Assessing sheep behavior through low-power microcontrollers in smart agriculture scenarios

Luís Nóbrega*

DETI/IT - Universidade de Aveiro, Portugal

Pedro Gonçalves

ESTGA/IT - Universidade de Aveiro, Portugal

Mário Antunes

ISCAA/IT - Universidade de Aveiro, Portugal

Daniel Corujo

DETI/IT - Universidade de Aveiro, Portugal

Abstract

Automatic animal monitoring can bring several advantages to the livestock sector. The emergence of low-cost and low-power miniaturized sensors, together with the ability of handling huge amounts of data, has led to a boost of new intelligent farming solutions. One example is the SheepIT solution that is being commercialized by iFarmtec. The main objectives of the solution are monitoring the sheep's posture while grazing in vineyards, and conditioning their behaviour using appropriate stimuli, such that they only feed from the ground or from the lower branches of the vines. The quality of the monitoring procedure has a linear correlation with the animal condition capability of the solution, *i.e.*, on the effectiveness of the applied stimuli. Thus, a Real-Time mechanism capable of identifying animal behaviour such as infraction, eating, walking or running movements and standing position is required. On a previous work we proposed a solution based on low-power microcontrollers enclosed in collars wearable by

*Corresponding author
Email address: `luisnobrega@av.it.pt` (Luís Nóbrega)

sheep. Machine Learning techniques have been rising as a useful tool for dealing with big amounts of data. From the wide range of techniques available, the use of Decision Trees is particularly relevant since it allows the retrieval of a set of conditions easily transformed in lightweight machine code.
The goal of this paper is to evaluate an enhanced animal monitoring mechanism and compare it to existing ones. In order to achieve this goal, a real deployment scenario was availed to gather relevant data from sheep's collar. After this step, we evaluated the impact of several feature transformations and pre-processing techniques on the model learned from the system. Due to the natural behaviour of sheep, which spend most of the time grazing, several pre-processing techniques were tested to deal with the unbalanced dataset, particularly resorting on features related with stateful history. Albeit presenting promising results, with accuracy over 96%, these features resulted in unfeasible implementations. Hence, the best feasible model was achieved with 10 features obtained from the sensors' measurements plus an additional temporal feature. The global accuracy attained was above 91%. Howbeit, further research shall assess a way of dealing with this kind of unbalanced datasets and take advantage of the insights given by the results achieved when using the state's history.

Keywords: Sheep, Animal behaviour, Machine Learning, Decision Trees, Microcontrollers

1. Introduction

The Internet of Things (IoT) [1] has been reducing the gap between the digital and physical worlds, leveraging powerful tools such as the miniaturisation of electronic components and the democratization of computational resources through the usage of cloud-based services. This has brought digital capabilities and transformation into a continuously growing number of societal sectors.

Monitoring and conditioning animal behaviour is a relevant task in the livestock sector, crucial for the sustainability [2] of the associated enterprises. On the one hand, supervising animals allows the enhancement of the welfare control processes, avoiding unnecessary casualties and allowing a more efficient reproductive process [2, 3]. On the other hand, it provides a better management of available pastures [4], and, consequently, natural resources. Typically, these tasks are performed by a human operator, susceptible to several limitations [5]: it is a scarce, expensive and jejune resource; it is prone to observation errors; and its presence may clout animal behaviour. The emergence of low-cost and low-power sensing devices, and their ability to be connected to the Internet, allowed the development of several commercial solutions [6, 7, 8, 9] that changed the way the livestock sector operates. Hence, devices capable of detecting animal behaviour and activity, started to be commonly available, allowing the extraction of huge amounts of data from individual animals and, consequently, from the entire herd.

The SheepIT project [10], a R&D project, also addressed such an issue, particularly through the development of a solution capable of both monitoring and conditioning the sheep's posture. The main purpose of the system is to create an automated mechanism that would allow sheep to weed vineyards without threatening the vines or fruits. The solution comprised the design of a complete IoT system, from the sensing devices to the Cloud platform, including all the infrastructural nodes needed to relay the data from sensors to the Cloud [11].

Regarding the sensing components of the system, monitoring and conditioning the sheep's posture are fundamental. The monitoring capabilities of the system allow the distinction of the sheep behaviour. The conditioning capabilities of the system stimulate sheep to revert from unwanted and threatening behaviour, namely through the use of suitable stimuli [1] Regarding the monitoring mechanism developed in the scope of the project, previous works explored the use of Machine Learning (ML) techniques to analyze the data gathered, with the goal of defining a suitable and feasible monitoring mechanism capable of running on a low-power microcontroller [12, 13]. This mechanism allows the distinction of a new behaviour, not previously addressed in the literature,

[1]The conditioning mechanism is out of the scope of the present paper.

the *infracting* state. This state represents behaviours where a sheep is likely to be feeding from higher branches of vines or fruits, which the system's solution intends to avoid. However, the reported mechanism assumes a simplified behaviour model where, for instance, walking and running were identified as being the same state [12]. Furthermore, the implemented mechanism presented limitations when detecting the *infracting* state, incurring in a considerable number of false positives and false negatives. Such limitation represents a strong weakness of the system since it carries unpredictable consequences in animal conditioning, particularly regarding the animal learning capabilities.

Therefore, the present paper explores and discusses the strategies towards an enhanced monitoring mechanism through the use of ML techniques. We focused on the use of Decision Trees (DT), due to their learning capabilities and readability. The DT model can be easily parsed into machine code, making it ideal for running on energy and computational constrained devices. Particularly, we targeted the differentiation of the following animal states: *infracting* (the distinctive state), *eating*, *moving*, *running* and *standing*.

The remainder of the paper is organized as follows. Section 2 explores the state-of-art on animal monitoring platforms, with a special highlight on how ML algorithms have been used on such context. Section 3 describes all the material and methods used, detailing how the data was collected and classified, which features were treated, selected and in which way, how models were built and how the results were evaluated. Section 4 presents the results obtained. A critical reflection on all the experiments and respective results are presented in Section 5, before concluding the paper in Section 6.

2. Related Work

Machine Learning (ML) has been used in different domains such as crop management, water management, soil management and livestock management [14]. In the latter, animal welfare monitoring [15], reproductive cycles optimization [16, 17, 18] and pasture management [19] are just some examples where

ML already plays an important role.

Many of these applications gather data from sensing devices placed on animals, capable of identifying behaviours (mainly using accelerometry) or locations (mostly using GPS). Regarding behaviour monitoring, despite some solutions that allow an almost Real-Time monitoring [20, 21] (transmitting the sensing data to computational platform), most employ an off-line monitoring, storing the sensing data in the memory of devices for further analysis using powerful computers [22].

This kind of monitoring platforms have been applied to different animals, as for instance horses [23], goats [24], sheep [25, 26, 27, 22, 28, 29, 30, 31, 32, 33], and cattle [34, 35, 36, 37], just to cite a few. Among these, solutions for cattle stand out, much due to the economic impact of that sector. As a consequence, several commercial solutions have already penetrated the market, as in the cases of CowScout [6], Cowlar [7] and MooMonitor [38], whose main goal is cattle activity monitorization towards improved lactation and insemination processes.

Albeit not being as popular as cattle monitoring platforms, sheep monitoring has gained a fresh impetus with some relevant works. These works differ mainly in how the monitoring device is coupled to sheep (ear tag, collar, leg tag or halter), the number and type of states to be classified, the features used in the model and the sampling rate of sensors.

In [28], the authors aim at monitoring jaw movements in order to detect grazing, ruminating and resting states. For that, they resort on a halter coupled to the sheep containing an accelerometer operating at a frequency of 62.5Hz and considering an observation window of 60s. To model the predicting system, they employed Discriminant Analysis (DA) using 12 different features, achieving an overall accuracy of 92,90%. Still related with this work, in [29], an evaluation of different observation window sizes was discussed, being the 30s window size the one that presented the best results for that particular system.

Walking, standing and lying were the states evaluated in [30, 31]. Accelerometer and gyroscope measurements were considered, using either a collar or an ear tag. In the former work, different frequencies (8Hz, 16Hz and 32Hz) and different observation window sizes (3s , 5s and 7s) were assessed applying Random Forest (RF) models. The best results were achieved both for ear tag and collar for the 32Hz and 5/7s configurations, with an overall accuracy of 95%. However, the authors concluded that, for Real-Time applications, the configuration of 16Hz and 7s configuration would be preferable due to the energy expenditure associated (overall accuracy of 93%). In the latter work, having as starting point the 16Hz and 7s configuration, a discussion on the impact of using different features (from a list of 44 features) and different ML algorithms was done, being the best results achieved with the RF algorithm and for the collar use case (overall accuracy of 92%).

The differences on using a collar, an ear tag or a leg tag were examined in [32]. Using only accelerometer measurements with a frequency of 12Hz and observation windows of 10s, 4 states were considered: grazing, walking, standing and lying. A Quadratic Discriminant Analysis (QDA) was employed using 14 features, being obtained overall accuracies of 86,83% for the collar use case.

In [33], the authors applied Linear and Quadratic Discriminant Analysis (LDA and QDA) to predict 5 different states (lying, standing, walking, running and grazing), achieving an overall accuracy of 89.69%. They also resorted on accelerometer data gathered with a frequency of 100Hz and observation windows of 5.12s.

Also considering the same 5 states tackled in [33], the work presented in [22] explored the use of DT to predict those states. The data was gathered through accelerometers placed under the sheep's jaw, operating at three different frequencies: 5Hz, 10Hz and 25Hz. Besides presenting a total of 44 features, the top-5 features were selected during the construction of the model. In the end, the overall accuracies reached from 82.9% for observation windows of 3s, to 85.5% for 5s and to 83.4% for 10s.

From the reported works, the ones that present solutions that could be eventually transposed to a low-power micro-controller, have limited their analysis to a low number of states (3). Contrary, the works that widen their analysis to more states, exploited algorithms that are computationally more demanding and hence not suitable to be employed in low-power microcontrollers. The exception is the work of [22], that explored the use of DT for differentiating 5 different states but, as all the other works, it did not distinguish the *infracting* state, a critical element in the SheepIT project. In [12], a first approach based in ML techniques was presented to tackle such an issue. In brief, the problem was pruned to a binary classification problem, being the two states defined as *infracting* and *not infracting*. The work explored different ML algorithms (including DT), being the accuracies obtained between 95% and 97%.

However, besides reporting a poor dataset, its feasibility when taken to a real implementation has revealed critical drawbacks. As detailed in [13], the resulting models were only based in the angle of the sheep's neck, which may entail significant problems. Two examples are: i) the sheep is standing, with its head up but without eating; and ii) the sheep is moving or running with its head up. To minimize this concern, the work presented in [13] split the problem into three binary classification problems. Specifically, the authors provided a different model for each of the following binary classifications: i) *infracting* and *not infracting - active state*; ii) *resting* and *not resting*; and iii) *running* and *not running*. Implementing DT, they achieved individual accuracies of 96%, 81% and 96%, respectively. In the end, the models were merged to construct an unique DT.

Besides solving some of the identified issues, this latter work still presented several limitations. On the one hand, it was also based in a poor dataset, with information about a single sheep. On the other hand, it did not allow the identification of *moving* behaviours, particularly when sheep move between different grazing areas. Such behaviour also presents (as the running state) characteristics that can approach the *infracting* state features, thus, being a potential point of failure of the system. Henceforth, the present work aims at minimizing the limitations of the current posture monitoring implementation, also taking advantage of ML techniques.

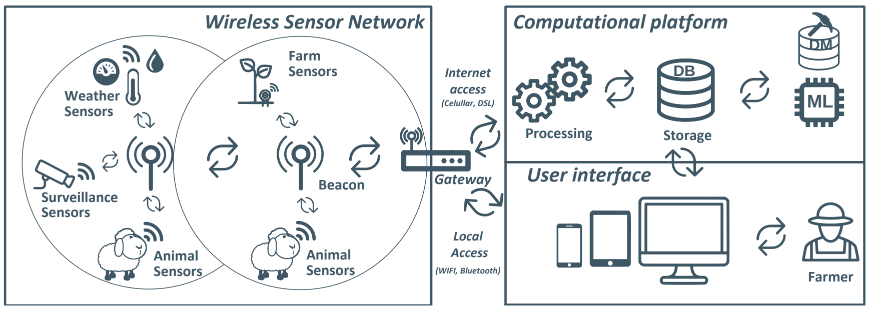

Figure 1: Overall system architecture [11]

3. Material and methods

The present section describes the experimental methodology, starting from the devices used, to the procedures followed to reach the identified goal: provide an enhanced monitoring mechanism, to be implemented in a low-power microcontroller, and that shall be capable of a Real-Time detection of different sheep behaviours such as *infracting*, *grazing*, *moving*, *running* and *standing*.

3.1. The monitoring platform

The present work was based in a platform that is being commercialized by iFarmTec [39] for sheep monitoring and control. This solution was based in the prototype developed in the scope of the SheepIT project [10], although several improvements were added.

As depicted in Figure 1, the system's architecture is composed of three main modules. A Wireless Sensor Network (WSN) responsible for the implementation of all the required local tasks (*e.g.* monitoring sheep, triggering the conditioning mechanism when necessary and uploading the sensing data), a Computational Platform (CP) able to be hosted in the cloud and that is responsible for receiving, storing and analysing the data sent by the WSN; and a User Interface (UI) for a smooth interaction with end users. For the purpose of the present work, it only becomes relevant to review the architecture and operation of the WSN.

The devices that implement the sheep monitoring feature are called collars, being placed around sheep's neck. Such devices are composed by a set of sensors and actuators controlled by a microcontroller. They are powered by a Li-Ion battery and communicate with the remaining WSN devices through a free-licensed radio The main sensor included is a three-axial accelerometer [40] that, besides being capable of measuring both static and dynamic acceleration in all the three axis, it also includes filtering capabilities, sampling configuration and an internal buffer for storing temporary measurements. In addition to the the accelerometer, the collar also implements a ultrasound transducer responsible for the measurement of the distance from the sheep's neck to the ground.

The relationship between sensors and actuators is defined by a posture control algorithm that may be seen as two interacting algorithms: a monitoring algorithm, whose performance is intended to be improved with the present work; and a conditioning mechanism. This posture control algorithm is continuously running in the microcontroller, ensuring a prompt and correct sheep monitoring and conditioning. However, the report of all the data generated in the collars is unattainable, both in terms of energy and bandwidth expenditure. In fact, the system's communication network was designed following a Time-Triggered paradigm [41], which means that all communications are executed following a predetermined scheduling. Hence, the gathered data is only periodically sent from collars to a set of statically placed devices (called beacons), that henceforth relay the data through the network towards a gateway. A more detailed description of the communication network can be found in [11].

3.2. The experiment

To gather the necessary data towards an improved behaviour monitoring mechanism, the existing monitoring platform was used, with the data was fetched from a deployed system in a livestock farm of a client of iFarmTec. The sheep were recorded using a video camera at the same time the data was being collected.

The grazing area was a vineyard of the Bairrada's region (center of Portugal) with around 2.5ha, with the vine lines separated by around 3m and with a fence surrounding the whole area, ensuring adequate and secure grazing. Typically, sheep graze on this parcel throughout all the year and are only taken out for a few weeks during the budburst phase, *i.e.,* in the beginning of spring, to prevent the destruction of the fragile vine buds.

The experiment was conducted over a period where the flock grazed under daily conditions and lasted for 3 days (from the 15th to the 17th of May 2019, hence, well after the budburst phase). Furthermore, each testing day was split into two test windows: morning (from 09:00 to 12:00) and afternoon (from 13:00 to 17:00). During these periods, sheep were grazing as usual with the iFarmtec system incorporated, with one different animal being video-recorded per test window to simplify the classification process. To ensure a correct synchronization between the timestamp of the video and the gateway responsible for storing the data, the video recording device and the gateway were temporally synchronized. This procedure allowed to have a perfect match between each sensing data entry and the video frames filmed.

3.3. Data collection

The collection of data was done using the monitoring platform described in Section 3.1. The configuration of the system deployed on the livestock farm allows the collection of a new trio dynamic acceleration values (each trio is composed by the x, y and z components of the dynamic acceleration vector) every 20ms (*i.e.* with a frequency of 50Hz). Nevertheless, each new value is not immediately sent through the wireless network, but stored in the internal buffer of the accelerometer until it gets completely filled (the buffer allows the storage of 25 trios). Whenever the buffer is filled, the stored data, that included the data concerning the dynamic acceleration measurements, a sample of the static acceleration on the three axis (and the respective pitch, roll and yaw angles) and the distance from the sheep's neck to the ground, were sent through the wireless network until reaching the system's gateway. Here, the data was decoded and stored for further processing.

3.4. Data classification

To enable the implementation of supervised learning techniques, it was necessary to create conditions for classifying the generated data. Hence, all test windows were video-recorded using a open source camera application for Android [42] that automatically generates a subtitle file with the associated current timestamp.

Concluded all the data collection proceedings, the resulting files (one per test window) were downloaded from the gateway. The subsequent phrase concerned the data classification procedure, where the videos associated with each test windows were watched and the observations classified following the dictionary described in Table 1. Dubious observations or observations that couldn't be classified (*e.g.* an animal out of sight) were classified with an X to be ignored. The summary of the observations can be found in Table 2. Most of the dataset observations are of the type *eating* (70%), which was expected since animals tend to be grazing most of the time. The less representative states are *infracting* (3%), *running* (2%) and *standing* (2%), which was also expected since sheep were free to pasture on a area with a lot of edible weed and without being exposed to external dangers (and thus not having the need to run away). Summing up, we are in the presence of an unbalanced dataset with a total of 12968 valid experiments, being 1675 observations discarded (approximately 12 %).

3.5. Feature transformation

All data processing and analysis was carried out using RStudio [43].The first step consisted of importing the data and verifying its integrity. For that purpose all features were summarized and their type, minimum, average and maximum values were checked. The available features were: i) timestamp - added in the gateway; ii) classification - added manually during the data classification; and iii) distance measured using the ultrasounds transducer (*dist.mm)*, the static acceleration on axis x, y and z (*acc_x*, *acc_y* and *acc_y*), the respective angles

Table 1: States differentiated during sheep behaviour classification

State	**Description**
Infracting (I)	Eating from branches above a certain height. For this test, it was defined the height of the irrigation tubes (˜50cm)
Eating (E)	Sheep is eating with its head down. Smooth movements were allowed since typically they seek for grass while moving
Moving (M)	There is a notorious and intended movement from one place to another. Typically, while this happens, the sheep is not seeking for food. Trotting was considered moving.
Running (R)	Sheep was running (running away from an obstacle or trying to reach the remaining herd)
Standing (S)	The head is still, *i.e.,* sheep is steady and still
Invalid (X)	Dubious observations or unclassifiable observations

Table 2: Dataset summary

State	E	M	I	S	R	X	Total
Number of observations	10305	1686	488	260	229	1675	14643

(*pitch*, *roll* and *yaw*) and the 25 measurements of the dynamic acceleration (on all the 3 axis) - coming directly from the collar.

Two distinct types of features are hence identified: static and dynamic. Static features are single values, *i.e.*, for each timestamp the features take a single value. Dynamic features are a sequence of values, *i.e.*, for each timestamp, the features have a sequence of values. This latter kind of features was added to capture the impact of movements and rapid accelerations. Although the dynamic features could be used as a whole, there are several benefits to summarize them into single values. Firstly, it enables the control of the number of features. Secondly, it enables the reduction of the size of the resulting model. In this specific scenario, the ML model has to run on a low-power microcontroller, thus constraining the complexity of the resulting model.

Taking advantage of the provided state-of-the-art, 8 main features were selected for application, both to the magnitude (Equation 1) of each dynamic acceleration vector and to the rate of change (1st derivative) of that magnitude (Equation 2)[2]. The description and formulas of all the features are detailed in Table 3.

$$\forall i \in [1, N], |dynA_i| = \sqrt{dyn\_x_i^2 + dyn\_y_i^2 + dyn\_z_i^2} \tag{1}$$

$$\forall i \in [1, N-1], |DdynA_i| = dynA_{i+1} - dynA_i, \tag{2}$$

where N is the number of measurements per observation.

As identified in Section 3.4, the dataset is clearly unbalanced due to the sheep natural behaviour. Additionally, a preliminary analysis of the dataset revealed a potential spatial and temporal relationship between states. Therefore, there was a need for seeking for additional features that, firstly, could confirm such relationship and secondly, that could support the model for identifying such relationships. Thus, three additional features were added to the already identified 24 features: i) the previous state (*prevState*); ii) the identification if there was a transition in the previous state *(transition)*; and iii) the number of consecutive and equals states (*nEqualStates*). In short, the dataset ended up with a total of 27 different features.

3.6. Feature Selection

The use of a high number of features is nowadays possible due to the available increased computational capability, either as standalone machines or using distributed computation. Nonetheless, the same is not true when dealing with constrained devices such as the microcontroller incorporated into the system's collar. Hence, it wouldn't be reasonable or even feasible to implement all the fea-

[2]The movement variation was calculated directly using the dynamic acceleration components (x,y and z axis), being the unique feature that was neither related to the magnitude, neither related to the 1st derivative.

Table 3: Features added to the existing dataset. The shown formulas are applied to acceleration magnitude values. A similar approach is followed for the 1st derivative, being only necessary to change N by N-1 and $dynA_i$ by $DdynA_i$. N is the number of measurements.

Feature	**Formula**
Minimum	Minimum value within the observation
Maximum	Maximum value within the observation
Average	$\frac{1}{N}\left(\sum_{i=1}^{N} dynA_i\right)$
Variance	$\sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}(dynA_i - \mu)^2$
Standard Variation	$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(dynA_i - \mu)^2}$
Zero Crossing	Number of crossings through zero after subtracting each observation value by the observation mean
Dominant frequency	Powerful frequency after appyling Fourier Transformation
Movement variation [1]	$\frac{1}{N}(\sum_{i=1}^{N-1} \lvert dyn\_x_{i+1} - dyn\_x_i\rvert + \sum_{i=1}^{N-1} \lvert dyn\_y_{i+1} - dyn\_y_i\rvert + + \sum_{i=1}^{N-1} \lvert dyn\_z_{i+1} - dyn\_z_i\rvert)$

tures herein described in the collar. Consequently, feature selection represents a specially relevant task in the present work.

Different feature selection methods can be found in the literature [44], most of them with existing implemented solutions. Thereupon, the approach followed in the present work was based in two main phases. Firstly, the correlation between the features was evaluated such that the most correlated ones could be removed. Then, some algorithms available on the *FSelector* package [45] were used to identify the most promising features. This referred package contains several functions that enable an easily processing of feature selection. From the available list, the ones selected are described in Table 4. The evaluation of such feature selection mechanisms was based in two different tasks:

1. **Repercussion in the total accuracy:** the repercussion on using each one of the techniques on the accuracy of a model was evaluated;

Table 4: Feature selection functions selected from the *FSelector* package [45]

Function Name	**Description**
CFS Filter	Resorts on the correlation and entropy measurements
Chi-squared filter	Weights discrete features using a chi-squared test
Consistency-based filter	Measures the consistency of the features
OneR algorithm	Associates rules between features and the class feature
RandomForest filter	Weights features applying the RandomForest algorithm
RReliefF filter	Weights features through the distance between instances

2. **Consistency of the selected attributes between techniques:** after discarding some techniques that could have negative impact, the repeatability of the most important features between techniques was assessed.

3.7. Modelling

Implementing a continuous monitoring mechanism on a low-power and low-processing microcontroller restrains the kind of ML algorithms that can be used. Thus, the present work focused entirely on the application of DT algorithms. On one hand, because they allow the extraction of a set of conditions, easily transposed to a set of *if's* and *elses's*, that can be efficiently incorporated in a constrained device, such as the system's collar. On the other hand, they allow a straightforward human interpretation. This latter condition is particularly relevant in the present use case since the animal behaviour is not always straightforward and predictable as it seems, thus requiring further human analysis.

As several tests were performed, a methodical strategy was followed, simplifying the collection and comparison of results. The following list describes such steps:

1. **Preparation of the training dataset:** two different strategies were followed, accordingly to the stage of the ML process. During the feature selection phrase, where several tests needed to be performed, a simple

random split was done in a proportion of 70% for training and 30% for testing. This simplistic process, allowed a faster identification of the scenarios with the highest potential to be carefully evaluated. Then, after having selected the most promising scenarios (from the feature selection phrase), their evaluation was done employing a split of data that could enable the use of a cross-validation strategy. Particularly, a 10-fold cross-validation was implemented, which means that the entire dataset was split into 10 subsets of similar size. Additionally, it shall be noted that the distribution of the classification labels was maintained roughly the same as the existing on the original dataset;

2. **Selection of features to be used by the model:** the feature selection techniques described in Section 3.6 were experienced;
3. **Application of sampling techniques:** as the dataset is unbalanced, upsampling and downsampling techniques were experimented;
4. **Model training**: the model was trained using the training dataset and the features selected. The *rpart* package was used;
5. **Model tuning**: the *rpart* function has two main tuning parameters, the *complexity parameter (cp)* and the *max depth* parameter. The former, represents how much the relative error is incremented when splitting a node. The later, represents the maximum number of split levels;
6. **Prediction evaluation**: to assess the prediction model, several metrics were used, being described in Section 3.8.

3.8. Prediction evaluation

To evaluate models, two types of tools were used. Firstly, the construction of a confusion matrix to easily visualize and interpret the prediction summary, namely the number of *True negatives*, *False Negatives*, *False Positives* and *True Positives.* Secondly, the evaluation of some performance metrics commonly used in classification problems in ML, for instance, *accuracy*, *micro* and *macro averages* of the *F1 score* [46] and the *K-category correlation coefficient* [47], a multiclass extension of the *Matthews Correlation Coefficient* [48]. The *accuracy* is used since it is a general and very common metric used in ML problems, found in almost every works of the state of art. However, as the dataset discussed is particularly unbalanced, there are two metrics that assume a higher importance, namely the *macro-average F1 Score* and the *K coefficient*. Both provide a more credible performance of a model in an imbalanced dataset. The respective equations are illustrated in Equations 3, 4, 7 and 10.

The most promising scenarios identified through the feature selection phase, were modeled using 10-fold cross validation. The goal with this strategy was to test if the model being tested would be capable of being generalized to other datasets, minimizing the chances of creating bias points in the dataset. This technique has natural consequences on the evaluation phase since, when implementing crossing validation, we need to model and test the same number of times as the number of folds. This means that when testing a scenario that implements a 10-fold crossing validation, it results in 10 confusion matrices and 10 values for the same metric. To summarize the results associated to each scenario, the confusion matrices were summed and the average of the values for each metric were considered.

$$OverallAccuracy = \sum_{k=1}^{N} \frac{TP_k}{TP_k + TN_k + FP_k + TN_k} \tag{3}$$

were N is the number of classes, *i.e.,* animal states.

$$F1_{micro} = 2\frac{P_{micro} \times R_{micro}}{P_{micro} + R_{micro}} \tag{4}$$

where P_{micro} and R_{micro} are given by equation 5 and equation 6, respectively.

$$P_{micro} = \frac{\sum_{k=1}^{N} TP_k}{\sum_{k=1}^{N} TP_k + FP_k} \tag{5}$$

$$R_{micro} = \frac{\sum_{k=1}^{N} TP_k}{\sum_{k=1}^{N} TP_k + FN_k} \tag{6}$$

$$F1_{macro} = \frac{2 \times P_{macro} \times R_{macro}}{P_{macro} + R_{macro}} \tag{7}$$

where P_{macro} and R_{macro} are given by equation 8 and equation 9, respectively.

$$P_{micro} = \frac{1}{N} \sum_{k=1}^{N} \frac{TP_k}{TP_k + FP_k} \tag{8}$$

$$R_{micro} = \frac{1}{N} \sum_{k=1}^{N} \frac{TP_k}{TP_k + FN_k} \tag{9}$$

$$R_K = \frac{\sum_{klm} C_{kk}C_{lm} - C_{kl}C_{mk}}{\sqrt{\sum_k \left(\sum_l C_{kl}\right)\left(\sum_{l',k' \neq k} C_{k'l'}\right)}\sqrt{\sum_k \left(\sum_l C_{lk}\right)\left(\sum_{l',k' \neq k} C_{l'k'}\right)}} \tag{10}$$

where C is the confusion matrix, C_{kk} is the number of correctly predicted observations concerning class k and C_{kl} are the observations predicted as k but belonging to class l ($l \neq k$).

4. Results and discussion

To achieve the desired goal, several tests were made following the methodical approach described in Section 3.7. As it is impracticable to present and discuss all of them in the scope of the present paper, we focused on the ones that could provide improved monitoring capabilities to the system. Henceforth, the present section presents and discusses the main results of the feature selection techniques as well as the most relevant models tested and obtained.

4.1. Feature Selection

The dataset available and described on the later section presents a total of 27 features, 7 directly generated by the collar, 2 added in the gateway, 15 obtained after feature transformation of dynamic acceleration measurements, and 3 additional features added during data processing regarding state's history.

The first approach to reduce the number of attributes was an evaluation of the correlation between attributes, since some redundancy was expected. The correlation matrix obtained is represented in Figure 2. From the results obtained, the features whose correlation module scored above 0.9 were selected for isolation. A lower value would also be acceptable but, as additional feature selection techniques were projected to be further applieded, a conservative value was chosen. From the most correlated features, an individual analysis was performed, being removed the features that presented a higher number of correlated features (*e.g. stdDeviationD* is highly correlated with *varianceD, maxD, minD, stdDeviationM, meanM* and *maxM*, hence it was removed). This process culminated with the removal of the following features: static acceleration on x-axis (acc_x), maximum value of the dynamic acceleration magnitude ($maxM$), standard deviation of the first derivative ($stdDeviationD$), variance of the dynamic acceleration magnitude ($varianceM$) and movement variation (MV).

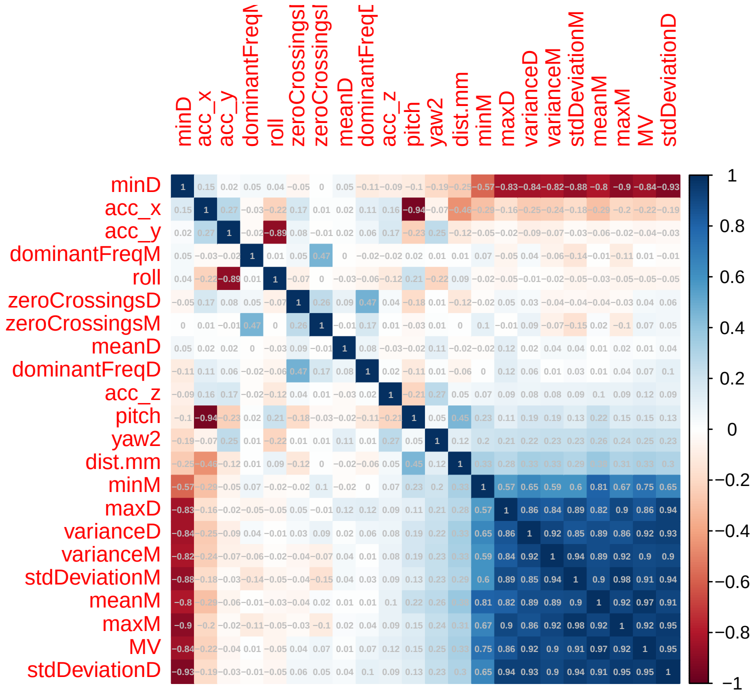

Figure 2: Correlation between features obtained after feature transformation

The second phase comprised the consistency evaluation of the feature selection techniques selected and described in Section 3.6. For that, the overall accuracy of the models obtained using the different techniques was assessed.

For a better understanding of the effects of using the three features related to the state's history, the feature selection techniques were experimented with and without these three features. Table 5 presents the order and features selected for each technique and the respective global accuracy (using equation 3), after running a decision tree algorithm and not considering the historical features. The analysis of the results allows the following conclusions:

- The results obtained using the *consistency-based filter* were automatically discarded since the order of the features obtained was exactly the same as the input order of the attributes, thus, not entailing valuable information;
- The *dist.mm* and *pitch* features are the most important features for modelling the system. Besides being always in the top four features among

Table 5: Feature Selection evaluation without stateful features

Technique/order	**CFS**	**Chi Squared**	**oneR**	**Random Forest**	**RReliefF**
1	dist.mm	meanM	dist.mm	pitch	pitch
2	pitch	pitch	pitch	dist.mm	zeroCrossingM
3	-	dist.mm	meaM	meanM	dominantFreqD
4	-	varianceD	varianceD	varianceD	dominantFreqM
5	-	stdDeviationM	stdDeviationM	stdDeviationM	meanM
6	-	minD	minM	minD	zeroCrossingsD
7	-	minM	minD	maxD	acc_z
8	-	maxD	maxD	minM	stdDeviationM
9	-	acc_z	acc_y	acc_z	acc_y
10	-	yaw	acc_z	yaw	varianceD
11	-	acc_y	yaw	acc_y	maxD
12	-	roll	meanD	roll	roll
13	-	dominantFreqD	roll	meanD	yaw
14	-	meanD	zeroCrossingsM	zeroCrossingsD	dist.mm
15	-	zeroCrossingsD	dominantFreqM	zeroCrossingsM	meanD
16	-	dominantFreqM	zeroCrossingsD	dominantFreqM	minD
accuracy	**88.51%**	**90.4%**	**90.46%**	**90.46%**	**90.46%**

all feature selection techniques, it is visible that the CFS algorithm only chooses these two features, neglecting all the remaining ones and maintaining an accuracy of 88.51%, only 1.94% below the best case;

- Splitting the features list into four groups (named from now on, tiers of features), it is possible to verify that the *Chi Squared* filter, the *oneR* algorithm and the *Random Forest* filter, present nearly all the same features inside the four tiers, although feature permutations may be found within them. These results show potential consistency between these methods;

- Regarding the *RReliefF* algorithm, a different distribution of features throughout the four tiers can be observed when comparing to the remaining ones, including in the top five features. Here, three of them (*zeroCrosingsM, dominanteFreqD* and *zeroCrossingsD*) were in the end of the list obtained for the other techniques/algorithms. This situation did not affect the global accuracy, probably because despite being selected by the feature selection techniques they were being disregarded by the model. However, this algorithm presented a higher variability when compared to the remaining ones, *i.e*, when considering different samples from the training dataset, relevant changes were noted in the feature ranking. Thus, considering the referred issue, together with the strong computational demands required to implement features such as the *dominantFreqD*, led to the disregard of the *RReliefF* algorithm;

- From the analysis of the ranking list of features obtained using the remaining algorithms (discarding the results from the *RReliefF* algorithm), a high level of concordance was detected in all top ten features. Thus, the total accuracy obtained when using the top ten features resulting from implementing the *oneR* algorithm was assessed to evaluate the impact of reducing the number of features. The result was 90.43%, only 0.03% below the best value obtained using all features. Consequently, hereinafter, only the top ten features selected by the *oneR* algorithm were used.

To evaluate the effects of using the three features related to the state's history, the top ten features resulted from the previous test were taken along with the referred three features. The procedure used was the same as the one applied to select the top ten features, with the results summarized in Table 6. Its analysis allows the following conclusions:

- The *prevState* feature becomes the most relevant. In fact, comparing the results obtained using the *CFS* filter (96.27%) with the remaining ones (98.50% and 98.37%), it seems that the model resorts almost exclusively on that feature. Such situation may indicate overfitting, which must be confirmed with a detailed analysis of the models (Section 4.2);
- The *transition* feature seems to be irrelevant to the model, and thus it won't be used in further tests;
- The *nequalStates* feature is highly ranked when using the *Random Forest* filter but it is poorly ranked when using the *CFS* filter and *Chi Squared* filter. Despite this, it was maintained to build the models;
- Albeit the *prevState* feature predominates, the main features of the previous analysis (*pitch* and *dist.mm*) are still well ranked.

4.2. Modelling tests

Following the strategy described in Section 3.7, several models were built, all resorting in DT and following a 10-fold cross validation approach. Moreover, the results were evaluated using the techniques described in Section 3.8. Considering the results of the feature selection procedures, three main study cases were considered, being evaluated and discussed in the following sections, particularly:

1. Using the top ten features obtained after feature selection, without using the state's history;
2. Using the same top ten features plus the *prevState*;
3. Using the same top ten features plus the *prevState* and *nEqualStates.*

Table 6: Feature Selection evaluation with stateful features

Technique/order	**CFS**	**Chi Squared**	**oneR**	**Random Forest**
1	prevState	prevState	prevState	prevState
2	-	meanM	dist.mm	nequalStates
3	-	pitch	pitch	pitch
4	-	varianceD	meanM	meanM
5	-	dist.mm	varianceD	stdDeviationM
6	-	stdDeviationM	stdDeviationM	varianceD
7	-	minD	minM	maxD
8	-	minM	minD	dist.mm
9	-	maxD	maxD	minD
10	-	nEqualStates	acc_z	minM
11	-	acc_z	transition	acc_z
12	-	transition	acc_y	acc_y
13	-	meanD	nequalStates	transition
accuracy	**96.27%**	**98.50%**	**98.50%**	**98.37%**

4.2.1. Case 1: Using the top ten features, without using the state's history

The confusion matrix obtained using the top ten features is illustrated in Table 7. The most critical situations for the posture control mechanism and which number is intended to be minimized, are signalled in bold and underlined. These observations correspond to missclassfied observations related to the *infracting* state, corresponding both to *False Positives* and *False Negatives.* Since both cases have a negative impact, a sum of all these situations was considered when comparing such results. In this case, a total of **366** situations were registered.

Table 11 summarizes the metrics calculated to evaluate the model. This model achieved a global accuracy of 90.53% with an average of a total of 25 splits. Analysing the results, the *micro average F1 score* presents a value very close to the accuracy. This means that the model performs well in overall, which was expected due to the imbalanced dataset and due to a high level of prediction correctness of the *eating* state, the predominant one. Contrary, the *macro average F1 score* presented a lower value, revealing the limitations of the

Table 7: Confusion matrix when modelling with the top-10 features

		Reference					
		E	I	M	R	S	Total
Predicted	E	10023	**43**	355	4	40	10465
	I	**46**	313	**123**	**4**	**30**	516
	M	158	**109**	1102	84	54	1507
	R	3	**0**	57	135	0	195
	S	16	**11**	19	0	120	166
	Total	10246	476	1656	227	244	12849

model to correctly classify individual classes. The same conclusion is given by the R_K coefficient.

As the *eating* states predominates, the model gives a higher importance to the *pitch* angle and to the distance to the ground (*dist.mm)* features. However, for the remaining states, the values for these two features are equivalent or at least very close to each other. Thus, to differentiate those states, the dynamic acceleration related features should be vital. However, the use of the dynamic acceleration measurements also presented some limitations when applied to this use case, particularly due, on the one hand, to the very smooth boundaries between states, and, on the other hand, to the unbalanced dataset. Consequently, strategies of downsampling and upsampling were tested, but without any relevant improvement.

Additionally, as a high number of misclassifications were detected in situations of transition between states and as, theoretically, there are transitions that are more likely to occur than others, the *prevState* feature was added. The idea was to introduce a temporal feature that could help the model to find relationships between transitions of states.

4.2.2. Case 2: Using the same top ten features plus the prevState

As it can be inferred trough the analysis of the confusion matrix represented in Table 8, adding the *prevState* feature allowed the decrease of a high number of

Table 8: Confusion matrix when using the top-10 features plus the *prevState* feature

		Reference					
		E	I	M	R	S	Total
Predicted	E	10119	**23**	111	7	27	10287
	I	**2**	423	**29**	**2**	**2**	458
	M	102	**27**	1479	23	18	1649
	R	2	**1**	23	194	0	220
	S	211	**2**	14	1	197	425
	Total	10436	476	1656	227	244	13039

false positives and false negatives of the *infracting* state. The sum of those cases is 88, with a total average of only 5 splits in the decision tree. Consequently, the evaluation metrics presented notorious improvements as it can been in Table 11.

However, after evaluating an example of a DT obtained during the modelling phrase, a set of unfeasible and unrealistic conditions for practical implementations was detected. A crass example is the existence of infinite loops, as it is exemplified in Figure 3. As it can be observed, the algorithm would stuck in

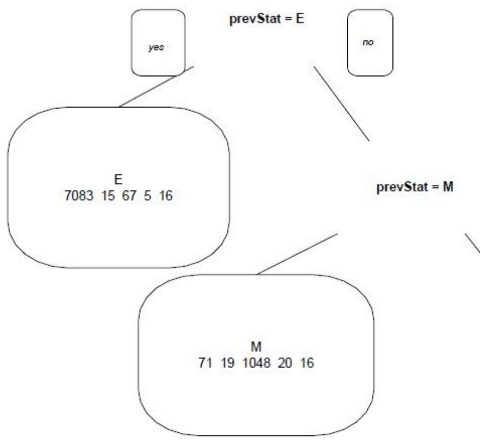

Figure 3: Example of a loop in the Decision Tree

the *eating* state whenever the *prevState* was *eating*. This undesirable behaviour may be explained because of the common sheep behaviour, *i.e.,* they typically stay in the same state for a considerable amount of time. Thus, for minimizing the error, the model assumes that it is better to predict the next state as being the same as the previous one. In brief, besides promising results, this model did not bring any value to the mechanism.

4.2.3. Case 3: Using the top ten features plus the prevState and nEqualStates

The previous test exposed a critical limitation of using the *prevState* feature for modelling the sheep behaviour: the creation of infinite loops within the same state, *i.e.,* in a real implementation scenario, the algorithm would stuck in the initial state defined. Aiming at overcoming this issue, the *nEqualStates* feature was added to the model. The justification for this choice was to provide additional information to the model about the transition between states, particularly, information that could avoid dead end conditions.

The obtained confusion matrix is depicted in Table 9. The number of false positives decreased again when comparing to the first two tests and the evaluated metrics also presented better results (Table 11). However, and again after analysing the DT obtained, a critical limitation was again found. Albeit direct infinite loops were avoided, hidden infinite loops were detected. For example, the conditions for being in the *eating* state are two: the *prevState* is *eating* and the *nEqualStates* is greater or equal to 2. This means that if an animal stays in the *eating* state more than one sample, it will be stuck forever in such state. Besides presenting only the specific case of the *eating* state, similar situations occur to the remaining states.

Downsampling and upsampling techniques were also tested in the present use case, being applied both to the classified behaviour states and to the number of transitions (trying to balance the number of transition and not transitions). Notwithstanding, no relevant changes or enhancements were detected.

Table 9: Confusion matrix when using the top-10 features plus the *prevState* and the *nEqualStates* features

		Reference					
		E	I	M	R	S	Total
Predicted	E	10219	**2**	42	5	7	10275
	I	**2**	436	**17**	**3**	**9**	467
	M	24	**35**	1589	13	34	1695
	R	0	**0**	5	206	0	211
	S	1	**3**	3	0	194	201
	Total	10246	476	1656	227	244	12849

555 *4.2.4. Using the same top ten features plus the nEqualStates*

Albeit the use of the *prevState* feature has revealed itself to be a misstep, at least with the present dataset, the existing improvements from case 2 to case 3 led to the consideration of a fourth use case, joining the top ten features with the *nEqualStates* feature. The obtained confusion matrix is detailed in Table 560 10. Comparing to the ones obtained for case 1, a decrease of 20 missclassified cases regarding the *infracting state* is observed (a decreasing of almost 5%).

The global accuracy obtained (Table 11) was 91.78% with a total average of

Table 10: Confusion matrix obtained when using the top-10 features plus the *nEqualStates* feature

		Reference					
		E	I	M	R	S	Total
Predicted	E	10043	**13**	287	16	29	10388
	I	**36**	330	**130**	**4**	**32**	532
	M	142	**118**	1176	88	58	1582
	R	2	**0**	42	119	0	163
	S	23	**15**	21	0	125	184
	Total	10246	476	1656	227	244	12849

Table 11: Results of the metrics evaluated for the four cases tested during the modelling phrase. Case 4 is next to Case 1 since it is an enhancement of it.

	Case 1	**Case 4**	Case 2	Case 3
Accuracy	0,9100	**0,9178**	0,9660	0,9840
microF	0,9100	**0,9178**	0,9660	0,9840
macroF	0,7031	**0,7086**	0,8947	0,9359
R_K	0,7314	**0,7571**	0,9010	0,9536

25 splits, more than 1% above the global accuracy obtained for case 1. Also, some relevant improvements are observed in the remaining metrics evaluated, including in the *macro average F1 Score* and R_K coefficient.

5. Discussion

All the experiences and tests herein presented were mainly aimed to investigate models that could contribute to the development of an enhanced monitoring mechanism, more efficient and less prone to errors than the existing ones. Additionally, despite the distinctive requirements of the work, namely the existence of a different set of behaviour states and the need of presenting a feasible set of conditions to be implemented on a constrained micro-controller, a comparison with the state of the art works is also pertinent.

After pre-processing the data collected by a customer of iFarmtec, the existence of an unbalanced dataset was noticed. This is the result of the natural behaviour of sheep that naturally feed/eat for long periods of time, being the remaining states mainly transient. Albeit upsampling and downsampling techniques have been evaluated without success, the relevant point is that real implementations need to deal with this issue because it is an inherent and continuous condition of the application scenario.

With the goal of minimizing such concern, additional features regarding the state's history were added. The objective was to give as input, additional information about the refereed natural behaviour of sheep. Although the metrics obtained for these cases were very promising, the truth is that the results were impracticable in terms of real implementations, particularly when resorting to the *prevState* feature. The appearance of infinite loops within the same state revealed that these states do not solve the issue. In fact, the number of transitions between states is not relevant comparing to the total number of samples. Hence, the model presents a lower error by classifying the next state as the previous. This fact precludes the model to seek for other kind of relationships.

Nonetheless, these results may give good insights for future works. For instance, solutions for forcing the models to break the loops shall be investigated as well as other features that could give better insights to the model about the transitions between states. Furthermore, even through in practical use cases, a imbalanced dataset will also be found, bigger and richer datasets would be important to provide a deeper study on this issue.

Taking advantage of the improvements observed from Case 2 to Case 3, a final test using the top ten features plus the feature containing the number of samples on which no changes in the state are registered (*nEqualStates)* was considered. This case not only allowed an improvement of the global accuracy (reaching values similar to the state of the art) but also a reduction of the number of total of false positives and false negatives associated to the *infracting* state. The analysis of the DT obtained did not show any restriction for its implementation in real scenarios, presenting a total of 25 splits.

Comparing the presented results with the ones from previous works that also considered the *infracting* state, the discussed model includes all the states that are relevant to implement the posture control algorithm and that were not considered previously, while maintainig the levels of accuracy. However, future work is required. Besides studying the phenomena observed when using the *prevState* feature on a bigger, richer and more balanced dataset, future work shall include the evaluation of the mechanism developed in a real scenario with different animals.

6. Conclusion

ML is a powerful tool for extracting information that may be hidden in vast and complex datasets. Sheep behaviour monitoring is one application where ML has been increasingly used for predicting behaviour states resorting on the data gathered by tri-axial accelerometers. In the present paper, a specific set of behaviours that are crucial for the SheepIT solution were considered, including the typical *eating*, *moving*, *running* and *standing* states but also a original one, namely the *infracting* state.

Previous works that have already addressed this challenge, present several limitations. Besides considering a small set of states (which entail significant practical operational limitations), they supported their conclusions in a very limited dataset, not exploring all the potentialities of the accelerometer module incorporated. Thus, the present study aimed at providing an improved algorithm for the posture control algorithm, taking advantage of the data gathered on a real application scenario.

The data collection phase resulted in an unbalanced dataset, consequence of the natural sheep behaviour. That led to the consideration of the state's history features (temporal features) in complement to the features based on the accelerometry measurements. In the former group, three different features were considered while in the latter, a set composed by the top-10 ranked features after feature selection were contemplated.

The modelling phase was summed up to the application of DT due to simple implementation on constraint hardware. From the modelling phase, two important conclusions can be taken. Firstly, using temporal features has revealed very promising results, although resulting in unpractical implementations. Even so, those results provided good insights for future research, that shall be focused on understanding how to deal with naturally unbalanced datasets from animal monitoring and more particularly, how can we take advantage of the presented temporal features to improve the posture control algorithm without losing the feasibility of the solution. Secondly, the final feasible model considered accuracies in the same order of the previous works, although it considered more states. For this latter scenario, although being conceptually validated, a field trial version shall be evaluated in future work.

Acknowledgements

This work is funded by FCT/MEC through national funds under the project PTDC/EEI-TEL/30685/2017, and by the Integrated Programme of SR&TD "SOCA" (Ref. CENTRO-01-0145-FEDER-000010). We would like to thank to José Pereira from iFarmtec for his availability to clarify the system's operation.

References

[1] ITU-T, Overview of the internet of things, Series Y: Global Information Infrastructure, Internet Protocol aspects and Next-Generation Network - Next Generation Networks – Frameworks and functional architecture models Y.2060 (2012) 1–32.

[2] I. A. Dave Ross, Craig Michie, Carol-Anne Duthie, Shane Troy, Advances in monitoring of livestock, in: Innovation in Livestock Production: From ideas to Practice, Warsaw, Poland, 31 August - 4 September 2015.

[3] R. M. D. Antunes, Sistema de Monitorizaçao de Comportamento Animal- O Ciclo Reprodutivo, phdthesis, Universidade de Trás dos Montes e ALto Douro (2007).
URL `http://repositorio.utad.pt/handle/10348/721`

[4] P. L. Greenwood, G. J. Bishop-Hurley, L. A. González, A. B. Ingham, Development and application of a livestock phenomics platform to enhance productivity and efficiency at pasture, Animal Production Science 56 (8) (2016) 1299–1311.

[5] N. Watanabe, S. Sakanoue, K. Kawamura, T. Kozakai, Development of an automatic classification system for eating, ruminating and resting behavior

of cattle using an accelerometer, Grassland Science 54 (4) (2008) 231–237. doi:10.1111/j.1744-697x.2008.00126.x.

[6] CowScout, Available online: https://www.gea.com/en/products/activity-detection-cowscout.jsp (Accessed on 2018-01-26).

[7] Cowlar, Available online: https://cowlar.com (Accessed on 2018-01-26).

[8] Agersens, Available online: https://agersens.com (Accessed 2018-1-26).

[9] Digitanimal, Available online: https://digitanimal.com (Accessed on 2018-05-17).

[10] Sheepit project, Available online: http://www.av.it.pt/sheepit (Accessed on 2018-04-15).

[11] L. Nóbrega, P. Gonçalves, P. Pedreiras, J. Pereira, An IoT-Based Solution for Intelligent Farming, Sensors 19 (3). doi:10.3390/s19030603.

[12] L. Nóbrega, A. Tavares, A. Cardoso, P. Gonçalves, Animal monitoring based on iot technologies, in: Proceedings of the IoT Vertical and Topical Summit on Agriculture, IEEE, Tuscany, Italy, 8-9 May 2018, pp. 1–5.

[13] A. Cardoso, J. Pereira, L. Nóbrega, P. Gonçalves, P. Pedreiras, V. Silva, SheepIT: Activity and Location Monitoring, in: Proceedings of the INForum 2018, Coimbra, Portugal, 5-6 September 2018, pp. 1–12.

[14] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, D. Bochtis, Machine learning in agriculture: A review (2018). doi:10.3390/s18082674.

[15] Á. Ortiz-Pelaez, D. U. Pfeiffer, Use of data mining techniques to investigate disease risk classification as a proxy for compromised Biosecurity of cattle herds in Wales, BMC Veterinary Research 4. doi:10.1186/1746-6148-4-24.

[16] L. Yin, T. Hong, C. Liu, Estrus detection in dairy cows from acceleration data using self-learning classification models, Journal of Computers (Finland) 8 (10) (2013) 2590–2597. doi:10.4304/jcp.8.10.2590-2597.

[17] M. R. Borchers, Y. M. Chang, K. L. Proudfoot, B. A. Wadsworth, A. E. Stone, J. M. Bewley, Machine-learning-based calving prediction from activity, lying, and ruminating behaviors in dairy cattle, Journal of Dairy Science 100 (7) (2017) 5664–5674. doi:10.3168/jds.2016-11526.

[18] G. A. Miller, M. A. Mitchell, Z. E. Barker, K. Giebel, E. Codling, J. Amory, C.-A. Duthie, Animal-mounted sensor technology to predit 'time to calving' in beef and dairy cows., in: Proceedings of the BSAS Annual Conference, Edinburgh, United Kingdom, 09-11 October 2019.

[19] M. L. Williams, N. Mac Parthaláin, P. Brewer, W. P. James, M. T. Rose, A novel behavioral model of the pasture-based dairy cow from GPS data using data mining and machine learning techniques, Journal of Dairy Science 99 (3) (2016) 2063–2075. doi:10.3168/jds.2015-10254.

[20] R. N. Handcock, D. L. Swain, G. J. Bishop-Hurley, K. P. Patison, T. Wark, P. Valencia, P. Corke, C. J. O'Neill, Monitoring animal behaviour and environmental interactions using wireless sensor networks, GPS collars and satellite remote sensing, Sensors 9 (5) (2009) 3586–3603. doi:10.3390/s90503586.

[21] Y. Guo, P. Corke, G. Poulton, T. Wark, G. Bishop-Hurley, D. Swain, Animal behaviour understanding using wireless sensor networks, in: Proceedings - Conference on Local Computer Networks, LCN, 2006, pp. 607–614. doi:10.1109/LCN.2006.322023.

[22] F. A. Alvarenga, I. Borges, L. Palkovič, J. Rodina, V. H. Oddy, R. C. Dobos, Using a three-axis accelerometer to identify and classify sheep behaviour at pasture, Applied Animal Behaviour Science 181 (2016) 91–99. doi:10.1016/j.applanim.2016.05.026.

[23] D. Gutierrez-Galan, J. P. Dominguez-Morales, E. Cerezuela-Escudero, A. Rios-Navarro, R. Tapiador-Morales, M. Rivas-Perez, M. Dominguez-Morales, A. Jimenez-Fernandez, A. Linares-Barranco, Embedded neural

network for real-time animal behavior classification, Neurocomputing 272 (2018) 17–26. doi:10.1016/j.neucom.2017.03.090.

[24] M. Moreau, S. Siebert, A. Buerkert, E. Schlecht, Use of a tri-axial accelerometer for automated recording and classification of goats' grazing behaviour, Applied Animal Behaviour Science 119 (3-4) (2009) 158–170. doi:10.1016/j.applanim.2009.04.008.

[25] C. Umstätter, A. Waterhouse, J. P. Holland, An automated sensor-based method of simple behavioural classification of sheep in extensive systems, Computers and Electronics in Agriculture 64 (1) (2008) 19–26. doi:10.1016/j.compag.2008.05.004.

[26] E. Nadimi, R. Jørgensen, V. Blanes-Vidal, S. Christensen, Monitoring and classifying animal behavior using ZigBee-based mobile adhoc Wireless Sensor Networks and Artificial Neural Networks, Computers and Electronics in Agriculture 82 (2012) 44–54. doi:10.1016/j.compag.2011.12.008.

[27] S. Navon, A. Mizrach, A. Hetzroni, E. D. Ungar, Automatic recognition of jaw movements in free-ranging cattle, goats and sheep, using acoustic monitoring, Biosystems Engineering 114 (4) (2013) 474–483. doi:10.1016/j.biosystemseng.2012.08.005.

[28] V. Giovanetti, M. Decandia, G. Molle, M. Acciaro, M. Mameli, A. Cabiddu, R. Cossu, M. G. Serra, C. Manca, S. P. Rassu, C. Dimauro, Automatic classification system for grazing, ruminating and resting behaviour of dairy sheep using a tri-axial accelerometer, Livestock Science 196 (2017) 42–48. doi:10.1016/j.livsci.2016.12.011.

[29] M. Decandia, V. Giovanetti, G. Molle, M. Acciaro, M. Mameli, A. Cabiddu, R. Cossu, M. G. Serra, C. Manca, S. P. Rassu, C. Dimauro, The effect of different time epoch settings on the classification of sheep behaviour using tri-axial accelerometry, Computers and Electronics in Agriculture 154 (2018) 112–119. doi:10.1016/j.compag.2018.09.002.

[30] E. Walton, C. Casey, J. Mitsch, J. A. Vázquez-Diosdado, J. Yan, T. Dottorini, K. A. Ellis, A. Winterlich, J. Kaler, Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour, Royal Society Open Science 5 (2). doi:10.1098/rsos.171442.

[31] N. Mansbridge, J. Mitsch, N. Bollard, K. Ellis, G. G. Miguel-Pacheco, T. Dottorini, J. Kaler, Feature selection and comparison of machine learning algorithms in classification of grazing and rumination behaviour in sheep, Sensors (Switzerland) 18 (10). doi:10.3390/s18103532.

[32] J. Barwick, D. W. Lamb, R. Dobos, M. Welch, M. Trotter, Categorising sheep activity using a tri-axial accelerometer, Computers and Electronics in Agriculture 145 (2018) 289–297. doi:10.1016/j.compag.2018.01.007.

[33] J. Marais, S. Le Roux, R. Wolhuter, T. Niesler, Automatic classification of sheep behaviour using 3-axis accelerometer data, in: Proceedings of Pattern Recognition Association of South Africa and AfLaT International Joint Symposium, Cape Town, RSA, 27-28 November 2014, pp. 1–6.

[34] M. Schwager, D. M. Anderson, Z. Butler, D. Rus, Robust classification of animal tracking data, Computers and Electronics in Agriculture 56 (1) (2007) 46–59. doi:10.1016/j.compag.2007.01.002.

[35] R. Dutta, D. Smith, R. Rawnsley, G. Bishop-Hurley, J. Hills, G. Timms, D. Henry, Dynamic cattle behavioural classification using supervised ensemble classifiers, Computers and Electronics in Agriculture 111 (2015) 18–28. doi:http://dx.doi.org/10.1016/j.compag.2014.12.002.

[36] J. A. Vázquez Diosdado, Z. E. Barker, H. R. Hodges, J. R. Amory, D. P. Croft, N. J. Bell, E. A. Codling, Classification of behaviour in housed dairy cows using an accelerometer-based activity monitoring system, Animal Biotelemetry 3 (1) (2015) 3–15. doi:10.1186/s40317-015-0045-8.

[37] A. L. H. Andriamandroso, F. Lebeau, Y. Beckers, E. Froidmont, I. Dufrasne, B. Heinesch, P. Dumortier, G. Blanchy, Y. Blaise, J. Bindelle,

Development of an open-source algorithm based on inertial measurement units (IMU) of a smartphone to detect cattle grass intake and ruminating behaviors, Computers and Electronics in Agriculture 139 (2017) 126–137. doi:10.1016/j.compag.2017.05.020.

[38] Moomonitor +, Available online: https://www.dairymaster.com/products/moomonitor (Accessed on 2018-1-26).

[39] iFarmtec, Available online: http://ifarmtec.pt/.

[40] ST, LSM303C, Ultra-compact high-performance eCompass module: 3D accelerometer and 3D magnetometer, Available online: https://www.st.com/resource/en/datasheet/lsm303c.pdf (2014).

[41] H. Kopetz, Time-triggered model of computation, in: Proceedingsof the Real-Time Systems Symposium, 1998, pp. 168–177.

[42] Open Camera, Available online: http://opencamera.org.uk/ (Accessed on 2019-10-18).

[43] RStudio - RStudio, Available online: https://rstudio.com/.

[44] M. DASH, H. LIU, Feature selection for classification, Intelligent Data Analysis 1 (1-4) (1997) 131–156. doi:10.1016/s1088-467x(97)00008-5.

[45] CRAN - Package FSelector, Available online: https://cran.r-project.org/web/packages/FSelector/index.html.

[46] Performance Measures for Multi-Class Problems, Available online: https://www.datascienceblog.net/post/machine-learning/performance-measures-multi-class-problems/.

[47] J. Gorodkin, Comparing two K-category assignments by a K-category correlation coefficient, Computational Biology and Chemistry 28 (5-6) (2004) 367–374. doi:10.1016/j.compbiolchem.2004.09.006.

[48] D. Chicco, Ten quick tips for machine learning in computational biology, BioData Mining 10 (1). doi:10.1186/s13040-017-0155-3.