Article

PRIPRO: A Comparison of Classification Algorithms for Managing Receiving Notifications in Smart Environments.

João Antônio Martins [1], Iago Sestrem Ochôa[1,2], Luis Augusto Silva [1], André Filipe Sales Mendes[2], Gabriel Villarrubia González[2], Juan De Paz Santana[2], Valderi Reis Quietinho Leithardt [1,3]

[1] Laboratory of Embedded and Distributed Systems-LEDS, University of Vale do Itajaí - Itajaí-SC, 88302-901, Brazil
[2] Departamento de Informática e Redes de Computadores, Instituto Federal Catarinense (IFC), Brusque 88354-300, Brazil
[3] Expert Systems and Applications Lab, Faculty of Science, University of Salamanca, Plaza de los Caídos s/n, 37008 Salamanca, Spain
[4] Departamento de Informática, Universidade da Beira Interior, 6200-001 Covilhã, Portugal
* Correspondence: joao_martins@edu.univali.br

Abstract: With the evolution of technology over the years, it has become possible to develop intelligent environments based on the concept of the Internet of Things, distributed systems, and machine learning. Such environments are incorporated with various solutions to solve user demands from services. One of these solutions is UBIPRI middleware, whose central concept is to maintain privacy in smart environments and to receive notifications as one of its services. However, this service is freely performed, disregarding the privacy that the environment employs. Another consideration is that based on the researched related works, it was possible to identify that the authors do not use statistical hypothesis tests in their solutions developed in the presented context. This work proposes an architecture for notification management in smart environments, composed by a notification manager named PRIPRO to assign it to UBIPRI and to aim to perform tests and comparisons between classification algorithms to delimit which one is most feasible to implement in the PRINM decision-making mechanism. The experiments showed that the J48 algorithm obtained the best results compared to the other algorithms tested and compared.

Keywords: Smart Environments; Notification Management; Machine Learning;

---

1. Introduction

Technology is increasingly incorporated into people's daily lives, becoming distributed and no longer traditional in various areas of its activities, establishing a new concept, contextualized as IoT (Internet of Things) [1]. Since many of the objects (electronic components, communication sensors, and mobile devices) that surround people's daily lives will be connected, a large amount of information will be generated due to data collection and transmission.

Ordinary everyday places can become intelligent environments when they respond to the presence of people in a versatile manner, meeting their specific needs with the help of IoT objects embedded in the [2] environment. Consequently, people do not notice that they are using a computer system directly but understand that the physical environment interfaces with the interaction with the computer system embedded there. Such environments are part of a distributed system that is a set of software running on one or more computers and coordinating actions by messaging [3].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

One of the technologies that aid IoT devices in intelligent environments and commonly used in distributed systems is MW (*Middleware*), which is a resource manager that offers your applications to share and deploy these resources efficiently. in a network [4]. In addition to resource management, MW offers services similar to those of an operating system, such as application communication facilities, security services, accounting services, and disaster recovery.

In work developed by [5], an MW was proposed for privacy control and management in intelligent environments called UBIPRI. The central concept is to enable devices to meet the needs of users or environments as a whole. Adapting to different environments and their infrastructure, adapting device limitations, and environmental privacy. Another goal of UBIPRI is to classify the type of user access in smart environments based on variables such as profile type, user frequency, environment type, and day of the week. Thus, providing users with the availability of services present in the environment accessed and specific actions of these services. One of the MW layer assignments is presented in Figure 1, and it is divided into modules, each module being tasked with a specific task.

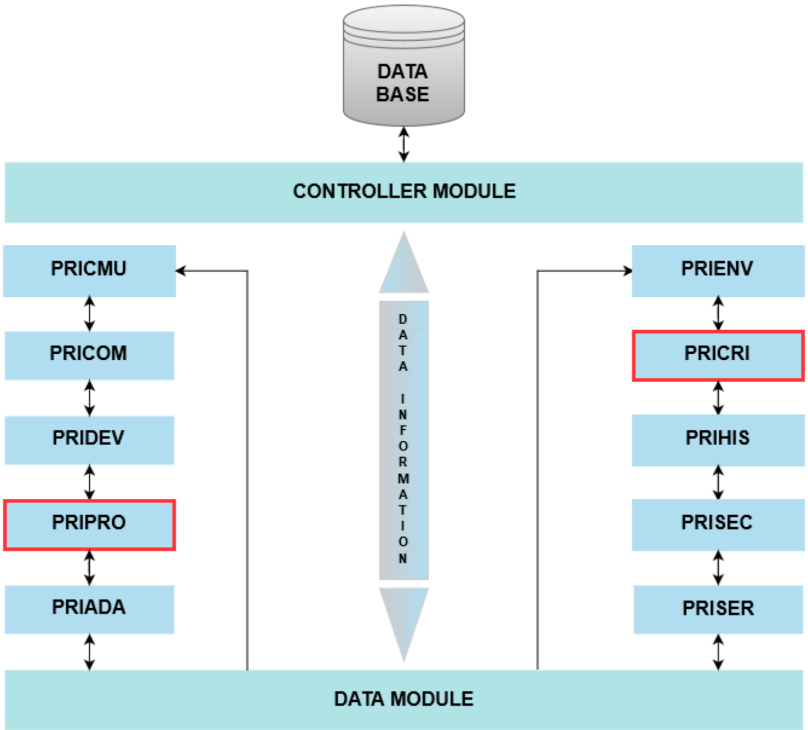

Figure 1. Privacy manager model. Ubipri (2015) [6]

According to Figure 1, the PRICMU, PRICOM, and PRIDEV modules are responsible for the management, privacy control, communication, and devices, respectively. The PRIADA module is responsible for adaptation management and control. PRIENV is the environment-related attributes registration module. PRIHIS is the module for storing and processing information related to user history. PRISEC is the module related to user safety and the environment. PRISER is the environment service management module. Connecting to all modules, the Data Module processes variables and parameters received from other modules. Controller Module is the module that receives access requests and performs the control of the database directly in the tables.

The modules described above are not relevant to the development of the work; in contrast, the modules PRIPRO and PRICRI are of high relevance. The first, respectively, is responsible for performing control transactions that are related to user profile management, aiming to distribute and direct synthesized information to the next modules. This information is adapted appropriately according to the individual privacy of the user and their profile adhered to by UBIPRI. The PRICRI module has rules, criteria, and environment definitions such as access, use, sharing, location, and other variables that can be added, changed, or modified — pointing out that each environment has unique characteristics, such that their definitions are treated individually by the other modules that have specific controls.

One of the services that UBIPRI provides is the receipt of notifications for IoT devices in smart environments that are performed freely without the intervention of MW, i.e., disregarding the privacy that the environment employs and which users should meet. Therefore, it is noted that the related works in the context of notification management in smart environments do not use statistical hypothesis tests as a complement to statistical evaluation.

From the gap found in UBIPRI that results in the privacy issue of notifications in smart environments, it is necessary to intelligently manage their receipt, as it is an MW of privacy control and management. Therefore, to ensure the privacy that environments employ, we propose the architecture of a notification manager. With its core component, an intelligent Decision Maker (DM) engine implemented with a Machine Learning (ML) algorithm that belongs to the supervised sorting task category for managing receiving notifications from users using UBIPRI.

This paper presents a comparison of classification algorithms to delimit, which is the most viable to be implemented in the PRINM DM mechanism that will be developed in UBIPRI, as well as the notification management architecture in the context of smart environments. Therefore, it aims to report the activities of delimitation and use of classification algorithms and statistical hypothesis testing, generation of artificial data sets, tests, and comparisons of classification algorithms and application scenario testing.

This article is structured in six sections: Section 2 presents the basis for developing tests and comparisons; Section 3 refers to the proposed architecture and application scenario; Section 4 describes the methodology of the tests and comparisons made; Section 5 presents work related to managing notifications in smart environments; Finally, Section 6 presents the conclusions and contributions obtained.

2. Background

This section presents the basis for developing the tests and comparisons performed in Section 4. The concept of ML is introduced, focusing on the classification task presenting which learning categories and their respective classification algorithms were listed and selected. The study and delimitation of a statistical hypothesis test. Finally, the process performed to generate artificial data sets.

2.1. Machine Learning Concept

The concept of ML can be defined as programming that makes computers make decisions using data from examples of past experiences. Based on a model with parameters to be optimized from learning training data. The model can be predictive for making future predictions or descriptive for data knowledge [7]. Therefore, there are two aspects of ML for model generation, supervised and unsupervised. Within each of them, there are also different types of tasks, such as classification, regression, grouping, and association, which consequently have different characteristics and algorithms to be used. Figure 2 presents an overview of the ML concept.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

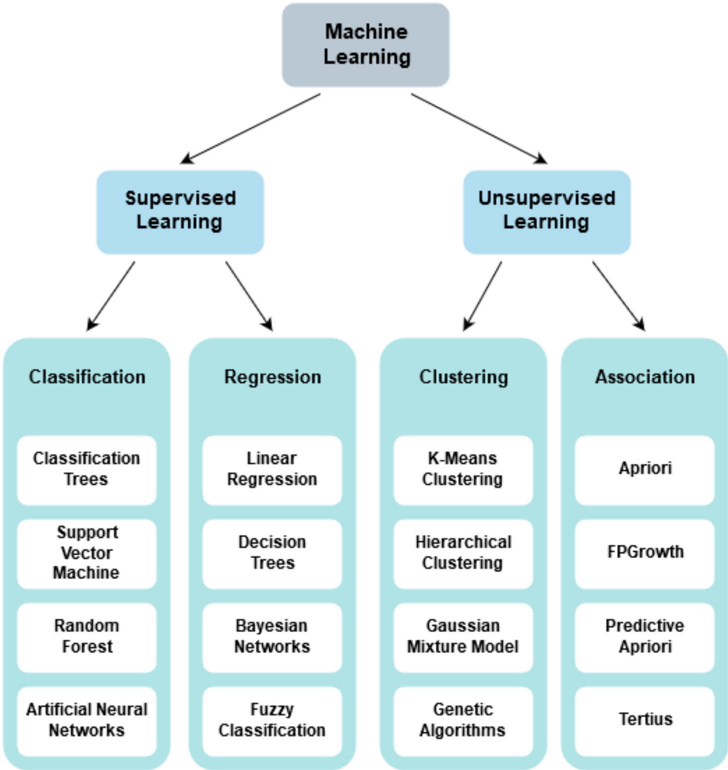

Figure 2. Machine learning structure.

From Figure 2, supervised learning uses predictive data\attributes that define a data record and a data\attribute class that specifies which category this data record belongs to in a data set. Dice. Attributes \ predictor data have the characteristics of something from the context that is trying to predict and attributes \ data classes have the category, thus predicting what a data record is [8]. Unsupervised learning does not need to determine a \ attribute given a class in the dataset, and they are usually techniques and algorithms for exploratory data analysis. It has the \ input data attributes that contain the characteristics needed to be extracted to identify exploratory context patterns [7].

Among the tasks presented, the regression investigates and models the relationship of attributes with continuous output results. Grouping aims at separating data into groups that contain similar attributes, thus discovering hidden patterns and information. Association has the purpose of identifying association rules between attributes/data that may or may not be related. The classification task is addressed and described with greater emphasis on the following subsection since it was selected for use in the development of the work.

2.1.1. Classification Task

Classification task supervised learning algorithms generally consist of recognizing models that describe and distinguish classes for the purpose of using the model to predict the class of data that has not yet been classified. However, the diversity of algorithms that exist within this task is large, so it is necessary to study to find out which ones are potentially better when applied to certain types of problems cite aggarwal2014data.

Figure 3 presents the learning categories on which the classification algorithms are based. Each learning category directly affects the computational behavior of the algorithm, delimiting how learning is performed and the generated predictor model.

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

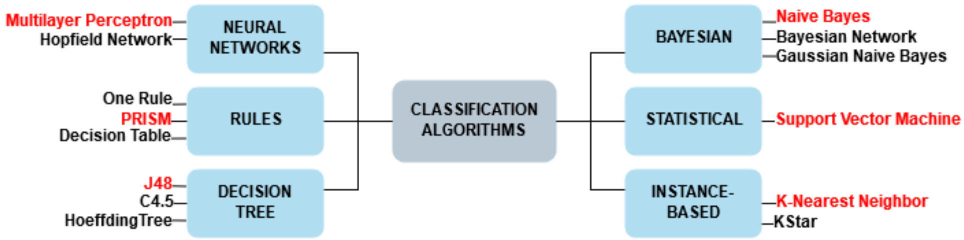

Figure 3. Classification task learning categories.

There are several types of algorithms in each learning category, so the development of the work was selected as an algorithm from each category. This was defined because, in the related literature searched, it was not possible to identify classification algorithms that act correctly in the context of notification management in smart environments. Another resolution for this definition is that classification algorithms act heterogeneously depending on the problem in which it is applied. Thus, the delimited algorithms were: Bayesian learning NB, decision tree learning J48, instance-based learning KNN, neural network learning Multilayer Perceptron (MLP), rule learning PRISM, and statistical learning SVM. The following subsections describe the concept of each learning category and its respective algorithms.

2.1.2. Bayesian Learning

In Bayesian learning, classification algorithms perform their classification by statistical inference, which is the process that makes intact probabilistic statements from incomplete information. From the Bayesian learning perspective, the statistical inference about any amount of interest is described as the modification that occurs in the uncertainties of new evidence. Bayes' theorem allows quantifying this modification [9].

The NB algorithm is considered as a simple classifier, due to its simplicity, it has broad applicability as in real-time forecasts, news classification, spam filtering, recommendation system, among others. A peculiarity of classification and being called *naive* (naive), is that the algorithm disregards the correlation between attributes of a data, i.e., it treats each attribute as if it were independent. Because it is a simple algorithm, NB has no adjustable parameters [10].

2.1.3. Decision Tree Learning

Algorithms that are based on decision tree learning generate a predictor model in the shape of a tree composed of nodes, arcs, and leaf nodes to perform their classification [11]. For a model generation, the training dataset is recursively partitioned into subsets, and the stopping point is when a subset obtains only the class attribute. In the course of creation, we analyze and compare the distribution of attributes to identify where each node will position itself in the [12] tree.

The J48 algorithm is considered a fast classifier and provides good sorting accuracy compared to other sorting task algorithms. Derived from its predecessor algorithms ID3, C4.5, and C5.0, J48 builds its tree based on the strategy of division and conquest, by calculating entropy and information gain. A peculiarity of classification is that the algorithm considers only the most relevant attributes, i.e., it discards specific attributes that are not relevant to the generation of the predictor model [13]. The main adjustable parameter is the use of pruning, which removes the dirt, thus providing a compact size tree [14].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

2.1.4. Instance-Based Learning

In instance-based learning, classification algorithms classify them from a single input and sequence of instances. Each instance is represented by a group of attributes, which are predictor attributes and class attributes. Thus, forming a dataset in a dimensional space of n instances. During classification, similarity calculation is used to calculate the proximity value of a new unrated instance against the other instances of the dataset [15].

The KNN algorithm is the best known and commonly used by the scientific community, among the many instance-based learning. It is categorized as *lazy*, as it does not generate a predictor model. Instead, it uses the similarity calculation with all data in the set to classify the new data entered. Therefore, their classification consists of storing training examples, which consequently postpone the processing of training data until new data needs to be classified [16]. The main adjustable parameters are the K variable, which determines the number of nearest neighbors to be discovered and the similarity calculation to be used.

2.1.5. Artificial Neural Networks Learning

Learning by artificial neural networks is based on the neuronal networks of the human brain, thus creating artificial neurons. By combining these neurons, an artificial neural network is formed with its architecture consisting of a single layer or multiple layers. From this, the operation of algorithms that are based on artificial neural networks is constituted by signals that are presented at the neuron inputs and that are multiplied by the weights. After this multiplication, they are summed by a sum that produces the activity level, which if exceeded will result in a given output [17].

The MLP algorithm consists of a simple system of artificial neurons connected by weights and output signals, which are a function of the sum of inputs for the modified neuron from a linear activation function. The network is divided into three layers: input, hidden and output. The input layer receives the value vector for network initialization, the hidden layer performs training, and the output layer receives the output vector [18]. The main adjustable parameters are the maximum amount of iterations, learning rate, momentum and the amount of neurons in the hidden layer.

2.1.6. Rules Learning

Algorithms that use rule learning perform their classification from *IF* - *THEN* statements, which consists of a condition or prediction. The forecast result is presented by a single rule or a combination of several rules. The rules created to follow a structure that *IF* a condition is met; *THEN* makes a delimited prediction. The predictor models generated from this learning category are the most interpretable, due to their instructional structure that resembles natural language and human thinking [19].

The PRISM algorithm is one of the pioneers of its learning category because it was from it that others were implemented. It has in its computational behavior of classification the induction of rules from a data set. This induction is represented by a fixed set of individual rules for each of the dataset classes. To do so, it has some limitations, such as: not generating value enumeration attributes, lacking the robustness of missing values, and performing no pruning, so it has no adjustable parameter [20].

2.1.7. Statistical Learning

Statistical learning is based on TAE (Statistical Learning Theory). Three steps are needed to generate the algorithm prediction model based on this learning, namely: (i) a x random vector generator, extracted independently of a fixed but unknown $P(x)$ distribution; (ii) A supervisor who returns an output vector y for every input vector x, according to a conditional distribution function $P(y|x)$, also fixed but unknown; (iii) a learning machine capable of implementing a $f(x\alpha, \alpha \in \wedge)$ [21]

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

function set. From TAE, the inductive principle of ERM (Structural Risk Minimization) was conceived, which aims to minimize the error of the training set simultaneously with the error of the test set of classification algorithms. The principle also develops theoretical limits to the generalizability of the predictor algorithms, thus formalizing a larger generalization that implies a greater number of hits in the [22] testing phase.

The SVM algorithm is one of the most efficient classifiers and is used in academia because it can classify from mathematical terms. Therefore, there is a need for a function that describes the factors that must be controlled and that guarantee the good performance of the classification. The SVM predictor model generation is based on support vectors, which are used to learn and define the best separation line in the created hyperplane. The algorithm learns the straight line considering the maximum margin defined by it, thus providing the classification between different classes [23]. The main adjustable parameters are kernel and cost.

2.2. Statistical Hypothesis Tests

The use of statistical hypothesis testing in comparisons of classification algorithms implies an analysis complement between them, indicating whether one algorithm is better than another in a specific task and to determine the probability of incorrectly detecting a statistical difference when there are no difference [24]. One of the goals of these tests is to verify the truth of the null hypothesis, which is the statement that there is no distribution difference between samples (data sets). Thus, the hypothesis verified is H0 (valid and not rejected) or H1 (not valid and rejected) [25].

There are different types of statistical hypothesis testing, namely:

1. Normality testing that is used to evaluate the assumption of a sample taken from a distributed population [26];
2. Correlation test that analyzes sample datasets to identify if two variables are related to each other [27];
3. Association test that reports on the relationship of statistical association between variables [28];
4. Variance test comparing the means of different populations [29];
5. Central tendency test that uses central tendency measures (arithmetic mean, median) to test a probability distribution [30].

From the relationship of the described test types, the central tendency test is the most suitable for the development of the work, because it uses the classification accuracy metric as a measure of central tendency. Table 12 presents the central trend tests.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

Table 1. Types of central tendency tests.

Name	Categorization	Variable	Group	Pairing
Z-test	Parametric	Quantitative	Individual	-
T-test	Parametric	Quantitative	Individual	-
Wilcoxon for 1 sample	No parametric	Quantitative, ordinal qualitative	Individual	-
T-test for 2 samples	Parametric	Quantitative, nominal	Pairs	No paired
T-test for 2 samples with different variances	Parametric	Quantitative, nominal	Pairs	No paired
T-test paired	Parametric	Quantitative, nominal	Pairs	Paired
ANOVA	Parametric	Quantitative, nominal	Multiple	No paired
Welch's ANOVA	Parametric	Quantitative, nominal	Multiple	No paired
ANOVA for repeated measures	Parametric	Quantitative, nominal	Multiple	Paired
Mann - Whitney	No parametric	Quantitative, ordinal qualitative, nominal	Pairs	No paired
Wilcoxon Paired	No parametric	Quantitative, ordinal qualitative, nominal	Pairs	Paired
Kruskal - Wallis	No parametric	Quantitative, ordinal qualitative, nominal	Multiple	No paired
Friedman	No parametric	Quantitative, ordinal qualitative, nominal	Multiple	Paired
Test for 1 proportion	Parametric	Nominal	Individual	-
Test for 2 proportion	Parametric	Nominal	Pairs	No paired

There are different characteristics among the types of tests presented in Table 12, as follows:

1. Categorization indicates whether the test is parametric or nonparametric. Parametric tests evaluate the null hypothesis from specific data or parameters (mean, standard deviation, etc.). Nonparametric tests evaluate the null hypothesis from distribution types and group relationships [31];
2. Variable indicates the types of variables the test supports;
3. Group, which matches whether the group comparison is individual, paired, or multiple. In this context the classification algorithms are the groups;
4. Pairing, which corresponds to whether it is paired or unpaired. Paired tests match that the data used for predictor model training are also used to test the predictor model, whereas unpaired tests use one data set for training and another for testing [32].

Based on the characteristics of the statistical hypothesis tests, those that fit the tests and comparisons performed in Section 4 were listed, namely: nonparametric, quantitative, multiple, and paired. The nonparametric characteristic has been selected, as it is necessary to identify whether there is really a statistical difference in classification performance between classification algorithms. The classification accuracy metric coincides with the quantitative variable characteristic. Therefore, it is necessary to use multiple comparison tests because the comparison uses six algorithms. Finally, paired tests are best suited as a single artificial data set is used for training and testing. Therefore,

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

Friedman test with these characteristics was listed, to be applied for the comparison of classification algorithms on the subsection 4.4 classification precision metric.

By testing the null hypothesis, it is possible to find out if data sets are different from each other, but it is not possible to identify which ones are. Therefore, to solve this impasse, the Friedman test is used, which performs multiple comparisons between equal-sized data sets analyzing the variance and randomization between them. The comparison is made from a ranking presented in Figure 4. To implement, it is necessary to transform the raw data into data that can be sorted [33].

	Algorithms			
Data Set	1	2	...	k
1	x11	x12	...	x1k
2	x21	x22	...	x2k
⋮	⋮	⋮	⋮	⋮
b	xb1	xb2	...	xbk

Figure 4. Friedman test ranking.

In the context of classification algorithms and datasets, x_{bk} represents the placement that the algorithm obtained relative to the ranking dataset. This matches that each ranking row corresponds to the random seed value with which the set was shuffled, and each column corresponds to the algorithm that was applied. Thus, placing x_{bk} corresponds to the value of the classification accuracy metric acquired from the predictor model generated with a given random seed value. Thus, the algorithm with the highest metric value gets the first ranking position, the second-largest will be ranked second, and so on [34]. Equation 1 presents the mathematical calculation of the ranking.

$$|R_i - R_j| \geq Z\left(\frac{\alpha}{k\,(k-1)}\right)\sqrt{\frac{N \times k\,(k+1)}{6}} \tag{1}$$

Where:

- R_i and R_j is the sum of the positions of the algorithms i e j in the ranking;
- $|R_i - R_j|$ is the difference between the sum of the algorithms;
- $Z\left(\frac{\alpha}{k(k-1)}\right)\sqrt{\frac{N \times k(k+1)}{6}}$ is the critical difference.

From the calculation of Equation 1, the value of the critical difference is the most important, because it indicates whether there is a statistical difference between the summation values of two algorithms in the ranking. This difference is discovered by subtracting these values. Thus, if the result of the subtraction obtained is greater than the critical distance, then it corresponds that the two algorithms are statistically different and that one of them is better in the face of the task adhered to them, that is, in the data set in which they were applied [35]. Therefore, with the Friedman test, it is possible to identify if there is a statistical difference between classification algorithms in the face of a given data set when there is such a difference.

2.3. Artificial Data Sets

To perform the tests and comparisons, three artificial data sets were generated from a script executed in the NetBeans IDE, consisting of predictor attribute values and classifier attributes arranged in the ARFF file format. All three sets use the same predictor attributes and different classifier attributes, so for each set, a different classification objective is defined, described below:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 502; doi:10.3390/app10020502

- Target: classify which user the notification should be notified to;
- Period: classify what time of day the notification should be notified;
- Setting: sorts which device configuration notification should be notified;

The number of data instances is precisely the same for each set containing 4320 data. The predictor attributes and their values are shown in Figure 5, and the classifier attributes and their values are shown in Figure 6.

```
@attribute user { Member1, Member2, Member3 }
@attribute profile { Blocked, Guest, Basic, Advanced, Administrator }
@attribute environment { Public, Private, Restrict }
@attribute activity { Relevance1, Relevance2, Relevance3 }
@attribute status { On, Off }
@attribute inPeriod { InMorning, InAfternoon, InNight, InDawn }
@attribute inTarget { InMember1, InMember2, InMember3, InAll }
```

Figure 5. Predictor Attributes.

```
@attribute outTarget { OutMember1, OutMember2, OutMember3, OutAll, OutTargetNone }
@attribute outPeriod { OutMorning, OutAfternoon, OutNight, OutDawn, OutPeriodNone }
@attribute outSetting { OutSilent, OutVibrate, OutCurrent, OutSettingNone }
```

Figure 6. Classifier Attributes.

The description of the predictive attributes is as follows: (i) *user* identifies which user is in the smart environment; (ii) *profile* determines which type of user profile. This attribute is related to the PRIPRO module; (iii) *environment* determines which type the environment has. This attribute is related to the PRICRI module; (iv) *activity* indicates the relevance of the activity the user is performing in the environment. (v) *status* indicates the status of the device that should be notified; (vi) *inPeriod* indicates the period that the notification was received in the smart environment; (vii) *inTarget* indicates to which user the notification should be notified.

To do so, the classifying attributes are: (i) *outTarget* sorts which environment user the notification should be notified to; (ii) *outPeriod* classifies at which time of day notification should be notified to the user; (iii) *outSetting* classifies what type of device configuration notification should be notified;

Predictor and classifier attributes are assigned different types of values and may contain one or more of them that are related to the context of the work. Thus, the *user* attribute is assigned the values *Member1*, *Member2*, *Member3*, stating that the environment has three members. The profile types are determined with the values*Blocked*, *Guest*, *Basic*, *Advanced*, *Administrator* of the attribute *profile*. There are three distinct types of environment indicated by the *Public*, *Private*, *Restrict* values of the *environment* attribute. The values attribute the relevance of activities performed by users *Relevance1*, *Relevance2*, *Relevance3* of the attribute *activity*, being respectively the first with less, the second with average and the third with high relevance. The *On, Off* values of the *status* attribute indicate respectively whether the device is on or off. The period in which notification was received in the environment is assigned by the *InMorning*, *InAfternoon*, *InNight*, *InDawn* valuesof the *inPeriod* attribute. Finally, the *InMember1*, *InMember2*, *InMember3*, *InAll* values of the *inTarget* attribute determine which user the notification should be notified to, either for specific users or for everyone.

Starting with the values of the classifier attributes, the *outTarget* attribute is assigned the values *OutMember1*, *OutMember2*, *OutMember3*, *OutAll*, *OutNone* indicating to which user the notification should be notified, either to specific users or to all. ; the *outPeriod* attribute that is assigned the values *OutMorning*, *OutAfternoon*, *OutNight*, *OutDawn*, *OutPeriodNone*, indicating the period in which notification should be notified to the user; Finally, the setting that the notification should be notified of is delimited by the *OutSilent*, *OutVibrate*, *OutCurrent*, *OutSettingNone* values of the *outSetting* attribute. Values ending with *None* match the notification will not be notified.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 502; doi:10.3390/app10020502

3. Notification Management Architecture

We divided the architecture into three layers, the smart environment layer, UBIPRI, and PRINM. In the smart environment layer, the sensors have the purpose of collecting information about the environmental context in which it operates, and the users present in it. The UBIPRI layer receives the information and delivers it to the PRICRI and PRIPRO modules, which send the environment, and user attributes to PRINM. At the PRINM layer, online services notifications collected by the *receiver* are received, sending attributes and notifications to the DM engine. Finally, the DM engine classifies for which user, period, and device configuration notifications should be notified from the acquired attributes. Figure 7 presents the architecture overview.

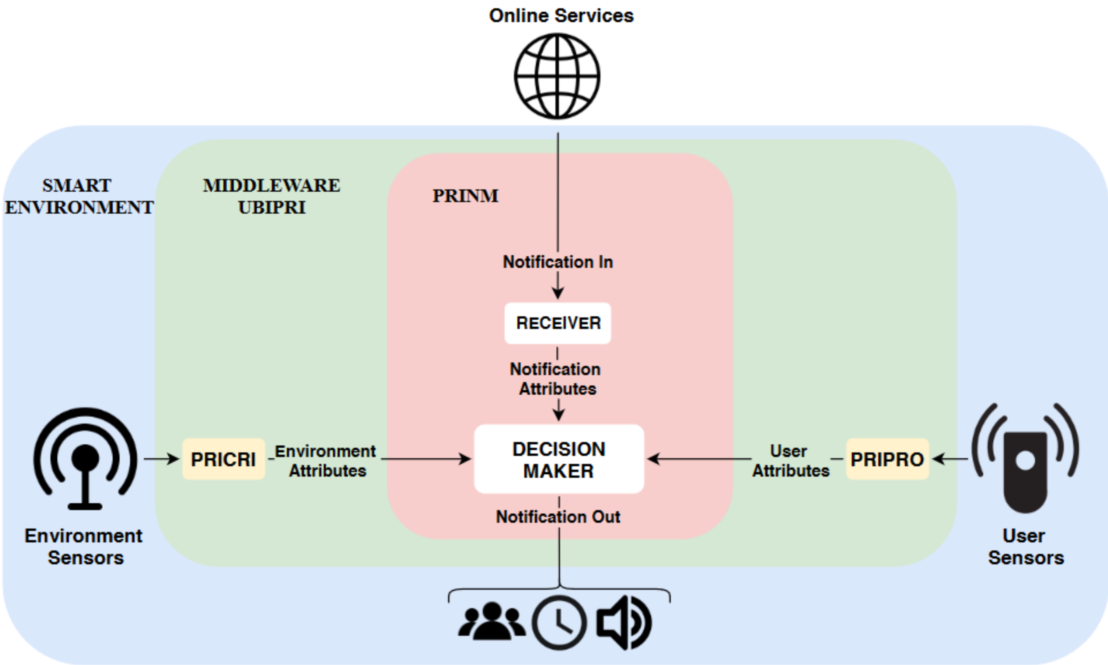

Figure 7. Notification Management Architecture.

Considering the architecture presented in Figure 7 together, the theoretical framework of smart environments and the UBIPRI discussed in Section 1 and the basis for the classification algorithms described in Section 2. PRINM is an implementation to be developed in UBIPRI to maintain the privacy of environments in the context of receiving notifications, utilizing an intelligent DM engine assigned a classification algorithm that receives attributes regarding the environment, users, and notifications. The manager ensures the delivery of notifications according to the privacy managed by UBIPRI in the smart environment in which it operates.

For a better understanding of the proposed architecture, an application scenario was created based on the generated artificial data sets and contextualization presented. The scenario is a car dealership that uses the services of UBIPRI, composed of four different areas, designated showroom, sales, kitchen, and office. Each area has a specific environment type. The scenario also has three users, denominated customer, employee, and owner, who have their IoT devices attached to UBIPRI. As long as users are in the dealership while receiving notifications, PRINM's decision making will notify them. Figure 8 shows the view of the dealership building and each area with its environment type.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 502; doi:10.3390/app10020502

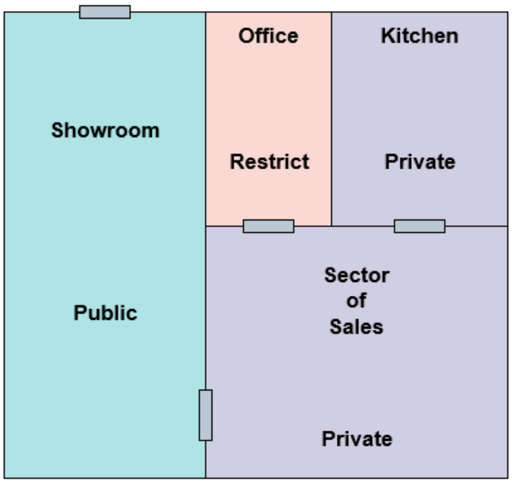

Figure 8. Car dealership building.

In order not to increase the scope of PRINM's performance in the application scenario, we defined that it would happen in just one day with pre-established actions for each user. Thus, for a better understanding and contextualization of the scenario, we individually described the actions of each user within the dealership:

- Customer: Arrives at the dealership in the middle of the morning to search for cars to buy in the showroom. It is serviced by the employee, who then directs him to the sales department to negotiate with the owner. It leaves late in the morning and returns midway through the afternoon. It is attended by the employee in the showroom, who then directs him to the sales department to continue negotiating with the owner. The deal is closed at the owner's office. It leaves late in the afternoon.
- Employee: Arrives at the dealership early in the morning to open it and perform its tasks in the showroom. Take a break in the kitchen. Serves the customer and forwards it for negotiation with the owner in the sales sector. Take a break for lunch in the kitchen. Opens the dealership in the afternoon and performs tasks in the showroom. It covers the homeowner on sales tasks and leaves late in the afternoon.
- Owner: Arrives at the dealership already opened by the employee early in the morning to and perform their tasks in the sales department. He/she performs tasks in his office and soon after goes to the sales department to attend the customer referred by the employee. Take a break for lunch in the kitchen. In the early afternoon, he/she performs tasks in the sales department. Meets the customer again, they close the deal in their private office. Take a break in the early evening in the kitchen. He/she does some chores in his/her office and leaves in the middle of the night.

Considering each user's actions in the developed application scenario, we developed tests in Subsection 4.6 to evaluate the architecture of PRINM behavior with the most viable classification algorithm delimited in its DM engine.

4. Tests and Comparisons of Classification Algorithms

This section presents the tests and comparisons performed with the delimited classification algorithms and the artificial data sets generated in Section 2. The tools used were the WEKA GUI, NetBeans, Excel, and RStudio. All tests were performed on a notebook with the following configurations:

- Intel Core i5-5200U
- CPU: 2.20 GHz

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

- RAM: 6,00 GB

Therefore, we present the test that identifies whether artificial data sets are suitable for use, the test of adjustable parameters, and CPU time for training and classification of the classification algorithms. The comparison test of classification algorithms on the classification accuracy metric. Friedman's test as a complement to statistical analysis. Finally, the application scenario test presenting the applicability of the architecture over PRINM.

4.1. Artificial Datasets Test

After the generation of the *Target*, *Period* and *Setting* artificial data sets presented in Subsection 2.3, it was necessary to perform tests to identify if they are suitable for application in other tests and comparisons. of this section. For this, we used the rule learning algorithm *ZeroR*, which aims to generate the baseline of the classification precision metric. This matches that if the metric of the algorithms intended to be used in the dataset is smaller than that of *ZeroR*, then it is not indicated or appropriate to use these algorithms. Another objective of the algorithm is to predict the majority class of the dataset, that is, it classifies unclassified data with the class that has the most instances in the training data set [36]. Table 2 presents the tests performed to generate the baseline with the three artificial data sets, using the cross-validation model *value 10*.

Table 2. ZeroR baseline generator.

Artificial Data Set	Classification Accuracy	Majority Class
Target	80%	OutTargetNone
Period	80%	OutPeriodNone
Setting	75%	OutSettingNone

Table 2 shows that the baseline in all sets was above 70%, which, according to [37], is considered the appropriate mean for testing in WEKA of the precision metric of classification. The majority class in each dataset references notifications that will not be notified to users in the smart environment, indicating that these values have the most in their datasets. Defining the baseline in each data set, the algorithms delimited in Subsection 2.1 were applied to identify whether they reach the percent of the classification accuracy metric above the baseline of the *ZeroR* algorithm in each set of artificial data. Table 3 presents the tests performed by applying the algorithms to the artificial data sets *Target*, *Period* and *Setting*, and using the validation model *cross validation* with the value 10.

Table 3. Baseline test.

Algorithms	Target	Period	Setting
NB	100%	80%	84,97%
SVM	100%	88,40%	86,59%
KNN	100%	96,99%	99,90%
MLP	100%	100%	100%
PRISM	100%	97,47%	98,81%
J48	100%	97,56%	99,60%

Presenting Table 3, we noticed that the percentage of the classification accuracy metric in all algorithms was above the baseline generated by the previous Table 2 test. Therefore, with the tests performed, we identified that the generated artificial data sets are suitable for use in managing receiving notifications in smart environments. We have also identified that all previously delimited algorithms are fit for use in the other tests and comparisons in this section.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 502; doi:10.3390/app10020502

4.2. Adjustable Parameters Test

We also developed adjustable parameter tests between algorithms to identify which parameters are the best. Thus, for the J48 algorithm, we tested the parameter that determines the use of pruning or not in the decision tree. In the KNN algorithm, we evaluated the K parameters and similarity calculation. The kernel and cost parameters were tested on the SVM algorithm. Finally, for the MLP algorithm, we tested the parameters of the maximum amount of iterations, learning rate, momentum, and the number of neurons in the hidden layer. The PRISM and NB algorithms do not have adjustable parameters; because of this, no tests were performed with them. The tests were performed with the three artificial data sets *Target*, *Period* and *Setting* by verifying the classification accuracy metric and using the validation *cross validation* model with the value 10. Table 4 presents the adjustable parameter test results for each algorithm.

Table 4. Test adjustable parameters.

Algorithms	Parameters
J48	Pruning: use
KNN	Distance calculation: Euclidean K: 3
SVM	Kernel: Linear Cost: 10
MLP	Iteration: 500 Learning rate: 0.3 Momentum: 0.2 Hidden layer neurons: attribute + class

In Table 4, we identify the best adjustable parameters for each classification algorithm. Thus, these parameters were used in the tests and comparisons presented in the following subsections.

4.3. CPU Time Test

After defining the best adjustable parameters in each algorithm, we developed tests of the CPU time metric for predictor model training and the CPU time for classification of new data instances. The CPU time metric was listed as relevant to the scope of the work because, in the context of IoT, there is a great need for information to be transmitted in real-time to both users and IoT devices. Therefore, the tests performed were conducted on each artificial data set and using predictive model validation *cross validation* with the value 10. Tables 5 and 6 show the test results respectively of training and classification time.

Table 5. CPU time test for training.

Artificial Data Set	NB	PRISM	J48	KNN	SVM	MLP
Target	0,47 ms	12,03 ms	2,19 ms	0,47 ms	43,44 ms	15815,31 ms
Period	0,47 ms	124,69 ms	5,63 ms	0,47 ms	417,66 ms	15969,53 ms
Setting	0,94 ms	48,91 ms	3,75 ms	0,16 ms	472,66 ms	14249,53 ms
Average	0,62 ms	61,87 ms	3,85 ms	0,36 ms	311,25 ms	62011,45 ms

Table 6. CPU time test for classification.

Artificial Data Set	NB	PRISM	J48	KNN	SVM	MLP
Target	1,25 ms	0,16 ms	0,63 ms	116,56 ms	6,25 ms	2,19 ms
Period	1,56 ms	1,87 ms	0,31 ms	123,59 ms	47,03 ms	3,13 ms
Setting	1,72 ms	0,47 ms	0,31 ms	123,28 ms	49,06 ms	1,25 ms
Average	1,51 ms	0,83 ms	0,41 ms	121,14 ms	34,11 ms	2,19 ms

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

With the results presented in Table 5, the NB, KKN, and J48 algorithms obtained the shortest times, respectively, following the PRISM and SVM algorithms, and as expected, the MLP algorithm obtained the longest training time. At the end of the test, it was identified that the KNN algorithm is better, and the MLP algorithm is the worst in the training time of a predictor model. Therefore, for Table 6, the algorithms J48, PRISM, and NB obtained the shortest times, followed by the algorithms, respectively, MLP, SVM, and KNN. At the end of the test, it was identified that the J48 algorithm is the best, and the KNN algorithm is the worst in the classification time of a new data instance.

4.4. Classification Precision Metric Test

Regarding the comparison of the classification algorithms on the classification precision metric, the six classification algorithms were applied to the three artificial data sets. For each algorithm, we used the parameters delimited by Table 4 and the test with predictor model validation *cross validation* equal to the value 10. Therefore, the algorithms were executed 30 times in each artificial data set. For each run, we used the random seed generator with different values following an increasing pattern, starting from the value 1. This was done to obtain greater randomness in the metric results, and for the Subsection Friedman test 4.5 was performed. Therefore, each result of the classification accuracy metric for each seed was placed in an Excel spreadsheet to calculate the average of the metric between the algorithms.

Unlike the other tests performed, the classification algorithms were executed in the NetBeans programming IDE using the WEKA package that contains the main features of the tool. This was necessary due to a large number of iterations performed in each algorithm, as it would take a long time if it were performed in GUI technology. Table 7 presents the average results of each randomization seed from the classification accuracy metric.

Table 7. Comparison of classification accuracy metric.

Artificial Data Set	NB	PRISM	J48	KNN	SVM	MLP
Target	100%	100%	100%	99,99%	100%	100%
Period	80%	96,87%	97,49%	97,22%	87,53%	99,96%
Setting	85,26%	98,86%	99,54%	99,89%	86,89%	100%

From Table 7, we observed that all algorithms obtained the metric value above 70 % in all artificial data sets. Analyzing the comparison of the algorithms separately in each set, in Target, almost all the algorithms obtained the percentage with the maximum value, and only the KNN algorithm obtained a lower value. In the *Period* set, the MLP algorithm obtained the best result, the PRISM, J48, and KNN algorithms obtained similar values close to the MLP algorithm value and the NB and SVM algorithms obtained lower values than the other algorithms. In *Setting*, the same behavior of the previous data set was maintained, being the MLP algorithm with the best value, the PRISM, J48, and KNN algorithms with values similar to the MLP value and finally the NB and SVM algorithms with lower values than other algorithms.

Considering the comparison made of the classification algorithms on the artificial data sets and analyzing it in general. It was observed that the MLP algorithm, in all sets, obtained the best classification performance related to other algorithms. Following the PRISM, J48, and KNN algorithms that obtained their performances very close to the MLP algorithm. Finally, the NB and SVM algorithms have always obtained lower-ranking performances than the other algorithms. Therefore, it was necessary to perform the Friedman statistical hypothesis test to verify if there is a statistical difference between the compared classification algorithms. It is portraying whether one algorithm has better or worse rating performance than another, even if its rating accuracy metric percentage is higher or lower.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

4.5. Friedman Test

As for Friedman's statistical hypothesis test, we proceeded from the comparison made in the previous subsection. The test was implemented in the R programming language, using the RStudio IDE together with the Excel program to transform quantitative data (classification accuracy) into ordinal qualitative data (ranking positions). As the Friedman test requires ordered data, the Excel program was used to generate a ranking for each set of simulated data, and the rankings have the rankings of each classification algorithm in each value of the seed generating randomness. Thus, the algorithm that obtains the highest value of the classification precision metric at a given seed value will be ranked first in the ranking and so on with the other algorithms according to their values. The classification precision metric values of each algorithm and seed were obtained through the 30 executions performed in the comparison of Subsection 4.4. Rankings have the rankings of each algorithm in each seed value, as well as the average rankings for each algorithm. After creating the rankings of each simulated data set, files were generated from them in CSV format with only the placement of the algorithms in each seed value, for import into IDE RStudio, and thus perform the Friedman test.

Table 8. Average ranking placements.

Artificial Data Set	NB	PRISM	J48	KNN	SVM	MLP
Target	3,45	3,45	3,45	3,75	3,45	3,45
Period	6,0	3,85	2,15	3,0	5,0	1,0
Setting	6,0	4,0	3,0	2,0	5,0	1,0

For a better understanding of the placement of the algorithms in each artificial data set is presented Table 8, which reflects directly with the classification accuracy averages of Table 7. Thus, among all artificial data sets, the MLP algorithm was ranked. First, the J48, KNN, and PRISM algorithms alternating in second, third, and fourth place, and the SVM and NB algorithms in fifth and sixth place respectively. This did not happen when there was a tie between the algorithms.

Friedman's test was performed using an external package called "tools for R," using the CSV files generated from the ranking of each artificial data set [38]. Figure 9 presents Friedman's tests performed with the *Target*, *Period* and *Setting* sets.

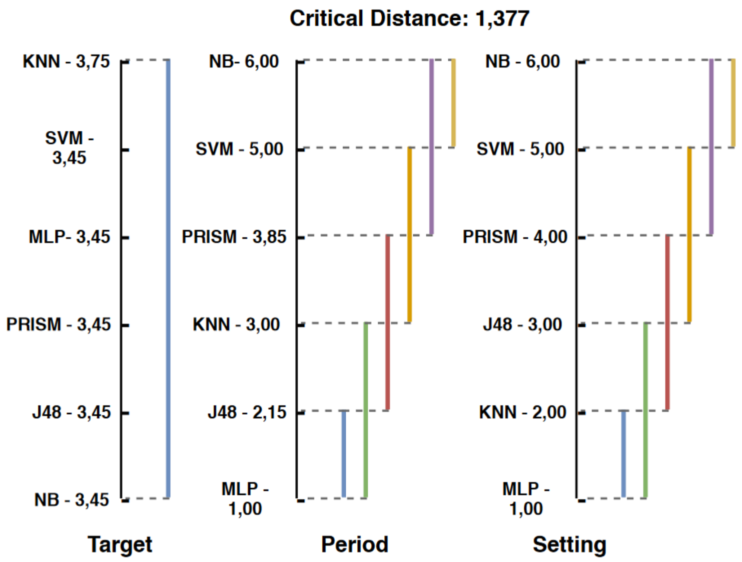

Figure 9. Friedman test.

In the *Target* set, all algorithms are statistically equal and obtained their averages from similar placements, only the KNN algorithm obtained its slightly lower average placement. Starting with

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

the *Period* set, the MLP algorithm obtained the best placement average and the NB algorithm the worst. The MLP, J48, and KNN algorithms were the most outstanding in this set, proving that they are statistically equal. This also happens for the *Setting* dataset, which has the same behavior as the previous set, with only the second and third place settings varying between the KNN and J48 algorithms.

Among all Friedman tests performed with artificial data sets, the algorithms that showed the most satisfactory results were the MLP, J48, KNN algorithms since they were always among the first three ranking positions on the classification accuracy metric and proving that their classification performances are statistically equal in the context of the proposed work. The NB, SVM, and PRISM algorithms have always got the worst ranks in all data sets, except when referring to the *Target* set.

With the tests and comparisons performed so far in this section, it was defined that the J48 algorithm is the most suitable to be used in the application scenario test. Because, taking into consideration the classification accuracy metric and the CPU time metrics analyzed, the algorithm has the best performance in both accurately classifying notifications for which user, time of day, and type of device configuration should be notified for the response time of classification and training of the predictor model. Therefore, in the application scenario test, its behavior will be tested in a real scenario in the context of notification management in smart environments.

4.6. Application Scenario Test

To test the J48 classification algorithm implemented in the PRINM DM engine from the attributes and their values of Figure 5 and Figure 6, the application scenario described in Section 3 was used. The predictive attributes used in the application scenario were the same as those used to generate the artificial *Target, Period*, and *Setting* data sets. Based on this, the J48 algorithm was trained with the three artificial data sets, thus creating three decision trees that ranked for which user, period, and device configuration notifications should be notified throughout the application scenario. The *Target* set tree has a size of 36 nodes, 27 of the leaf nodes. The tree of the *Period* set has a size of 235 nodes, 169 of the leaf nodes. Finally, the tree of the *Setting* set had a size of 224 nodes, 159 of the leaf nodes.

Predictive data were collected from an artificial and a real source. For the artificial source, the data were collected from the users' actions in the application scenario for the values of the attributes *user, profile, environment* and *activity*. For the real source data were collected from the mobile *PRISER: Notification Collector* application developed by [39] for the values of the attributes *status, inPeriod* and *inTarget*. Data from both sources were merged, thus generating three unclassified data sets (test sets). Artificial source data is presented in Table 9, Table 10 and Table 11 for each user.

Table 9. Customer user data.

Period	Profile	Environment	Activity
10:10 - 10:25 ∣ Morning	Guest	Show Room ∣ Public	1
10:25 - 11:50 ∣ Morning	Guest	Sector of Sales ∣ Private	2
16:00 - 16:15 ∣ Afternoon	Guest	Show Room ∣ Public	1
16:15 - 16:30 ∣ Afternoon	Guest	Sector of Sales ∣ Private	2
16:30 - 18:00 ∣ Afternoon	Basic	Office ∣ Restrict	3

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 November 2019 doi:10.20944/preprints201911.0022.v1

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

Table 10. Employee user data.

Period	Profile	Environment	Activity
7:00 - 9:30 | Morning	Basic	Show Room | Public	2
9:30 - 10:00 | Morning	Advanced	Kitchen | Private	1
10:00 - 11:50 | Morning	Basic	Show Room | Public	2
11:50 - 14:00 | Afternoon	Advanced	Kitchen | Private	1
14:00 - 16:30 | Afternoon	Basic	Show Room | Public	2
16:30 - 18:00 | Afternoon	Basic	Sector of Sales | Private	3

Table 11. Owner user data.

Period	Profile	Environment	Activity
7:30 - 8:30 | Morning	Administrator	Sector of Sales | Private	2
8:30 - 10:25 | Morning	Administrator	Office | Restrict	3
10:25 - 11:50 | Morning	Administrator	Sector of Sales | Private	2
11:50 - 14:00 | Afternoon	Administrator	Kitchen | Private	1
14:00 - 16:30 | Afternoon	Administrator	Sector of Sales | Private	2
16:30 - 18:00 | Afternoon	Administrator	Office | Restrict	3
18:00 - 19:00 | Night	Administrator	Kitchen | Private	1
19:00 - 20:00 | Night	Administrator	Office | Restrict	2
20:00 - 22:00 | Night	Administrator	Office | Restrict	1

Real source data was collected by the mobile application installed on three different mobile devices, representing the three users of the application scenario. The application collected notifications from the three mobile phones for 24 hours, generating information in the JSON format of each notification. For each JSON were extracted only the information corresponding to the attributes *status, inPeriod* and *inTarget*, which coincide with the periods of each user within the dealership. Therefore, they were extracted for the client user from 10:10 until 11:50 and 16:00 until 18:00, for the employee user from 7:00 until 18:00 and the owner user from 7:30 until 10 pm Other notifications outside these times respectively of each user were discarded. Therefore, 1377 notifications were received within the defined times for the client user (mobile phone 1) being extracted 209, 813 for the employee user (mobile phone 2) being extracted 399, and 413 for the owner user (mobile phone 3) being extracted 333.

The application scenario test aims to analyze the behavior and classification performance of the J48 algorithm on the classification accuracy metric using predictive data from the test sets and classifying them. However, after merging the predictive data from the two sources for generating test sets, they were also merged into them, data classes following the same logic as the script that generated the artificial data set *Target, Period* and *Setting* . This was necessary because it would not be possible to show in this article all the values of the class attributes that the J48 algorithm would classify from the predictive data of the test sets in the application scenario. Therefore, the test sets merged with the client, employee, and owner-user class data were applied to the decision trees created by the J48 algorithm. The results showed that the J48 algorithm was able to classify with 100 % accuracy new unclassified data inserted in each decision tree. Thus, it is concluded that the J48 algorithm has the proper behavior regarding the classification performance of notifications received in smart environments and that it is the most viable for implementation in the PRINM DM mechanism that will be developed in UBIPRI.

5. Related Works

First, [40] restricts itself to detecting disruptive phone calls that are a major source of annoyance to users. To this end, they evaluated six types of learning algorithms, namely: SVM (*Support Vector Machine*), NB (*Naive Bayes*), KNN (*K-Nearest Neighbors*), RUSBoost, GP (*Genetic Programming*) and AR (*Association Rule learning*). The data set used for the assessment was collected over 16 weeks with the help of a mobile app. Similar to this work, we also used the NB and KNN algorithms, and collected data from a mobile application to create the application scenario test.

Following notification-related studies, [41] developed an architecture of an intelligent notification system that uses classification algorithms to manage the receipt of notifications according to contextual perception and user habits. The system consists of modules that monitor the environment and users, collecting information to send them to a DM engine. The primary relationships with this work are the comparison of the classification algorithms on the classification precision metric, the use of a simulated data set, and the system that implements a classification algorithm in the DM mechanism.

The authors at [42] report in their article a proposal for location verification and user confirmation in smart environments, in the context of notification control and management. User verification and notification control are performed based on parameters such as environment type, user profile type, location, time criteria, priority, and user preferences. The authors' work is also based on one of the modules of UBIPRI, being PRISER. Already this work is based on the modules PRIPRO and PRICRI.

In the work of [43], a system has been developed to reduce manual user efforts by addressing receiving relevant notifications by wireless communication in a university setting. In development, the *Knuth-Morris Pratt* (KMP) algorithm was applied to a real data set with the following attributes: admin, dept, notice, noticereadby, registration details, staff and user table. Similar to this work, attributes related to the context were used to perform the notification management.

The reference work [44] discusses in their article an assessment of an artificial data set in a notification management system. In general, the set has the characteristics of the notification content, user context, and the receiving method, together with the synthetically entered data. In the evaluation, the Fuzzy InferenceSystem (FIS) algorithm was used to verify the behavior of the generated artificial data set. The primary relationship with this work is the generation and use of an artificial dataset for PRINM evaluation.

Table 12 presents a synthesis of related works, pointing to the use of classification algorithms, artificial data sets, if the proposed comparison of algorithms on the classification precision metric, and finally, if statistical hypothesis tests are used. Comparisons of literature with approaches are defined as: (i) *Yes* literature treats the approach; (ii) *No* the literature does not address the approach; (iii) *Partial* The literature partially addresses the approach.

Table 12. Summary of related work.

Work	Authors	Uses Classification Algorithms	Uses Artificial Data	Uses Algorithm Comparison	Uses Hypothesis Tests Statistics
[40]	Smith (2014)	Yes	No	No	No
[41]	Corno (2015)	Yes	Yes	Yes	No
[44]	Fraser (2017)	No	Yes	No	No
[43]	Ghodse (2018)	No	No	No	No
[45]	Martins (2018)	Yes	Yes	Yes	Partial
[42]	Silva (2019)	Yes	No	No	No
This Work	This Work	Yes	Yes	Yes	Yes

The work presented in [45] does not have a direct relationship with the notification management context, but it is part of researches that is related to UBIPRI. Thus, compared to this work on the notification management approach, only the works [44], and [43] do not use classification algorithms to manage notifications, but they still have similarity in the approaches of notification — artificial data sets. Using artificial data sets is not a good practice when it comes to using ML approaches. However, this practice is becoming increasingly applied as in the works [41] and [44]. The works [40], and [42] do not address the comparison of classification algorithms over the classification accuracy metric.

Since there is a range of algorithms to be studied and compared. Only the work [41] addresses this comparison.

It is clear to realize that none of the related works use any statistical hypothesis test as a complement to statistical analysis. This consequently partially affected the solutions developed by the authors, because with the application of a single statistical hypothesis test appropriate to the context of these works, it would be possible to analyze the statistical differences of the algorithms when compared thoroughly. As seen in this paper, the test performed on Subsection 4.5 identified that three classification algorithms have the same classification performance, even though they obtain their distinct classification precision metric values.

It is worth mentioning that the gap of not using statistical hypothesis tests presented in the related works is applied in the context of notification management in smart environments. Therefore, they were useful for the development and elaboration of tests and comparisons that contributed to solving the problem listed in this work. Because of this, this work proposes to perform the activities of delimitation and use of classification algorithms and statistical hypothesis testing, generation of artificial data sets, tests, and comparisons of classification algorithms and application scenario testing. At the end of the work development, it was delimited that the J48 algorithm is the most viable for implementation in the PRINM DM mechanism that will be developed in UBIPRI.

6. Conclusion

This paper presents a comparison of classification algorithms for managing receiving notifications in smart environments. UBIPRI was used as a base, which is an MW that has as its primary objective the treatment of privacy in smart environments. Therefore, it was proposed an architecture that presents the performance of PRINM together with the PRIPRO and PRICRI modules provided by the addressed MW. With it, it was possible to manage notifications that are received in environments that UBIPRI operates, being notified for which user, time of day, and device configuration from the PRINM DM engine.

The activities carried out were the delimitation and use of NB, J48, KNN, MLP, PRISM and SVM algorithms, delimitation and application of Friedman test, generation of artificial data sets, tests, and comparisons of classification algorithms and scenario test of application. The delimited algorithms obtained high efficiency in the tests in which they were applied, satisfactorily contributing to the developed solution of the work. With Friedman's statistical hypothesis test, it was possible to identify statistical differences in the classification performance of the classification algorithms. Regarding the artificial data sets, they were suitable for use. Tests and comparisons identified the best tunable parameters, CPU time for training and classification, and classification accuracy metric values between classification algorithms. Finally, the application scenario test presented the applicability of PRINM on the notification management architecture.

Regarding the use of the Friedman test, it was of great importance for the development of the work solution. For without its application, it would not be possible to identify that the algorithms MLP, J48, and KNN, have the same classification performances. It is thus proving among the three and from the other tests and comparisons, that the J48 algorithm is the most viable to implement in the PRINM DM engine that will be developed in UBIPRI.

The first item for future work is the development of PRINM to assign it in UBIPRI, based on the coined architecture and the implementation of the J48 algorithm in the DM engine. It is also suggested to optimize artificial data sets by inserting new attributes, as well as using other MW modules addressed, so that notification management becomes more meticulous about the privacy that UBIPRI employs in smart environments.

Funding:

This research was supported in part by the Programa de Bolsas Universitárias de Santa Catarina/SC — UNIEDU for Instituições Comunitárias de Ensino Superior and CAPES.

Peer-reviewed version available at Appl. Sci. 2020, 10, 502; doi:10.3390/app10020502

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001. Supported by project PLATAFORMA DE VEHÍCULOS DE TRANSPORTE DE MATERIALES Y SEGUIMIENTO AUTÓNOMO — TARGET. 463AC03.

Project co-financed with Junta Castilla y León, Consejería de Educación and FEDER funds, including a cooperation with the project international cooperation project Control and History Management Based on the Privacy of Ubiquitous Environments—Brazil/Portugal.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems* **2013**, *29*, 1645–1660. doi:10.1016/j.future.2013.01.010.
2. Guerra, C.A.N. Um modelo para ambientes inteligentes baseado em serviços web semânticos. PhD thesis, Universidade de São Paulo, 2007. http://www.teses.usp.br/teses/disponiveis/45/45134/tde-03102007-172316/.
3. Birman, K.P. *Guide to Reliable Distributed Systems: Building High-Assurance Applications and Cloud-Hosted Services*; Springer Science & Business Media, 2012.
4. Tanenbaum, A.S.; Van Steen, M. *Distributed systems: principles and paradigms*; Prentice-Hall, 2007.
5. Leithardt, V.R.Q. UbiPri: middleware para controle e gerenciamento de privacidade em ambientes ubíquos. PhD thesis, UFRGS - Universidade Federal do Rio Grande do Sul, 2015. https://lume.ufrgs.br/handle/10183/147774.
6. Leithardt, V.R.; Correia, L.H.A.; Borges, G.A.; Rossetto, A.G.; Rolim, C.O.; Geyer, C.F.; Silva, J.M.S. Mechanism for Privacy Management Based on Data History (UbiPri-His). *Journal of Ubiquitous Systems and Pervasive Networks* **2018**, *10*, 11–19. doi:10.5383/JUSPN.10.01.002.
7. Alpaydin, E. *Introduction to machine learning*; MIT press, 2009.
8. Mohri, M. Foundations of Machine Learning Lecture 11 **2012**.
9. Kinas, P.G.; Andrade, H.A. *Introdução à análise bayesiana (com R)*; Consultor Editorial, 2017.
10. Pinto, C.M.S.; Gama, J.; do Porto, U. Algoritmos Incrementais para Aprendizagem Bayesiana. *Faculdade de Economia da Universidade do Porto* **2005**.
11. Shiba, M.H.; Santos, R.L.; Quintanilha, J.A.; Kim, H.Y. Classificação de imagens de sensoriamento remoto pela aprendizagem por árvore de decisão: uma avaliação de desempenho. *Anais do XII Simpósio Brasileiro de Sensoriamento Remoto, Goiânia-GO* **2005**, pp. 4319–4326.
12. Holsheimer, M.; Siebes, A. *Data mining: The search for knowledge in databases*; Centrum voor Wiskunde en Informatica, 1994.
13. Crepaldi[1], P.G.; Avila, R.N.P.; de Oliveira[3] Paulo, J.P.M.; Rodrigues, R.; Martins, R.L. Um estudo sobre a árvore de decisão e sua importância na habilidade de aprendizado **2011**.
14. Patil, T.R.; Sherekar, S.; others. Performance analysis of Naive Bayes and J48 classification algorithm for data classification. *International journal of computer science and applications* **2013**, *6*, 256–261.
15. Aha, D.W.; Kibler, D.; Albert, M.K. Instance-based learning algorithms. *Machine learning* **1991**, *6*, 37–66. doi:https://doi.org/10.1007/BF00153759.
16. Aha, D.W. *Lazy learning*; Springer Science & Business Media, 2013.
17. Haykin, S. *Redes neurais: princípios e prática*; Bookman Editora, 2007.
18. Gardner, M.W.; Dorling, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment* **1998**, *32*, 2627–2636. doi:https://doi.org/10.1016/S1352-2310(97)00447-0.
19. Fürnkranz, J.; Gamberger, D.; Lavrač, N. *Foundations of rule learning*; Springer Science & Business Media, 2012.
20. Cendrowska, J. PRISM: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies* **1987**, *27*, 349–370. doi:https://doi.org/10.1016/S0020-7373(87)80003-2.
21. Vapnik, V.N. An overview of statistical learning theory. *IEEE transactions on neural networks* **1999**, *10*, 988–999. doi:10.1109/72.788640.
22. SANTOS, E.M.d.; others. Teoria e aplicação de Support Vector Machines à aprendizagem e reconhecimento de objetos baseado na aparência. **2002**. https://docplayer.com.br/1958837-Teoria-e-aplicacao-

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 502; doi:10.3390/app10020502

de-support-vector-machines-a-aprendizagem-e-reconhecimento-de-objetos-baseado-na-aparencia-eulanda-miranda-dos-santos.html.

23. Chand, N.; Mishra, P.; Krishna, C.R.; Pilli, E.S.; Govil, M.C. A comparative analysis of SVM and its stacking with other classification algorithm for intrusion detection. 2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Spring). IEEE, 2016, pp. 1–6. doi:10.1109/ICACCA.2016.7578859.
24. Dietterich, T.G. Statistical tests for comparing supervised classification learning algorithms. *IWS IWS FV RUVW HGX SXE WJG SDSHUV* **1997**. doi:10.1162/089976698300017197.
25. Lehmann, E.L.; Romano, J.P. *Testing statistical hypotheses*; Springer Science & Business Media, 2006.
26. Yazici, B.; Yolacan, S. A comparison of various tests of normality. *Journal of Statistical Computation and Simulation* **2007**, *77*, 175–183. doi:https://doi.org/10.1080/10629360600678310.
27. Billings, S.; Zhu, Q. Nonlinear model validation using correlation tests. *International journal of control* **1994**, *60*, 1107–1120. doi:https://doi.org/10.1080/00207179408921513.
28. Estatistica, A. Teste do Qui-quadrado para análise da associação estatística entre variáveis categóricas, 2018. http://analise-estatistica.pt/2015/12/teste-do-qui-quadrado-para-analise-da-associacao-estatistica-entre-variaveis-categoricas.html.
29. Milone, G. *Estatística: geral e aplicada*; Pioneira Thomson Learning, 2004.
30. Weisberg, H.; Weisberg, H.F. *Central tendency and variability*; Number 83, Sage, 1992.
31. Dutta, A. Parametric and nonparametric event study tests: A review. *International Business Research* **2014**, *7*, 136–142. doi:10.5539/ibr.v7n12p136.
32. Stacey, R. Statistical Tests for Comparing Machine Learning and Baseline Performance, 2019. https://towardsdatascience.com/statistical-tests-for-comparing-machine-learning-and-baseline-performance-4dfc9402e46f.
33. Zimmerman, D.W.; Zumbo, B.D. Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks. *The Journal of Experimental Education* **1993**, *62*, 75–86. doi:https://doi.org/10.1080/00220973.1993.9943832.
34. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research* **2006**, *7*, 1–30.
35. Martins, J.A.; Rolim, C.O.; Leithardt, V.R. Uma Proposta de Análise de Algoritmos de Machine Learning para o Envio e Recebimento de Notificaç oes em Ambientes Inteligentes **2018**. https://www.setrem.com.br/erad2019/data/pdf/forum_ic/192073.pdf.
36. Witten, I.H.; Frank, E.; Trigg, L.E.; Hall, M.A.; Holmes, G.; Cunningham, S.J. Weka: Practical machine learning tools and techniques with Java implementations **1999**.
37. Abernethy, M. Data mining with WEKA, Part 2: Classification and clustering. *IBM Dev* **2010**.
38. Kourentzes, N. TStools for R, 2018.
39. Silva, L.A.; dos Santos, D.A.; Dazzi, R.L.S.; Silva, J.S.; Leithardt, V.R.Q. PRISER-Utilização de BLE para localização e notificação com base na privacidade de dados. *Revista Eletrônica Argentina-Brasil de Tecnologias da Informação e da Comunicação* **2018**, *2*. doi:http://dx.doi.org/10.5281/zenodo.1336806.
40. Smith, J.; Lavygina, A.; Ma, J.; Russo, A.; Dulay, N. Learning to recognise disruptive smartphone notifications. Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services. ACM, 2014, pp. 121–124. doi:10.1145/2628363.2628404.
41. Corno, F.; De Russis, L.; Montanaro, T. A context and user aware smart notification system. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). IEEE, 2015, pp. 645–651. doi:10.1109/WF-IoT.2015.7389130.
42. Silva, L.A.; Leithardt, V.R.Q.; Rolim, C.O.; González, G.V.; Geyer, C.F.; Silva, J.S. PRISER: Managing Notification in Multiples Devices with Data Privacy Support. *Sensors* **2019**, *19*, 3098. doi:https://doi.org/10.3390/s19143098.
43. Ghodse, A.G.; Kshirsagar, V.; Nagori, M. Pattern Based Smart Notification in Education. 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE, 2018, pp. 1–6. doi:10.1109/ICCCNT.2018.8493981.
44. Fraser, K.; Yousuf, B.; Conlan, O. An in-the-wild and synthetic mobile notification dataset evaluation. 2017 Intelligent Systems Conference (IntelliSys). IEEE, 2017, pp. 510–519. doi:10.1109/IntelliSys.2017.8324343.

Peer-reviewed version available at *Appl. Sci.* **2020**, *10*, 502; doi:10.3390/app10020502

45. Martins, J.A.; Silva, L.A.; Rolim, C.O.; Leithardt, V.R. Uma Avaliação de Algoritmos para a Classificação de Perfis Evolutivos em Ambientes Inteligentes. *Anais SULCOMP* **2018**, *9*. http://periodicos.unesc.net/sulcomp.