NB and KNN, as well as a worsening in MLP and XGB. However, in this case, DT worsens slightly. Anyhow, as can be seen, the results are, in general, somewhat worse than those obtained with all the sensors. Although there is indeed some case with some improvement, like the one found in [19], with SVM and 20 s of a window, it is considered that the models work better with the complete case. In this way, the gyroscope does provide a slight improvement in the models, as shown in previous works that included it in their tests. Similarly, the F_1-scores for this case are also shown in Fig. 5.

Regarding the winning hyperparameter combinations of each algorithm, for each window size and dataset, the best values of F_1-score obtained for each of those combinations are shown in Fig. 6. Here, the "Feature dataset" field refers to the combination of the set of features and the dataset used in each case, as shown in Table 10. Thus, the numbers 1 and 2 would correspond to the dataset containing all sensors, while 3 and 4 to the dataset not including gyroscope data. At the same time, the numbers 1 and 3 would also correspond to the initial set of features, while 2 and 4 to that proposed in this paper. As can be seen, no single winning hyperparameter combination is obtained for each algorithm, but different sets depending on the case study. Therefore, the data displayed there for each of the applied algorithms are detailed below:

- SVM. The RBF kernel was by far the most selected, with only a few cases of grade 3 polynomial kernels in the larger window sizes when using the proposed feature set. The linear kernel was always lower in overall performance. As for C, the values fluctuated a lot, but seemed to settle more for the intermediate values of 10, 100 and 1000, leaving quite apart from the value 1. Finally, with gamma something similar happened, although the extreme values of 1 and 0.0001 were practically never selected, being clearer the dominance of the rest of the values, more or less in equal parts.
- DT. The maximum depths in DT always remained at the low values of 5 and 8, except for some isolated cases for the complete dataset with the primary feature set (case 1). In the latter case, the chosen value was 15. As for the minimums per leaf and to divide the node, their values were very arbitrary and all were selected more or less equally, so it is not possible to draw a reliable conclusion.
- MLP. The case of (100,) was by far the most selected for the size of the hidden layers. However, there were also

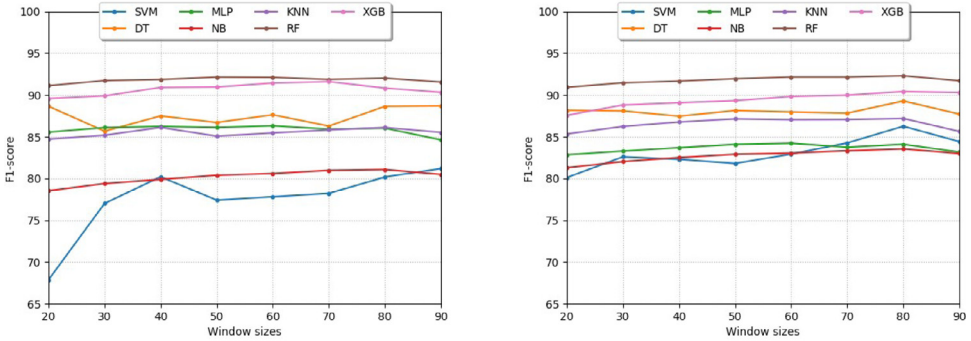


Fig. 4. F_1-scores for: (a) The initial feature set and the complete dataset. (b) The proposed feature set and the complete dataset.

Table 8
Accuracy results comparison between algorithms and sliding window sizes for the initial feature set and the dataset missing the gyroscope's measurements.

	Window size							
	20	30	40	50	60	70	80	90
SVM	74.39% ±10.75%	73.47% ±9.66%	76.77% ±10.98%	81.00% ±9.41%	80.66% ±12.96%	82.56% ±7.68%	80.70% ±9.22%	81.77% ±9.28%
DT	82.74% ±13.36%	83.67% ±13.72%	87.07% ±8.23%	87.06% ±9.19%	87.88% ±8.75%	88.82% ±6.29%	88.60% ±7.44%	87.28% ±8.93%
MLP	86.35% ±4.95%	86.70% ±4.97%	86.46% ±5.22%	86.63% ±6.06%	86.88% ±6.56%	87.16% ±6.85%	87.00% ±7.10%	87.01% ±7.42%
NB	80.23% ±7.30%	80.39% ±7.27%	80.62% ±7.50%	81.24% ±7.32%	80.94% ±7.35%	80.97% ±7.63%	81.32% ±7.63%	81.42% ±7.13%
KNN	84.61% ±6.15%	86.10% ±5.13%	86.28% ±5.57%	86.81% ±5.38%	86.84% ±5.52%	86.68% ±5.85%	86.92% ±5.88%	86.18% ±9.49%
RF	89.34% ±6.67%	89.75% ±7.27%	90.03% ±7.64%	90.45% ±7.44%	90.62% ±7.52%	90.63% ±7.92%	**90.76% ±8.01%**	90.36% ±8.06%
XGB	87.40% ±10.57%	89.21% ±7.34%	89.75% ±7.22%	90.35% ±6.80%	90.27% ±7.41%	89.53% ±8.61%	89.75% ±8.06%	90.50% ±6.62%

Table 9
Accuracy results comparison between algorithms and sliding window sizes for the proposed feature set and the dataset missing the gyroscope's measurements.

	Window size							
	20	30	40	50	60	70	80	90
SVM	80.65% ±11.65%	81.34% ±9.45%	80.66% ±11.48%	82.57% ±9.56%	82.77% ±8.71%	83.71% ±7.96%	84.64% ±7.86%	84.88% ±7.75%
DT	82.15% ±12.88%	81.97% ±12.50%	84.77% ±10.49%	84.71% ±12.07%	84.04% ±12.62%	84.64% ±12.34%	85.55% ±11.90%	86.66% ±9.72%
MLP	84.07% ±7.82%	84.57% ±8.02%	85.03% ±7.98%	85.45% ±7.68%	85.69% ±7.54%	85.51% ±7.82%	85.94% ±7.55%	86.17% ±7.42%
NB	81.49% ±6.44%	82.18% ±6.70%	82.53% ±6.93%	82.60% ±7.23%	82.72% ±7.34%	82.92% ±7.66%	83.04% ±8.04%	82.75% ±7.72%
KNN	85.43% ±6.49%	86.25% ±6.50%	86.96% ±6.34%	87.16% ±6.17%	87.47% ±6.05%	87.45% ±6.04%	87.60% ±5.64%	86.73% ±6.05%
RF	88.94% ±6.22%	89.55% ±6.24%	90.17% ±6.29%	90.19% ±7.15%	90.41% ±7.51%	90.50% ±7.85%	**90.62% ±7.97%**	90.22% ±7.90%
XGB	85.57% ±11.17%	84.96% ±13.02%	87.33% ±11.12%	86.15% ±12.98%	86.57% ±12.75%	87.44% ±11.37%	88.33% ±10.61%	87.32% ±10.79%

some cases of (70,), for the 90-second windows, and (5,) for the 20-second ones (for the primary feature set). As for the activation functions, the tanh (Hyperbolic Tangent) case dominated for the primary feature set (cases 1 and 3), while ReLU (Rectified Linear Unit) was always the chosen function in the proposed one (cases 2 and 4). Besides, the ReLU was also always chosen for the 20 and 30-second windows of the cases 1 and 3. Regarding the alpha values, generally, those of 0.1 were chosen much more, with some appearances of 0.05 and only one choice of 0.0001.

- NB. No hyperparameters to choose.
- KNN. Here, the number of neighbours was always 34 for the smaller window sizes (until 40 or 50 s). For the following ones, the number was regularly diluted to the lowest values in the list with the size of 90 s, except for the complete dataset with the primary feature set (case 1), where these high values were relatively constant. As for the weight, it was pretty arbitrary in all cases, so it is not possible to draw a reliable conclusion. Concerning the chosen metric, it was always Manhattan, in absolutely all measurements. Finally,

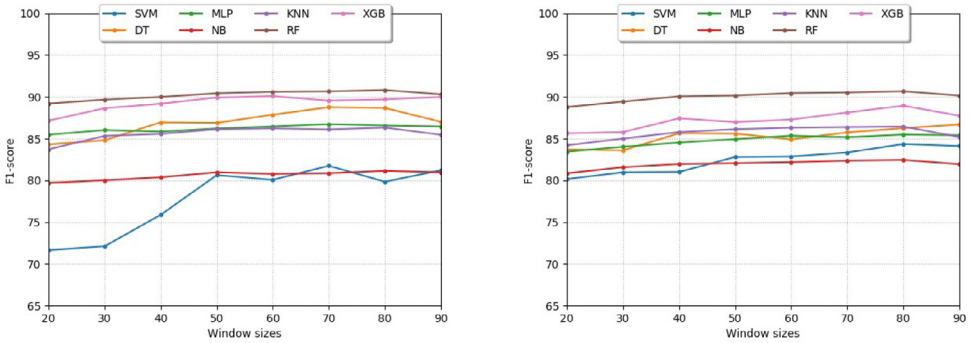


Fig. 5. F_1-scores for: (a) The initial feature set and the dataset missing the gyroscope's measurements. (b) The proposed feature set and the dataset missing the gyroscope's measurements.

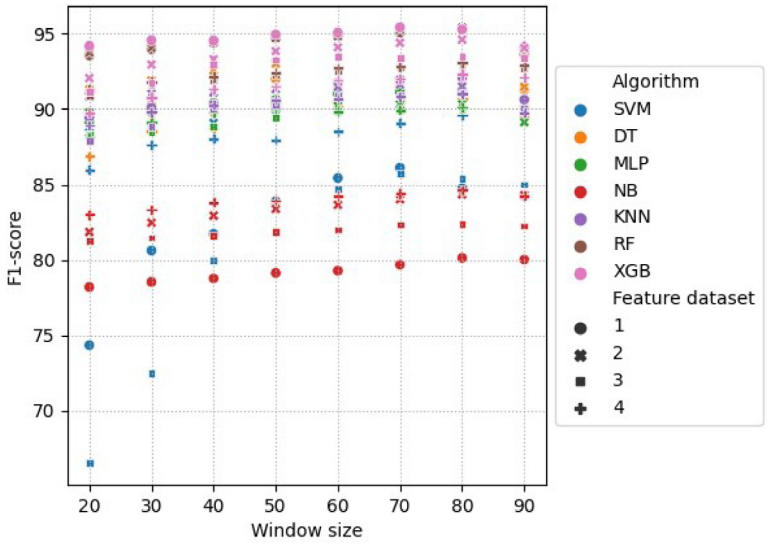


Fig. 6. F_1-scores for each winning hyperparameter combination of each algorithm used.

the size of the leaves did not seem to have any influence whatsoever on the results, as the accuracy was always the same for any of the three values studied.

- RF. In the vast majority of cases, the number of estimators remained at the value of 1000 and, to a lesser extent, 500. The value of 1000 dominated mostly in cases where the proposed feature set was applied (cases 2 and 4). On the other hand, the maximum depth was more inclined towards the mean values of 12 and 25, with some cases of 50. The dominance of these mean values was clearest in cases 2 and 4, as with the previous parameter. As for the minimum per leaf, the most selected value was 1, although in case 1, 4 was the most dominant by far. Finally, the minimum to split the node does not seem to be entirely conclusive, as all cases were selected more or less equally, with some tendency towards the lower values of 2 and 5. The value of 10 was only ever selected for cases 3 and 4 (non-gyroscope ones).
- XGB. In this case, the values are pretty arbitrary. However, there does seem to be a trend towards the number of 1200 estimators compared to the rest, especially with the non-gyroscope dataset. As far as the maximum depth is concerned, 5 and 8 were the most widely used values. Finally, the minimum to divide the node was centred much more on the lower numbers of 1 and 3.

To select the best-resulting model, a Critical Difference diagram was carried out, as shown in Fig. 7. This diagram was constructed from all the datasets, features and window sizes used, with the best values obtained for each case (the ones showed in Tables 6, Table 7, Tables 8 and 9). As it is shown in the aforementioned figure, RF, XGB, DT and MLP models appear to be statistically equivalent. From these four, given the results, it was decided to select RF, because it has the highest accuracy peaks and is less computationally complex than XGB. Additionally, although it requires more time and computational resources than MLP and DT, its performance is considerably better, as well as being a more advanced version of the latter one.

As can be seen in all the tables and figures shown, the best model obtained is the one thrown by the Random Forest algorithm, for 80-second time windows and for the proposed set of features. This case yields the average confusion matrix shown in Table 11, along with its particular metrics (recall, precision and accuracy). The model manages to correctly classify all activities, although some problems with the "active" action are visible. That is because this activity is very diffuse, and can include both moments of activity and inactivity while the individual remains "active". Thus, some confusion can be expected from the classifier with the rest of the activities concerning this one. Although there is still room for improvement, it is considered that the classifier

Table 10
Combinations of dataset and feature set used.

	Dataset		Features	
	Complete	Missing gyroscope	Primary set	Proposed additions
Case 1	X		X	
Case 2	X			X
Case 3		X	X	
Case 4		X		X

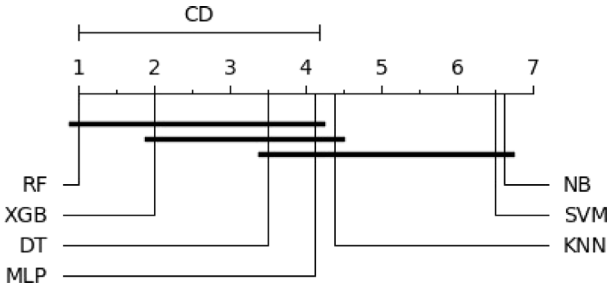


Fig. 7. Critical Difference diagram made from all the results obtained with all the algorithms used.

Table 11
Average confusion matrix for the best combination found.

	Ground truth				Precision
	Inactive	Active	Walking	Driving	
Inactive	19,965	230	261	13	97.54%
Active	888	12,980	1,005	373	85.14%
Walking	24	325	6,043	94	93.17%
Driving	50	44	29	5,157	97.67%
Recall	95.40%	95.59%	82.35%	91.49%	92.97%

Table 12
Average confusion matrix for the 20-second window size option of the best case found.

	Ground truth				Precision
	Inactive	Active	Walking	Driving	
Inactive	20,451	328	190	22	97.43%
Active	852	13,089	1,359	493	82.88%
Walking	51	474	6,700	106	91.39%
Driving	58	115	90	5,550	95.48%
Recall	95.51%	93.45%	80.35%	89.94%	91.71%

achieved far exceeds what was expected, with a resulting accuracy of 92.97%, a much higher value than that of 74.39% achieved in other works [19].

That was the highest result among all the combinations of feature set, window size and dataset. However, after performing a Tukey test, it was clear that there were no significant differences among the different window sizes, as it is evident if we take a look at Fig. 8. Thus, any window size would be feasible to be selected as the best, depending on the problem in which it would be used. In this case, it is considered that 20-second windows would be more than enough to obtain good results, since it would allow a more suitable classification of the activities to be studied by being able to separate them into 20-second intervals. The average confusion matrix for this case would be the one shown in Table 12. As can be seen, the most fundamental differences lie in the "active" activity, as noted above. Even so, the results are statistically similar to those of the best case found with window sizes of 80 s, so it is considered feasible to select this one option preferably.

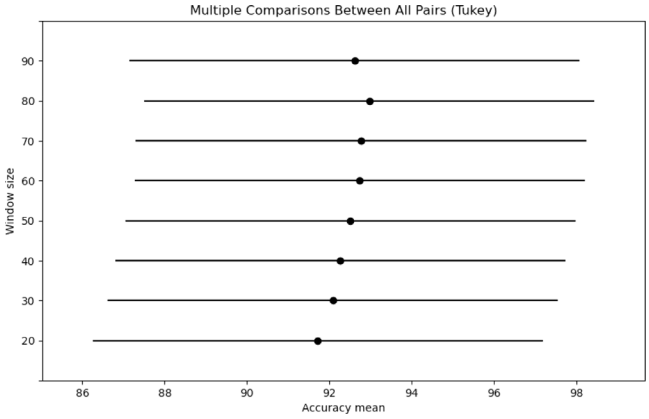


Fig. 8. Results of the Tukey test performed for all window sizes used with Random Forest, for the complete dataset and proposed feature set (case 2).

4.2. Discussion

The results obtained in this paper manage to advance to a great extent towards that real-life environment that is so much sought after. The resulting accuracy of the best model found is exceedingly superior to the best obtained so far, from 74.39% to 92.97%. Besides, several findings have been made about the given dataset, as it appears that the size of the sliding window is not as crucial as first thought. Consequently, the results are quite similar between all the window sizes used in the vast majority of cases.

On the other hand, it has also been possible to reinforce the demonstration of the advantages of using the gyroscope as opposed to not using it in HAR, resulting in a better performance than when it is not used. However, in an effort to further improve the final results, it has not been possible to enhance the primary set of features used in [19]. The additions proposed in this paper have not been entirely conclusive, since the comparison between using each set has quite similar performances to each other, with improvements and worsenings depending on the algorithm and window size used. While there are clear cases of an upgrade such as SVM and NB, the same cannot be said of MLP, XGB and even DT. Perhaps the most robust examples in this sense were KNN and RF, with pretty slight variations in both cases. Probably the proposed features are adding noise to some algorithms and hence are not entirely favourable.

In any case, given the results, it can be concluded that the best algorithms to use in this matter are the tree-based ones (DT, RF and XGB), since they have been the ones that have given the best outcomes with a considerable difference from the rest. Even so, MLP and KNN deserve special mention, since its results have been kept only a little below the latter.

Also, on the other hand, it should be noted that there are still problems in optimally discerning the activity of "active", unlike the rest, where the percentage of success is very high. Although this activity is very diffuse and can be easily confused with cases of "inactive" or "walking" (especially with this one),

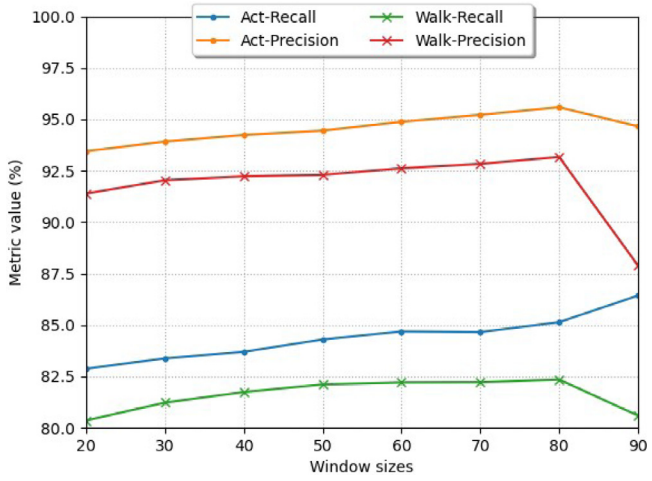


Fig. 9. Recall and precision values for the "active" and "walking" activities and every sliding window size used.

it is possible that with a more in-depth study for this case the final solution will be found. In fact, looking at the confusion matrices for each of the window sizes of the case indicated in Table 11, a relevant trend is indeed observed. With smaller window sizes, this confusion is more pronounced, with more samples misclassified among these activities. Similarly, with larger sizes, this confusion is milder. This is shown in Fig. 9. However, as previously discussed, window size did not end up being a crucial parameter for the overall performance of the models studied, although it seems to be a trend to have the highest accuracy peaks on the 80-second windows. Additionally, as can be seen in that figure, there is a clear downgrade when the window size reaches the value of 90. Perhaps using other more influential features for this activity could make it possible to obtain the optimal model for this case. Also, the application of other types of algorithms, such as LSTM and CNN, characteristic of the deep learning aspect, could improve the performance, as they are giving outstanding results in HAR in recent years.

5. Conclusions and future work

In this work, a series of experiments were carried out in search of the improvement of the last model developed in HAR, for a dataset oriented to the introduction of this problem in a real-life environment. By using various machine learning algorithms, different features and much larger sliding windows, the results were severely improved. In addition, it has been observed that the preliminary results on the dataset in which gyroscope's measurements were missing were inconclusive. This sensor finally proved to improve the final performance in a general way in all the algorithms used, as in the rest of the works that included it in their experiments.

Unfortunately, the proposed set of features has not obtained such good results. The performance of the algorithms with this set and the primary one is quite similar in most cases, with ups and downs depending on the algorithm to be used, so its impact is not entirely conclusive. Perhaps it is necessary to think of other types of features that could improve the classification of the "active" activity, which, although the current models differentiate it relatively well, is quite improvable. Also, it could be interesting to make some kind of evaluation and selection of features, applying, for example, a Principal Component Analysis (PCA), to see which ones are the most suitable.

Another point worth to be mentioned is the fact that the size of the windows does not have as much influence on the final results as was first thought. For that reason, a window of 20 s could be perfectly chosen (at least in the best case found, for the Random Forest algorithm), as it would be easier to apply in a real-life environment. In this way, it could be possible to detect and classify each activity more finely, as well as making it possible to get these results in a shorter time since the start of the action.

For the aforementioned reasons, it is considered that the results can still be upgraded. Perhaps also with the application of algorithms purely focused on deep learning, such as CNN or LSTM, which are the two most widely used algorithms in recent years in HAR and which, apparently, are offering the best results nowadays. For this purpose, the exploration will continue in order to improve these results in the future, probably with the algorithms mentioned above, in search of the optimal model.

CRediT authorship contribution statement

Daniel Garcia-Gonzalez: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Daniel Rivero:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – review & editing. **Enrique Fernandez-Blanco:** Conceptualization, Formal analysis, Methodology, Supervision, Writing – review & editing. **Miguel R. Luaces:** Funding acquisition, Project administration, Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We would like to thank the support given by CESGA and CITIC to execute the code related to this paper.

Funding

This research was partially funded by MCIN/AEI/10.13039/501100011033, NextGenerationEU/PRTR, FLATCITY-POC, Spain [grant number PDC2021-121239-C31]; MCIN/AEI/10.13039/501100011033 MAGIST, Spain [grant number PID2019-105221RB-C41]; Xunta de Galicia/FEDER-UE, Spain [grant numbers ED431G 2019/01, ED481A 2020/003, ED431C 2022/46, ED431C 2018/49 and ED431C 2021/53]. Funding for open access charge: Universidade da Coruña/CISUG.

Appendix. Supplementary materials

The complete dataset, as well as the scripts used to preprocess its data, are available online at http://lbd.udc.es/research/real-life-HAR-dataset. Similarly, these have also been uploaded to Mendeley Data [59]. Additionally, the code used to carry out the experiments in this work is available online at http://gitlab.lbd.org.es/dgarcia/new-machine-learning-har.

References

[1] E. Kim, S. Helal, D. Cook, Human activity recognition and pattern discovery, IEEE Pervasive Comput. 9 (1) (2009) 48–53.
[2] J.K. Aggarwal, L. Xia, Human activity recognition from 3d data: A review, Pattern Recognit. Lett. 48 (2014) 70–80.
[3] E. Soleimani, E. Nazerfard, Cross-subject transfer learning in human activity recognition systems using generative adversarial networks, Neurocomputing 426 (2021) 26–34.

[4] C. Torres-Huitzil, A. Alvarez-Landero, Accelerometer-based human activity recognition in smartphones for healthcare services, in: Mobile Health, Springer, 2015, pp. 147–169.
[5] A. Zahin, R.Q. Hu, et al., Sensor-based human activity recognition for smart healthcare: A semi-supervised machine learning, in: International Conference on Artificial Intelligence for Communications and Networks, Springer, 2019, pp. 450–472.
[6] J. Manjarres, P. Narvaez, K. Gasser, W. Percybrooks, M. Pardo, Physical workload tracking using human activity recognition with wearable devices, Sensors 20 (1) (2020) 39.
[7] N. Zhu, T. Diethe, M. Camplani, L. Tao, A. Burrows, N. Twomey, D. Kaleshi, M. Mirmehdi, P. Flach, I. Craddock, Bridging e-health and the internet of things: The sphere project, IEEE Intell. Syst. 30 (4) (2015) 39–46.
[8] Y. Du, Y. Lim, Y. Tan, A novel human activity recognition and prediction in smart home based on interaction, Sensors 19 (20) (2019) 4474.
[9] O.D. Lara, M.A. Labrador, A survey on human activity recognition using wearable sensors, IEEE Commun. Surv. Tutor. 15 (3) (2012) 1192–1209.
[10] F. Demrozi, G. Pravadelli, A. Bihorac, P. Rashidi, Human activity recognition using inertial, physiological and environmental sensors: a comprehensive survey, 2020, arXiv preprint arXiv:2004.08821.
[11] M. Shoaib, S. Bosch, O. Incel, H. Scholten, P. Havinga, Complex human activity recognition using smartphone and wrist-worn motion sensors, Sensors 16 (4) (2016) 426.
[12] M.M. Hassan, M.Z. Uddin, A. Mohamed, A. Almogren, A robust human activity recognition system using smartphone sensors and deep learning, Future Gener. Comput. Syst. 81 (2018) 307–313.
[13] F. Attal, S. Mohammed, M. Dedabrishvili, F. Chamroukhi, L. Oukhellou, Y. Amirat, Physical human activity recognition using wearable sensors, Sensors 15 (12) (2015) 31314–31338.
[14] C. Xu, D. Chai, J. He, X. Zhang, S. Duan, Innohar: a deep neural network for complex human activity recognition, Ieee Access 7 (2019) 9893–9902.
[15] N. Lane, Y. Xu, H. lu, S. Hu, T. Choudhury, A. Campbell, F. Zhao, Enabling large-scale human activity inference on smartphones using community similarity networks (CSN), in: UbiComp'11 - Proceedings of the 2011 ACM Conference on Ubiquitous Computing, 2011, pp. 355–364.
[16] G. Weiss, J. Lockhart, The impact of personalization on smartphone-based activity recognition, in: AAAI Publications, Workshops At the Twenty-Sixth AAAI Conference on Artificial Intelligence, 2012.
[17] A. Ferrari, D. Micucci, M. Mobilio, P. Napoletano, On the personalization of classification models for human activity recognition, IEEE Access PP (2020) 1.
[18] R. Solis Castilla, A. Akbari, R. Jafari, B.J. Mortazavi, Using intelligent personal annotations to improve human activity recognition for movements in natural environments, IEEE J. Biomed. Health Inf. (2020) 1.
[19] D. Garcia-Gonzalez, D. Rivero, E. Fernandez-Blanco, M.R. Luaces, A public domain dataset for real-life human activity recognition using smartphone sensors, Sensors 20 (8) (2020) 2200.
[20] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones, in: Esann, 2013.
[21] J.R. Kwapisz, G.M. Weiss, S.A. Moore, Activity recognition using cell phone accelerometers, ACM SigKDD Explor. Newsl. 12 (2) (2011) 74–82.
[22] A. Ignatov, Real-time human activity recognition from accelerometer data using convolutional neural networks, Appl. Soft Comput. 62 (2018) 915–922.
[23] N. Sikder, M.S. Chowdhury, A.S. Arif, A.-A. Nahid, Human activity recognition using multichannel convolutional neural network, in: 2019 5th Int. Conf. Adv. Electr. Eng, 2019.
[24] S. Seto, W. Zhang, Y. Zhou, Multivariate time series classification using dynamic time warping template selection for human activity recognition, in: 2015 IEEE Symposium Series on Computational Intelligence, IEEE, 2015, pp. 1399–1406.
[25] W. Sousa, E. Souto, J. Rodrigres, P. Sadarc, R. Jalali, K. El-Khatib, A comparative analysis of the impact of features on human activity recognition with smartphone sensors, in: Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web, ACM, 2017, pp. 397–404.
[26] J. Figueiredo, G. Gordalina, P. Correia, G. Pires, L. Oliveira, R. Martinho, R. Rijo, P. Assuncao, A. Seco, R. Fonseca-Pinto, Recognition of human activity based on sparse data collected from smartphone sensors, in: 2019 IEEE 6th Portuguese Meeting on Bioengineering, ENBENG, IEEE, 2019, pp. 1–4.
[27] R.-A. Voicu, C. Dobre, L. Bajenaru, R.-I. Ciobanu, Human physical activity recognition using smartphone sensors, Sensors 19 (3) (2019) 458.
[28] Z. Chen, Q. Zhu, Y.C. Soh, L. Zhang, Robust human activity recognition using smartphone sensors via CT-PCA and online SVM, IEEE Trans. Ind. Inform. 13 (6) (2017) 3070–3080.
[29] C.A. Ronao, S.-B. Cho, Human activity recognition with smartphone sensors using deep learning neural networks, Expert Syst. Appl. 59 (2016) 235–244.
[30] F. Hernández, L.F. Suárez, J. Villamizar, M. Altuve, Human activity recognition on smartphones using a bidirectional LSTM network, in: 2019 XXII Symposium on Image, Signal Processing and Artificial Vision, STSIVA, IEEE, 2019, pp. 1–5.
[31] M. Badshah, Sensor-based human activity recognition using smartphones, 2019.
[32] S. Wan, L. Qi, X. Xu, C. Tong, Z. Gu, Deep learning models for real-time human activity recognition with smartphones, Mob. Netw. Appl. (2019) 1–13.
[33] W. Qi, H. Su, C. Yang, G. Ferrigno, E. De Momi, A. Aliverti, A fast and robust deep convolutional neural networks for complex human activity recognition using smartphone, Sensors 19 (17) (2019) 3731.
[34] Q. Teng, K. Wang, L. Zhang, J. He, The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition, IEEE Sens. J. 20 (13) (2020) 7265–7274.
[35] Y.E. Ustev, O. Durmaz Incel, C. Ersoy, User, device and orientation independent human activity recognition on mobile phones: Challenges and a proposal, in: Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication, ACM, 2013, pp. 1427–1436.
[36] V. Janko, N. Rešçiç, M. Mlakar, V. Drobnič, M. Gams, G. Slapničar, M. Gjoreski, J. Bizjak, M. Marinko, M. Luštrek, A new frontier for activity recognition: The sussex-huawei locomotion challenge, in: Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers, 2018, pp. 1511–1520.
[37] S. Rosati, G. Balestra, M. Knaflitz, Comparison of different sets of features for human activity recognition by wearable sensors, Sensors 18 (12) (2018) 4189.
[38] D. Nielsen, Tree boosting with xgboost-why does xgboost win" every" machine learning competition?, (Master's thesis), NTNU, 2016.
[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[40] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[41] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.
[42] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (2004) 101–141.
[43] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.
[44] J.R. Quinlan, C4. 5: Programs for Machine Learning, Elsevier, 2014.
[45] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC Press, 1984.
[46] H. Taud, J. Mas, Multilayer perceptron (MLP), in: Geomatic Approaches for Modeling Land Change Scenarios, Springer, 2018, pp. 451–455.
[47] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[48] I. Rish, et al., An empirical study of the naive Bayes classifier, in: IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence, Vol. 3, (22) 2001, pp. 41–46.
[49] K.P. Murphy, et al., Naive Bayes Classifiers, 18, (60) University of British Columbia, 2006.
[50] L.E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.
[51] P. Cunningham, S.J. Delany, K-nearest neighbour classifiers–, 2020, arXiv preprint arXiv:2004.04523.
[52] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[53] S. Athey, J. Tibshirani, S. Wager, et al., Generalized random forests, Ann. Statist. 47 (2) (2019) 1148–1178.
[54] M. Hossin, M. Sulaiman, A review on evaluation metrics for data classification evaluations, Int. J. Data Min. Knowl. Manag. Process 5 (2) (2015) 1.
[55] M. Grandini, E. Bagli, G. Visani, Metrics for multi-class classification: an overview, 2020, arXiv preprint arXiv:2008.05756.
[56] M. Bekkar, H.K. Djemaa, T.A. Alitouche, Evaluation measures for models assessment over imbalanced data sets, J. Inf. Eng. Appl. 3 (10) (2013).
[57] R. Kohavi, et al., A study of cross-validation and bootstrap for accuracy estimation and model selection, in: Ijcai, Vol. 14, (2) Montreal, Canada, 1995, pp. 1137–1145.
[58] P. Liashchynskyi, P. Liashchynskyi, Grid search, random search, genetic algorithm: A big comparison for nas, 2019, arXiv preprint arXiv:1912.06059.
[59] D. Garcia-Gonzalez, D. Rivero, E. Fernandez-Blanco, M. R. Luaces, A public domain dataset for real-life human activity recognition using smartphone sensors, 2020, Mendeley Data, V2, Available online: https://data.mendeley.com/datasets/3xm88g6m6d/2.