Available online at www.sciencedirect.com

SCIENCE DIRECT®

Mathematical and Computer Modelling 44 (2006) 790–806

MATHEMATICAL AND COMPUTER MODELLING

www.elsevier.com/locate/mcm

Inductive learning models with missing values[☆]

I. Fortes[a,*], L. Mora-López[b], R. Morales[b], F. Triguero[b]

[a] *Dept. Matemática Aplicada, E.T.S.I. Informática, Univ. Málaga, Campus Teatinos, Málaga 29071, Spain*
[b] *Dept. Leng. y C. de la Computación, E.T.S.I. Informática, Univ. Málaga, Campus Teatinos, Málaga 29071, Spain*

Received 30 September 2004; received in revised form 7 February 2006; accepted 21 February 2006

Abstract

In this paper, a new approach to working with missing attribute values in inductive learning algorithms is introduced. Three fundamental issues are studied: the splitting criterion, the allocation of values to missing attribute values, and the prediction of new observations. The formal definition for the splitting criterion is given. This definition takes into account the missing attribute values and generalizes the classical definition. In relation to the second objective, multiple values are assigned to missing attribute values using a decision theory approach. Each of these multiple values will have an associated confidence and error parameter. The error parameter measures how near or how far the value is from the original value of the attribute. After applying a splitting criterion, a decision tree is obtained (from training sets with or without missing attribute values). This decision tree can be used to predict the class of an observation (with or without missing attribute values). Hence, there are four perspectives. The three perspectives with missing attribute values are studied and experimental results are presented.

Keywords: Missing values; Decision tree; Decision theory; Data mining; Machine learning

1. Introduction

One of the main tools of data mining is rule induction from raw data represented by a database. Real-life data are frequently imperfect: erroneous, incomplete, uncertain and vague. In this paper we have investigated one of the forms of data incompleteness: missing attribute values. Missing data randomness can be divided into three classes, as proposed by [1]:

- **Missing completely at random (MCAR).** This is the highest level of randomness. It happens when the probability of an instance (case) with a missing value for an attribute does not depend on either the known values or the missing data. In this level of randomness, any missing data treatment method can be applied without risk of introducing bias into the data.
- **Missing at random (MAR).** When the probability of an instance with a missing value for an attribute may depend on the known values, but not on the value of the missing data itself.

☆ This work has been partially supported by MOISES project number TIC 2002–04019–C03–02 of the CICYT-Spain.

* Corresponding address: Universidad de Malaga, Campus de Teatinos, Dpto. Matematica Aplicada E.T.S. De Ingenieria Informatica, Bulevar Luis Pasteur s/m, Malaga 29071, Spain.
E-mail address: ifortes@ctima.uma.es (I. Fortes).

0895-7177/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mcm.2006.02.013

- **Not missing at random (NMAR).** When the probability of an instance with a missing value for an attribute could depend on the value of that attribute.

This paper only considers the MCAR model and uses it to introduce missing values when performing experiments. We assume that the format of input data files is in the form of a table, which is called a decision table. In this table, each column represents one attribute, which represents some feature of the examples, and each row represents an example (training case) by all its attribute values. The domain of each attribute may be either symbolic or numerical. The last column is the target class, i.e. the values of the other attributes determine the target class values.

Methods for treating missing data

In general, two groups of algorithms to process databases that contain missing values can be distinguished. The first group concerns unsupervised algorithms that do not use target class values. The second group is supervised algorithms that use target class values and which are commonly implemented by using supervised inductive learning algorithms [2]. Other works on dealing with missing values in supervised algorithms have been undertaken in the context of rough set theory [3–8]. The proposal in [3] is to reduce knowledge by eliminating only the non-essential information in order to classify or to make decisions; this is based on the discernibility function. Another direction, related to the previous one, is considered in [8], where missing attributes are classified in three groups: "lost", "non relevant" and "attribute–concept value"; in each case, a decision to complete or to expand or to not include the data is taken.

For unsupervised learning, statistically intensive methods have been developed mainly to manage incomplete data:

- Conventional methods
  A decision could be to discard the training cases with missing attribute values. If the training cases with missing attribute values are discarded (listwise deletion and pairwise deletion methods), we would decrease the ability to find patterns, and obtain worse predictions when using the decision tree in Data Mining, because fewer training cases are taken into consideration. Thus, our decision, and the main goal of this paper, is to adapt the algorithms to work with missing attribute values.
- Global imputation based on a missing attribute
  If we look at the other values taken by a variable with a missing item, and we find that some of them are more frequent than others, then we may decide to use this fact to assign the most frequent value to a missing one. We fill the missing attribute values with a measure of central tendency (see [9–11] for a comparison of mean imputation and the most common imputation attribute value). These methods are usually regarded as inadequate, because the standard deviation of the sample is underestimated even when data are MCAR.
- Parameter estimation and global imputation based on non-missing attributes
  These have proven to be very effective in many situations, especially the EM algorithm [1] and multiple imputation [12]. The main problem with these techniques is the need for strong model assumptions, which are usually difficult to justify in the knowledge discovery process.

Our method for working with missing attribute values is based on a well-known approach to supervised inductive learning algorithms, that is, tree-based classification. From incomplete data, a decision tree is built, which is also able to classify unseen records with missing values. For a short overview of tree-based classification techniques, see [13,14].

The methods cited above fill all the missing values, and in the tree-based classification method there are cases in which it is not necessary to fill all the missing values, because they are unnecessary in tree construction. This fact simplifies the computation. Besides this, it is not necessary to model assumptions in decision tree classification.

The original idea of generating a decision tree from a set of training cases arose in [15–18], where experiments with several implementations of concept learning systems (CLS) are described. Other studies include [19], ID3 [16–18,20], PLS1, and ASSISTANT86 [21]. The basic TDIDT (Top Down Induction of Decision Tree) algorithms [18, 20] assume that the attribute values for all training cases of the training set are known.

If the training set contains missing attribute values, we must decide how to work with them. The most famous algorithm of this kind is C4.5 [22], which has proved to be very effective (see [23] for an experimental comparison among supervised inductive learning algorithms, CLIP4, Naive–Bayes, and C4.5).

CLIP4 [24,25] is a rule-based algorithm that works in three phases. It works only with discrete data.

Naive–Bayes [26] is a classification technique based on computing probabilities. It works only with discrete data.

C4.5 [22] is a decision tree algorithm. The usual splitting criterion to select an attribute is the gain ratio [20] based on the information concept defined by Shannon [27]. Each tree level is generated by dividing the data at a given node into a number of subsets, which are represented by branches. For each division, the gain ratio is used to select the best attribute, whose values are used to divide the data into subsets. Each subset contains data that take on one of the values of the selected attribute. It can work with both discrete and continuous data.

It has been shown in [23] that, in general, supervised imputation algorithms have better performance compared to unsupervised algorithms. Another important trend shows that increasing the number of attributes and examples results in increasing the quality of imputation for the supervised imputation methods. Besides this, several measures indicate that the supervised imputation method used in C4.5 has the best overall performance. A probabilistic approach is adopted: each case in each subset has an associated weight representing the probability that the case belongs to each subset. If the case has a known outcome, this weight is 1; if the case has an unknown outcome, the weight is just the probability of outcome of each value at this point. Each subset is then a collection of possibly fractional cases.

In [28], the problem of missing values is studied from the point of view of several committee methods (Bagging, Boosting, SASC, and SASCMB). Committee methods are methods for groups of classifiers/predictors to select one or more options from many, taking into account the individual classification/prediction of the group members. It is shown that SASCMB is the most robust. Our work occupies a previous stage. We can assign values to missing values, and then the committee methods can be applied.

2. Our approach to working with missing attribute values

The main contributions of this work are the following: the definition of a general splitting criterion, taking into account the missing attribute values and generalizing the classical one; the assignation of values to missing attribute values by means of a decision theory approach, taking into account the information of observations and classes with associated parameters (confidence and error); and finally the prediction of new observations (in all perspectives) with two associated parameters.

This approach to processing missing attribute values was introduced in a brief and preliminary version in [30]. It can be integrated straightforwardly in any software dealing with inductive learning: academic software [31,32] or commercial software [22].

Our approach to working with missing attribute values is a supervised inductive learning algorithm, specifically a decision tree algorithm. The method defined in [29] has some similarities to our method. However, we have developed a complete process from data with missing values to decision trees, by including the prediction process.

We assume that the unknown values (in the training cases or in the observations) are missing completely at random. To select an attribute in the building process of the decision tree, a general splitting criterion will be formally defined. To do so, the missing value is first taken as a new value of the attribute. Second, the splitting criterion takes into account this new value. Finally, a reduced weight is assigned to the attribute with missing values. In this way, every splitting criterion defined as working without missing values is adapted to work with missing attribute values.

Let us suppose that an attribute has been selected. A node of the decision tree is then expanded and several child nodes are generated (one child for each known value of the selected attribute).

The training cases with the selected attribute value known are split in the child nodes. The training cases with exactly that attribute value missing will appear in every child node. Each of these training cases in each child node will have two associated parameters: confidence and error (real numbers). Obviously, training cases with that value known have confidence one and error zero associated with their class. The process to obtain the values of these parameters is performed within a Decision Theory enviroment [33]. The attribute with missing values is interpreted as a statistical parameter. We also take into account the class to which the training case with an unknown value belongs. With this information, we calculate the Bayes' probability (confidence parameter) and the Bayes' risk (error parameter). We calculate the Bayes' risk through the loss matrix obtained from the distance between the different values of the attribute.

After applying our method by using a specific splitting criterion, a decision tree is obtained (from training sets with or without missing attribute values). Regarding the building of the decision tree, if the value of the current node's attribute is unknown in the training case, then its parameters are calculated as explained before. Then the training case is included in the subsets of the partition in the current node, the probabilities are multiplied and the risks added. This decision tree can be used to predict the class of an observation (with or without missing attribute values). Hence, there are four perspectives. The three perspectives with missing attribute values are studied and experimental results are included. In all these predictions, we give a confidence parameter and an error parameter calculated from the number of training cases associated with each node and from the accumulated probabilities and accumulated risks associated with each node of the decision tree.

The rest of this paper is organized as follows. The next section introduces the notation used and preliminary concepts. The fourth section is devoted to explaining how to modify the splitting criterion and how to adapt the algorithms to work with missing attribute values. Prediction with missing attribute values is explained in Section 5. Finally, experimental results are presented along with conclusions and future work.

3. Notation and previous concepts

In this section, we provide some basic notions about Decision Trees [20,18] and the necessary Decision Theory concepts [33].

3.1. Decision Trees

Let *TRS* be a training set. Let $X_i, i = 1, \dots, f$ be the attributes in the training cases of *TRS*.

The attributes take values denoted by $\{O_{i1}, O_{i2}, \dots, O_{im(i)}\}$, where $m(i)$ is the function $(m(i) : \{1, \dots, f\} \to \mathbb{N})$. Let Y be a special attribute called "class" which takes values denoted by $\{y_k, k = 1, \dots, h\}$. This is the set formed by h classes that belong to the training cases of *TRS*. A training case $e \in TRS$ is represented by $e = (X_1(e), X_2(e), \dots, X_f(e), Y(e))$, where $Y(e)$ is the class that belongs to e and $X_i(e)$ is the value of the attribute X_i evaluated in e, and $e-$ denotes a training case with missing attribute values. By analogy with e and $e-$, the observations b and $b-$ are defined as: $b = (X_1(b), X_2(b), \dots, X_f(b))$, where $X_i(b)$ is the value of the attribute X_i evaluated in b, and $b-$ denotes an observation with missing values. b and $b-$ correspond to cases in some data testing sets.

This method works with symbolic and numerical attributes (continuous and discrete). If there are continuous attributes, they are processed in a discretization process.

The generation of a Decision Tree from a training set *TRS* is as follows:

- At each node, for each attribute X_i, a test X is chosen that makes a partition over the values of the attribute. This partition is denoted by $\{\theta_j, j = 1, \dots, n(i)/n(i) \le m(i); n : \{1, \dots, f\} \to \mathbb{N}\}$. The sets θ_j are also called values. (The algorithm's basic version [20,18] always assumes that $n(i) = m(i)$ for the partition.)
- An attribute is chosen (by means of any splitting criterion) in order to expand the decision tree.
- Partitioning the current training set. The tree is expanded in that node (new child nodes are obtained under that node) and the training set (*TRS*) is partitioned among the child nodes: $\{T_1, T_2, \dots, T_{n(i)}\}$, where $T_j = \{e \in TRS : X_i(e) \in \theta_j\}$ $j = 1, \dots, n(i)$.
- In each child node, the process is applied again.

The tree-building process tries to build a tree that reveals the structure of the domain and therefore has predictive power. To achieve this, we need a significant number of cases at each leaf; in other words, the partition must have as few blocks as possible. The problem of finding the smallest decision tree consistent with a training set is NP-complete [34]. Therefore, the solution for each problem depends a great deal on the choice of suitable tests and the suitable splitting criterion.

3.2. A Decision Theory problem with risk

Now, we give three basic definitions related to Decision Theory. These definitions include its relation to our problem. We will use these concepts in the next section. An expanded version can be found in [33].

Definition 1. A decision problem is defined as $(\Omega, \mathcal{A}, L)$, where

$\Omega = \{\theta_j;\ j = 1, \dots, n(i)\}$ is the parametric space (values of the partition of the attribute X_i)

$\mathcal{A} = \{a_r;\ r = 1, \dots, o\}$ is the space of actions for the considered attribute X_i (multiple values that will be assigned to every missing attribute value)

$L : A \times \Omega \to \mathbb{R}$ is the loss function $C_{rj} = L(a_r, \theta_j)$. The matrix defined by L is called the loss matrix. In our case, $\Omega = \mathcal{A}$.

With a decision problem, we try to minimize the loss incurred when we assign the value to each missing attribute value. A loss function is defined for each decision and each of the classes. The function represents the loss incurred when a specific decision is made conditional on a specific class. The loss function is unit-less. The loss function can be a combination of costs, risk, etc. The purpose of the loss function is to quantify performance. For example: if a patient suffering from heartburn is sent to the emergency room, it creates unnecessary inconvenience for the patient and a monetary cost to the emergency room treating the patient; in contrast, if a patient suffering a myocardial infarction (heart attack) is told to stay home and see if the symptoms get worse, then they may suffer irreversible harm, or even possible death. In both of these examples, the decisions would be classified as wrong, but one is clearly more wrong than the other. The loss function synthesizes all of the various factors to form real-valued quantities that can be compared numerically.

Now, we define an experiment as a situation involving chance or probability that leads to results called outcomes; for example, if a single six-sided die is rolled, then that is an experiment.

Definition 2. An experiment $\mathcal{E}$ is $\mathcal{E} = \{Y, P(Y), P(Y \mid \theta), \theta \in \Omega\}$, where
Y is the random variable "class"
$Y = \{y_k;\ k = 1, \dots, h\}$ is the sample space (possible outcomes of an experiment)
$P(Y)$ are the events
$P : Y \times \Omega \to [0, 1]/P(Y \mid \theta) = p(Y = y_k \mid \theta_j)$ is the matrix of the experiment. Every element of the matrix represents a conditional probability.

Definition 3. The "a posteriori" risk corresponding to a_r and y_k is defined as the function $R : A \times Y \to \mathbb{R}^+$:

$$R(a_k \mid y_k) = \sum_{j=1}^{n(i)} L(a_r, \theta_j) p(\theta_j \mid y_k).$$

The "a posteriori" risk is the expected value of the loss with respect to the conditional probability. When using the 0–1 loss function, this risk is the probability that a missing attribute value is incorrectly labeled.

4. Building decision trees with missing attribute values

In this section, the core of the paper is developed, i.e. a new approach to working with missing attribute values in TDIDT algorithms is presented. We deal with three basic points: first, we modify the splitting criterion in order to include missing attribute values; second, we explain how to assign values to the training cases with missing attribute values; and, finally, these training cases are added to subsets of the partition.

4.1. Adapting a splitting criterion

Let us assume that the current node is an intermediate node. Our universe is *TRS*. To simplify the notation, we denote the actual training set of *TRS* in the current node as T.

The philosophy of the splitting criterion (defined below) is:

1. Test X is chosen using a measure similar to the gain ratio, adjusted by a factor that depends on the number of records complete on the tested attribute.

   The gain ratio measures the information that is gained by partitioning T in accordance with test X. The gain ratio selects a test to maximize this information gain (which is also known as the mutual information between test X and the class). It is defined as:

   $$\text{gain}(X) = \text{info}(T) - \text{info}_X(T)$$

   where $\text{info}(T)$ measures the average amount of information needed to identify the class of a case in T (this quantity is also known as the entropy of set T). $\text{info}_X(T)$ is similar to $\text{info}(T)$, but after T has been partitioned in accordance with the n outcomes of a test X; the expected information requirement can be found as the weighted sum over the subsets.

2. Every incomplete record is then assigned to all partitions, with a probability which depends on the size of the partition.
3. When an unseen record has to be classified but the testing attribute is unknown, all possible paths are examined, leading to more than one possible class.

Let m be a function (data score) that is used with known values only (with this function, we will introduce a splitting criterion that uses missing values). This measure decides the partition of a set of training cases to build the decision tree. We are going to define the generalization of the gain ratio:

$$m : \mathcal{P}(TRS) \to \mathbb{R}^+ \quad \text{where } \mathcal{P}(TRS) \text{ is the power set of } TRS.$$

If we choose m as the gain ratio, the measure that expresses the improvement in the classification power of the tree after the partition can also be used in the presence of missing values. However, after partitioning, the gain ratio has an inherent bias in some situations [22]. In order to avoid this bias, the function m' with the name split info was introduced in [22], and we adapt its use to cases with missing attributes.

For X_i, given the function m and test X, function m' (split info) is defined as follows:

$$m'_X : \{X_i, i = 1, \dots, f\} \times \mathcal{P}(TRS) \to \mathbb{R}^+$$

$$m'_X(X_i, T) = \sum_{j=1}^{n(i)} \frac{|T_j|}{|T|} \times m(T_j) \quad i = 1, \dots, f$$

where $T = T_1 \cup T_2 \cup \dots \cup T_{n(i)}$.

m' is the weighted sum of the data score for the children of the current node (with data T).

A general criterion is defined below, where the missing values are considered in order to select the attribute in the current node. Let $X_i(e-) = ?$, that is, the value of the X_i attribute is unknown in the training case $e-$. From test X, we obtain the outcomes of X_i to partitioning in the current node. Let $\{\theta_j, j = 1, \dots, n(i)\}$ be the partition of the values of X_i (outcomes) given by X. Let us assume that the ratio between the amount of known values of the attribute X_i and the cardinal of the set TRS is F_i.

The gain information is the difference between m and m'. This gain information must be affected by F_i in a proportional way. For this reason:

Definition 4. Let X_i be an attribute. Let X be the test associated with X_i. Function Δ is defined as:

$$\Delta_X : \{X_i, i = 1, \dots, f\} \times \mathcal{P}(TRS) \to \mathbb{R}^+$$

$$\Delta_X(X_i, T) = F_i \times (m(T) - m'_X(X_i, T)) \quad i = 1, \dots, f.$$

Δ is the change in score that results from the split, multiplied by the fraction of cases in which X_i is present. The Δ measure ignores cases in which X_i is missing.

Up until now, we have only considered the known values of attribute X_i. In the following definition, the unknown value of attribute X_i is taken into account. All the training cases in this situation belong to a new subset (only used to measure the disorder in the set of cases) [22].

Definition 5. Let X_i be an attribute. Let X be the test associated with X_i. The splitting criterion $m''_X(X_i, T)$, $i = 1, \dots, f$, is defined in the subsets of the partition $\{T_1, T_2, \dots, T_{n(i)}, T_{n(i)+1}\}$, where the subset $T_{n(i)+1}$ is defined as $T_{n(i)+1} = \{e- \in T / X_i(e-) = ?\}$.

m'' is the same as m' when '?' is treated as another state for attribute X_i.

Finally, the splitting criterion follows the idea of [22].

Definition 6. Let X_i be an attribute. Let X be the test associated with X_i. Function Δ' is defined as:

$$\Delta'_X(X_i, T) = \Delta_X(X_i, T) / m''_X(X_i, T) \quad i = 1, \dots, f.$$

For every attribute X_k, we compute the value $\Delta'_X(X_k, T)$. Let X_i be the attribute such that $\Delta'_X(X_i, T)$ is maximum. Then the attribute X_i is selected to expand the tree in the current node (that is, to obtain new child nodes).

Table 1
The training set *TRS*

	$X_1(e)$	$X_2(e)$	$X_3(e)$	$X_4(e)$	$Y(e)$
e_1	Sunny	75	70	Yes	Play
e_2	Sunny	80	90	Yes	Don't Play
e_3	Sunny	85	85	No	Don't Play
e_4	Sunny	72	95	No	Don't Play
e_5	Sunny	69	70	No	Play
e_6	?	72	90	yes	Play
e_7	Overcast	83	78	No	Play
e_8	Overcast	64	65	Yes	Play
e_9	Overcast	81	75	No	Play
e_{10}	Rain	71	80	Yes	Don't Play
e_{11}	Rain	65	70	Yes	Don't Play
e_{12}	Rain	75	80	No	Play
e_{13}	Rain	68	80	No	Play
e_{14}	Rain	70	96	No	Play

We have formally defined the concept of a splitting criterion to work with missing attribute values. Our development considers a general splitting criterion m. Thus, it can be adapted to every particular splitting criterion.

Example 1. This simple example, taken from [22], will be used throughout the paper. Some climatological characteristics are observed, then we decide to play or not to play. Let X_1 be the outlook, X_2 the temperature, X_3 the humidity, X_4 be windy, and Y the class. In Table 1 we show the training set *TRS*.

The tests X for each X_i are chosen. Let us assume that, for X_1, the partition is θ_1 = sunny, θ_2 = overcast, θ_3 = rain. Let us also assume that the splitting criterion m is the gain ratio criterion:

$$m(T) = \frac{-8}{13}\log_2\left(\frac{8}{13}\right) - \frac{5}{13}\log_2\left(\frac{5}{13}\right) = 0.961 \text{ bits}$$

$$\begin{aligned} m'_X(X_1,T) &= \frac{5}{13}\left(\frac{-2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)\right) \\ &\quad + \frac{3}{13}\left(\frac{-3}{3}\log_2\left(\frac{3}{3}\right) - \frac{0}{3}\log_2\left(\frac{0}{3}\right)\right) \\ &\quad + \frac{5}{13}\left(\frac{-3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right) = 0.747 \text{ bits} \end{aligned}$$

$$\Delta_X(X_1,T) = \frac{13}{14}(0.961 - 0.747) = 0.199 \text{ bits}$$

$$\begin{aligned} m''_X(X_1,T) &= \frac{-5}{14}\log_2\left(\frac{5}{14}\right) - \frac{3}{14}\log_2\left(\frac{3}{14}\right) \\ &\quad - \frac{5}{14}\log_2\left(\frac{5}{14}\right) - \frac{1}{14}\log_2\left(\frac{1}{14}\right) = 1.809 \text{ bits} \end{aligned}$$

$$\Delta'_X(X_1,T) = 0.199/1.809 = 0.110 \text{ bits.}$$

Let us assume that X_1 maximizes $\Delta'_X(X_i,T)$. Then, the following step is the partition of the training set.

4.2. Assigning multiple values

C4.5 uses a probabilistic approach to handling missing data. Given a training set T, C4.5 finds a suitable test, based on a single attribute, that has one or more mutually exclusive outcomes $\theta_1, \theta_2, \dots, \theta_n$. T is partitioned into subsets $T_1, T_2, \dots, T_n$, where T_i contains all the training cases in T that satisfy the test with outcome θ_i. The same algorithm is applied to each subset T_i until a stop criterion is obeyed. Once a test based on an attribute X is chosen, C4.5 uses a probabilistic approach to partition the training cases with missing values in X. When a training case in T with known value is assigned to a subset T_i, this indicates that the probability of that training case belonging to subset T_i is 1

Table 2
Contingency table

	θ_1	θ_2	
Y_1	5	0	5
Y_2	0	5	5
	5	5	10

Table 3
Contingency table 2

	θ_1	θ_2	
Y_1	7.5	2.5	10
Y_2	2.5	7.5	10
	10	10	20

Table 4
Adjusted frequencies

	θ_1	θ_2	
Y_1	10	0	10
Y_2	0	10	10
	10	10	20

and to all other subsets is 0. When the value is not known, only a weaker probabilistic statement is made from C4.5. This associates with each training case in T_i a weight representing the probability of that training case belonging to T_i. If the training case has an unknown value, this training case is assigned to all partitions with different weights for each one. The weight for the partition T_i is the probability of that training case belonging to T_i. This probability is estimated as the sum of the weights of training cases in T known to satisfy the test with outcome θ_i, divided by the sum of weights of the cases in T with known values on the attribute X.

The idea of C4.5 working with missing attribute values is based on the concept that cases with missing values are distributed across the values of the attribute with missing vales in proportion to the relative frequency of these values in the training set.

C4.5 only considers the values of the attribute (it does not consider the class). If the goal is to make predictions rather than to classify, then a new approach is in order. Our approach is based on considering the values of the attributes and the class too. Note that, in a training case, values of attributes determine class values. This relationship must be taken into account.

Example 2. Consider the 2×2 contingency Table 2 of those cases whose value for attribute X is known, where Y_i and θ_i $(i = 1, 2)$ represent class and attribute values, respectively. Suppose that there are another five cases of class 1 and five cases of class 2 with missing values for attribute X. Then, if we adjust the frequency counts according to the column proportions, Table 3 can be derived.

The reason why this method can give such unsatisfactory estimates for missing values is that the procedure takes no account whatsoever of the structure in the database. Missing value estimates are assigned merely on the basis of prior attribute value probabilities. In our method, the estimates are made conditional upon class membership. Thus, in the example just considered, the adjusted frequencies are shown in Table 4.

This is clearly preferable.

Let us turn our attention to the assignation of values to the missing attribute values. There are h classes (recall Section 3). Thus, every training case has an associated vector with $2h$ components: the h first components are probabilities (one for every class) and the h last components are risks (one for every class).

In a Decision Theory problem with risk, we have a parameter with an unknown value. We must take a decision (called action). This decision is to assign one or several values to that parameter. We aim at minimizing the losses. In our case, the parameter is X_i. In the training case $e-$, the value of the X_i attribute is unknown, that is, $X_i(e-) = ?$

Let $\Omega = \{\theta_j \;\; j = 1, \ldots, n(i)\}$ be the values of X_i in the e training cases. We consider attributes to be random variables.

We have prior information about the possible values of the attribute; that is, $p(X_i(e) \in \theta_j, e \neq e-) = p(\theta_j)$. This preliminary information is taken from the frequencies of the values of X_i.

The action is to take a decision regarding the value of $X_i(e-)$. To do so, for each k we calculate $p(Y(e-) = y_k \mid X_i(e-) \in \theta_j),\;\; j = 1, \ldots, n(i)$. The decision is taken with the help of an experiment. This experiment is performed with a random variable whose probability distribution is related to the values θ_j. In this case, the random variable is $Y =$ "class" and the values of the sample space are $y_k,\;\; k = 1, \ldots, h$. Clearly, the distribution of Y is related to the values θ_j. Next, we apply Bayes' theorem and obtain $p(X_i(e-) \in \theta_j \mid Y(e-) = y_k)$.

The decision about the value of an attribute has an associated risk that indicates the weighted loss caused by this decision. These losses are represented in a matrix called the loss matrix, which is used to calculate the risk associated with the decision. Every element of the loss matrix measures our belief (a real number belongs to [0,1]) about how far or how near the assigned value is from the true value of the attribute. This idea is related to the distance concept. To calculate the loss matrix, we take into account all attribute types. We classify these into Nominal (Order or Non-Order) attributes and Linear (Continuous or Discrete) attributes.

Let us assume that we have sets of values θ_i and θ_j. By considering the attribute types, we define a loss function as follows:

$$L(\theta_i, \theta_j) = \begin{cases} d(\theta_i, \theta_j) & \text{linear} \\ 1/n(i) & \text{nom. non-order and} \quad \theta_i \cap \theta_j = \emptyset \\ 0 & \text{nom. non-order and} \quad \theta_i \cap \theta_j \neq \emptyset \\ |\mathrm{ord}(i) - \mathrm{ord}(j)|/n(i) & \text{nominal order and} \quad \theta_i \cap \theta_j = \emptyset \\ 0 & \text{nominal order and} \quad \theta_i \cap \theta_j \neq \emptyset \end{cases}$$

where $\mathrm{ord}(i)$ is the ordinal position of the outcome θ_i.

Finally, it is necessary to define the distance between two sets when the attribute is linear, that is, $d(\theta_i, \theta_j)$. Several definitions for calculating distances between attribute values can be found in [35]. We have adapted and normalized these distance functions to our case to measure distances between sets of values from an attribute.

Definition 7. The following are some possible functions d:

$$d_1(\theta_i, \theta_j) = \frac{\|\theta_i, \theta_j\|}{\sum\limits_{k=1, k\neq i}^{n(i)} \|\theta_i, \theta_k\|}$$

$$d_2(\theta_i, \theta_j) = \frac{\|\theta_i, \theta_j\|^2}{\sum\limits_{k=1, k\neq i}^{n(i)} \|\theta_i, \theta_k\|^2}$$

where $\|\theta_i, \theta_j\|$ is defined as

$$\|\theta_i, \theta_j\| = \min_{o \in \theta_i, o' \in \theta_j} |o - o'|.$$

Example 3. Let us use the data in Example 1. Let us assume that the first selected attribute to expand the first node of the decision tree was X_1. Let us suppose that test X for the attribute X_1 gave the finest partition. Let us assume that the training case e_6 is $e_6 = (?, 72, 90, \text{yes}, \text{Play})$. Then, we assign multiple values to this missing value.

Table 5
The loss matrix

A	Ω		
	θ_1 = sunny	θ_2 = overcast	θ_3 = rain
Sunny	0	1/3	2/3
Overcast	1/3	0	1/3
Rain	2/3	1/3	0

Let $p(\text{sunny}) = 5/13$, $p(\text{overcast}) = 3/13$, $p(\text{rain}) = 5/13$ be the prior information. In Table 5 the losses are shown.

The "a posteriori" probabilities are
$p(\text{sunny} \mid \text{play}) = 0.25$, $p(\text{overcast} \mid \text{play}) = 0.375$, $p(\text{rain} \mid \text{play}) = 0.375$.

Therefore, the risks associated with these decisions and calculated from Table 5 are the following:

$$\begin{aligned} R(\text{sunny} \mid \text{play}) &= p(\text{sunny} \mid \text{play})L(\text{sunny}, \text{sunny}) \\ &\quad + p(\text{overcast} \mid \text{play})L(\text{sunny}, \text{overcast}) \\ &\quad + p(\text{rain} \mid \text{play})L(\text{sunny}, \text{rain}) = 0.375 \\ R(\text{overcast} \mid \text{play}) &= 0.20 \\ R(\text{rain} \mid \text{play}) &= 0.29 \end{aligned}$$

4.3. *Allocate training cases and assign parameters*

Let X be the test associated with X_i that produces the partition $\{\theta_1, \theta_2, \ldots, \theta_{n(i)}\}$, such that $\Delta'_X(X_i, T)$ is maximum. Then, X_i is the attribute in the current node. Now, we have to partition the current training set. As before, T is the current training set. T is split into $\{T_1, \ldots, T_{n(i)}\}$.

There are h classes (recall Section 3). Thus, every training case has an associated vector with $2h$ components: the h first components are probabilities (one for every class) and the h last components are risks (one for every class).

Then, in an intermediate step, we associate every training case with the corresponding T_j in the following manner:

- If the value of attribute X_i in the training case e is known, the vector associated with e has probability one and risk zero assigned to its class. The components of the other classes have associated probability zero and risk zero; when partitioning for the attribute X_i, e is included in the appropriate T_j and the probabilities are multiplied and the risks are added.
- If the value of attribute X_i in the training $e-$ is unknown, the vector associated with $e-$ has assigned for each class probability $p(X_i(e-) \in \theta_j \mid Y(e-) = y_k)$ and risk $R(\theta_j \mid y_k)\, \forall k$. When partitioning for the attribute X_i, the $e-$ training case is included in every T_j and the probabilities are multiplied and the risks are added.

The risk is interpreted as the confidence that we have in the prediction of the class (so zero risk means total confidence and high risk means low confidence).

Example 4. Let us consider Example 1 again. The value of attribute X_1 is unknown in this training case e_6. Let us assume that X_1 was the attribute selected for expansion and test X gave the finest partition of the values of the attribute. These values induce a partition in the current training set T into T_1, T_2, and T_3. The probabilities and risks for e_6 were computed in Example 3. Hence, the training case e_6 is assigned to T_1 with probability 0.25; to T_2 with probability 0.375; and to T_3 with probability 0.375. Hence, overcast or rain are more probable than sunny. The other 13 training cases with the value of X_1 known are assigned to

$$T_1 = \{e_1, e_2, e_3, e_4, e_5\}, \qquad T_2 = \{e_7, e_8, e_9\}, \qquad T_3 = \{e_{10}, e_{11}, e_{12}, e_{13}, e_{14}\}.$$

Every training case has an associated vector with four components (two probabilities and two risks, one for every class:"Play" or "Don't Play"). Thus, every T_j is as follows: (p_1, p_2, R_1, R_2).

In the current decision tree, each leaf has the associated vector $(N/E/R)$, where

- N is the number of training cases that reach the leaf (N is the sum of the probabilities of training cases being placed on the leaf).

- E is a k-dimensional vector where each component is the number of training cases belonging to each class (E_k is the sum of the probabilities of these training cases). The nominated class in the leaf is the class with the highest value in E_k.
- R is a k-dimensional vector of risk associated with each class.

Therefore, the partition of the training set is:
For T_1,

	$X_3(e)$	$Y(e)$	p_1	p_2	R_1	R_2
e_1	70	Play	1	0	0	0
e_2	90	Don't Play	0	1	0	0
e_3	85	Don't Play	0	1	0	0
e_4	95	Don't Play	0	1	0	0
e_5	70	Play	1	0	0	0
e_6	90	Play	0.25	0.6	0.375	0.26

sky = sunny
humidity ≤ 75 ($N = 2/E = (2; 0)/R = (0; 0)$)
humidity > 75 ($3.25/(0.25; 3)/(0.375; 0)$)
For T_2,

	$Y(e)$	p_1	p_2	R_1	R_2
e_6	Play	0.375	0	0.2	0.33
e_7	Play	1	0	0	0
e_8	Play	1	0	0	0
e_9	Play	1	0	0	0

sky = overcast ($3.375/(3.375; 0)/(0.2; 0)$)
For T_3,

	$X_4(e)$	$Y(e)$	p_1	p_2	R_1	R_2
e_6	yes	Play	0.375	0.4	0.29	0.4
e_{10}	yes	Don't Play	0	1	0	0
e_{11}	yes	Don't Play	0	1	0	0
e_{12}	no	Play	1	0	0	0
e_{13}	no	Play	1	0	0	0
e_{14}	no	Play	1	0	0	0

sky = rain
windy = yes ($2.375/(0.375; 2)/(0.29; 0)$)
windy = no ($3/(3; 0)/(0; 0)$)

The final Decision Tree is

```
outlook = sunny:
  humidity ≤ 75: Play (N = 2/E = (2; 0)/R = (0; 0))
  humidity > 75: Don't Play (3.25/(0.25; 3)/(0.375; 0))
outlook = overcast: Play (3.375/(3.375; 0)/(0.2; 0)
outlook = rain:
  windy = yes: Don't Play (2.375/(0.375; 2)/(0.29; 0)
  windy = no: Play (3/(3; 0)/(0; 0))
```

5. Prediction

The third aim is to predict the class of new observations. We explain how to predict the class of a b observation in Decision Trees constructed from training cases with missing attribute values. In addition, we illustrate how to predict the class of a $b-$ observation in Decision Trees constructed either from training sets with missing attribute values in the training cases or from training sets without unknown values.

5.1. How to predict the class of a new b observation in a decision tree constructed from e− training cases

Let us assume a new b observation with all known values. Then, b goes down the tree through a unique path. The b observation arrives at a leaf t. In this leaf t, each training case has an associated class $y_i, i = 1, \ldots, k$. We select the class that appears more times in the training cases; let us assume, for example, that it is y_ℓ. Thus, we predict that b belongs to y_ℓ. This prediction is completed with a confidence parameter and an error parameter. Class y_ℓ has an associated risk R_ℓ in the leaf t. The risk is interpreted as the error committed in the prediction of the b observation.

Therefore, we predict that b belongs to $Y(b) = y_\ell$ with probability p = (♯ training cases in t with associated class y_ℓ/♯ training cases in t) and risk R_ℓ.

Example 5. We are going to predict the class of the observation

$$b = (\text{sunny}, 70, 80, \text{no}).$$

We have $Y(b) =$ "Don't Play" with probability $3/3.25 = 0.92$ and risk zero.
For the case $b = (\text{rain}, 68, 81, \text{no})$, we have $Y(b) =$ "Play" with probability one and risk zero.
Another example is that, if $b = (\text{overcast}, 70, 65, \text{no})$, then $Y(b) =$ "Play" with probability one and risk 0.2.

5.2. How to predict the class of a new b− observation

The philosophy for the prediction of the class of the $b-$ observation is the same in decision trees constructed from training cases of type e or in decision trees constructed from training cases of type $e-$. The $b-$ observation goes down the tree. In a given moment, the $b-$ observation arrives at a node. We assume that the attribute related to this node is the missing value in $b-$. Then, $b-$ is directed down all the branches. This causes the $b-$ observation to appear in several leaves of the decision tree. Friedman's approach [36] assigns the $b-$ observation to the class with the largest representation in the union of these leaves. We follow [22].

In each node where the $b-$ observation goes down all the branches, we calculate the ratio of training cases directed to each branch from the totality of training cases in the node. (For example, if the node has s training cases and three branches with s_1, s_2, s_3 training cases, respectively, the proportion for each branch is $s_i/s \;\; i = 1, \ldots 3$, respectively.) The probabilities of each class appearing in the leaves are calculated. Finally, by computing the conditional probabilities, the probability distribution over $Y(b-)$ is calculated. We predict the class for $b-$ with maximum probability. In this case, the risks are not considered.

Example 6. We are going to predict the class of the observation

$$b- = (\text{sunny}, 70, ?, \text{no}).$$

$X_1(b-) =$ sunny and $X_3(b-) =?$ then

If humidity ≤ 75 then $Y(b-) =$ "Play" with probability one.
If humidity > 75, then $Y(b-) =$ "Don't Play" with probability $3/3.25$ and $Y(b-) =$ "Play" with probability $0.25/3.25$.
The training set was partitioned in 2 and 3.25 training cases for the values of $X_3 =$ humidity. Therefore
$Y(b-) =$ "Play" with probability $(2/5.25).1 + (3.25/5.25).(0.25/3.25) = 0.43$
$Y(b-) =$ "Don't Play" with probability $(3.25/5.25).(3/3.25) = 0.57$
We choose $Y(b-) =$ "Don't Play"

Another example is $b- = (?, 72, 90, \text{yes})$
If it is sunny, then $Y(b-) =$ "Play" with probability $0.25/3.25$ and $Y(b-) =$ "Don't Play" with probability $3/3.25$.
If it is overcast, then $Y(b-) =$ "Play" with probability one.
Then, $Y(b-) =$ "Play" with probability $0.375/2.375$ and $Y(b-) =$ "Don't Play" with probability $2/2.375$.

The partition of the 14 training cases was 5.25, 3.375 and 5.375 for the values of $X_1 =$ sky. Then:

$Y(b-) =$ "Play" with probability
$(5.25/14).(0.25/3.25) + (3.375/14).1 + (5.375/14).(0.375/2.375) = 0.33$
$Y(b-) =$ "Don't Play" with probability
$(5.25/14).(3/3.25) + (3.375/14).0 + (5.375/14).(2/2.375) = 0.67$
Therefore, we choose $Y(b-) =$ "Don't Play"

Table 6
Datasets

Prob	Size	Missing	Ex.maj.class	Continuous	Nom	No. class
BC	286	9	70.3	0	9	2
CH	3196	0	52.2	0	36	2
GL	214	0	35.3	9	0	7
G2	163	0	53.4	9	0	2
HD	303	7	54.5	5	8	2
HE	155	167	79.4	6	13	2
HO	368	1927	63.0	7	15	2
HY	3163	5329	95.2	7	18	2
IR	150	0	33.3	4	0	3
LA	57	326	64.9	8	8	2
LY	148	0	56.7	2	16	4
MU	8124	2480	51.8	0	22	2
SE	3163	5329	90.7	7	18	2
SO	47	0	36.2	0	35	4
VO	435	381	61.4	0	16	2
V1	435	392	61.4	0	16	2

6. Experimental results

We have implemented our algorithm and used standard datasets to obtain experimental results. We have collected the usual and known datasets from Holte [37]. The 16 problems that Holte used to compare his 1R method with the C4.5 method of Quinlan [22] were used in our work. This set of problems aims to represent all real-life problems. Holte claimed that the experimental results obtained for these problems by learning systems could shed light on the performance of each learning system in a real situation. All the selected problems comply with two conditions: representing a real-life problem that has not been constructed artificially; and the examples are described by means of attributes used naturally in real life. The only dataset that complies with the second condition is CH. This dataset represents the endgames in chess. According to Holte, this dataset is designed to fit well in Quinlan's ID3.

As mentioned in the introduction, the supervised imputation method based on C4.5 has the best overall performance, as indicated in [23] with several examples. This is the main reason that we compare our method with C4.5. We also use the same database as in [23], in addition to some other databases not considered in that work.

6.1. Description of datasets

There are several versions of Holte's problems. Basically, there are changes in the number of training cases or attributes used. Here we have used the version stored in the UCI Repository of Automated Learning Repository in California University, Irvine [38].

In Table 6, we present a brief summary of the characteristics of these datasets: the first column shows the code of the dataset, the second shows the number of training cases, the third includes the number of missing values, the fourth shows the percentage of occurrence of the most frequent value of the class, the fifth and sixth show the number of continuous and discrete attributes, respectively, and the last column shows the number of different values for the class.

For the experiment, two algorithms were implemented to build decision trees. In the first algorithm, the training cases with missing values are distributed following the approach of C4.5; that is, the probability distribution function of the attribute under consideration is used to distribute the cases with missing values in this attribute. In the second algorithm (our method), the cases with missing values are distributed taking into account the probability distribution obtained using the attribute and the class.

Our method is associated with the total percentage of missing attribute values in the training cases, and these missing values are distributed in a random way among the attributes. The method works independently of the number of attributes with missing values, that is, missing values can appear in any attribute. We have checked that the proposed model does not fail when we increase the number of attributes with missing values and the total percentage of missing values holds.

Table 7
Results

Code		Original C4.5–PostP	10% C4.5–PostP	20% C4.5–PostP	30% C4.5–PostP
BC	(error)	12.24–11.54	9.16–8.64	12.97–11.50	15.87–13.95
	(std d)	0.0–0.0	1.17–1.14	0.86–1.22	1.33–1.37
CH	(error)	0.38–0.38	3.00–2.84	6.74–5.45	11.92–9.27
	(std d)	0.0–0.0	0.18–0.17	0.31–0.25	0.40–.023
GL	(error)	6.07–6.07	11.78–10.14	17.24–13.55	28.08–18.64
	(std d)	0.0–0.0	2.70–2.29	2.12–1.77	1.54–2.90
G2	(error)	7.97–7.97	11.41–10.61	15.77–12.82	18.83–14.48
	(std d)	0.0–0.0	1.68–1.45	2.69–2.15	2.26–2.07
HD	(error)	4.29–3.63	7.03–6.53	10.33–8.15	12.61–10.66
	(std d)	0.0–0.0	0.77–1.18	1.84–1.43	1.40–1.63
HE	(error)	1.94–2.58	6.84–4.84	7.48–5.55	10.32–7.55
	(std d)	0.0–0.0	1.79–0.88	0.76–1.09	1.09–1.52
HO	(error)	9.24–5.98	10.22–7.55	11.85–9.27	13.64–12.04
	(std d)	0.0–0.0	1.29–0.65	1.00–1.01	0.92–0.97
HY	(error)	0.35–0.35	0.74–0.65	1.13–0.94	1.62–1.19
	(std d)	0.0–0.0	0.08–0.06	0.12–0.11	0.16–0.11
IR	(error)	2.67–2.67	3.80–3.67	4.80–4.00	9.27–6.00
	(std d)	0.0–0.0	0.73–0.77	1.52–0.98	1.48–1.69
LA	(error)	3.51–1.75	9.65–4.03	21.93–3.86	24.21–7.72
	(std d)	0.0–0.0	8.14–2.14	7.03–0.96	6.66–2.51
LY	(error)	6.08–6.08	4.86–4.32	6.82–4.86	10.74–6.62
	(std d)	0.0–0.0	1.17–1.14	1.78–1.14	2.05–1.59
MU	(error)	0.0–0.0	0.17–0.21	0.65–0.40	1.55–0.78
	(std d)	0.0–0.0	0.05–0.02	0.06–0.05	0.12–0.05
SE	(error)	1.01–1.01	2.24–2.10	3.64–3.32	4.75–4.39
	(std d)	0.0–0.0	0.23–0.21	0.22–0.26	0.30–0.34
SO	(error)	0.0–0.0	2.55–1.06	5.53–0.64	10.0–1.91
	(std d)	0.0–0.0	1.43–1.35	3.14–0.94	3.36–1.21
VO	(error)	1.84–2.30	2.62–2.62	3.77–3.22	5.70–4.18
	(std d)	0.0–0.0	0.46–0.51	0.46–0.48	0.48–0.45
V1	(error)	4.60–4.37	4.71–4.07	6.23–5.03	7.56–5.77
	(std d)	0.0–0.0	0.65–0.46	0.57–0.28	0.85–0.76

During the first stage, the original datasets were used; subsequently, we randomly introduced a percentage of missing values in the original training cases (10%, 20%, and 30%, respectively). This process of introducing missing values was done again ten times for each percentage. In this way, the possibility of bias is minimized. Thus, 31 experiments were performed for each dataset. The classification error has been calculated for the 31 estimated decision trees. For each group of decision trees belonging to the same percentage of missing values, mean errors and standard deviations were calculated.

6.2. Results

The results obtained are shown in Table 7. The first column shows the dataset code, the second the error in the sample using C4.5 (first value) and the error using our approach PostP (second value). In columns 3, 4, and 5, the results obtained for the dataset with 10%, 20%, and 30% of missing values, respectively, are shown. For each percentage, the mean error of the experiments is shown in the first row of each dataset, and the standard deviation of this error appears in the second row.

There are no essential differences between decision trees generated with complete data and those with missing values; only one more branch could appear in each level but, in general, the size of the trees in both methods is similar.

Table 8
Other datasets

Prob	Size	Missing	Continuous	Nom	No. Class
Adult	3000	759	6	8	2
Bands	400	986	20	17	2
Dermatology	250	1	34	2	6
Horse colic	250	67	10	17	2
House-votes	315	395	0	17	2
Post-operative	70	3	0	9	3

Table 9
More results

Code		Original C4.5–PostP	10% C4.5–PostP	20% C4.5–PostP	30% C4.5–PostP
Adult	(error)	7.00–7.23	9.25–8.72	11.43–10.33	13.88–12.57
	(std d)	0.00–0.00	0.37–0.28	0.62–0.39	0.32–0.36
Bands	(error)	4.25–4.25	5.95–5.10	10.42–8.40	15.47–12.22
	(std d)	0.00–0.00	0.63–0.63	0.97–1.24	0.77–0.82
Dermatology	(error)	2.00–2.00	2.68–2.64	4.96–4.32	9.00–5.36
	(std d)	0.00–0.00	0.57–1.35	0.76–1.08	1.22–1.31
Horse colic	(error)	6.05–6.85	13.87–8.31	18.47–12.06	23.27–15.93
	(std d)	0.00–0.00	2.04–1.33	2.89–1.84	3.78–1.15
House-votes	(error)	1.90–1.90	2.29–1.94	3.56–2.57	4.70–3.27
	(std d)	0.00–0.00	0.55–0.31	0.59–0.25	0.68–0.63
Post-operative	(error)	14.29–14.29	15.29–15.57	18.86–17.57	21.43–20.00
	(std d)	0.00–0.00	2.05–2.86	1.32–1.65	2.41–2.41

If the rate of missing values is large (30%), the trees from PostP are a little larger than the trees from C4.5; this result is reasonable, because the PostP method distributes the training cases using the attribute values and classes. However, in the case of large trees obtained with PostP, the error is substantially less. The mean error for all experiments with 30% of missing values with C4.5 is 11.67, and with PostP it is 7.82. Thus, our method is 33% better than the C4.5.

We can conclude that the results are appreciably better in most cases when the PostP algorithm is used. For several datasets, the error decreases to 80% (for example, with the dataset SO, the mean error with C4.5 is 4.52 and with our method it is 0.90).

We note that the experiments with the dataset LA give very good results: in the experiments with 30% missing values, the error is reduced by more than 69%. The best result with the PostP algorithm, compared to C4.5, is obtained with the dataset SO.

To sum up, C4.5 obtains better results only in 6% of all experiments. In 3% of the cases, the results are similar for both methods, and in 82% our method improves the classification; the remaining 15% belong to the experiments without missing values.

6.3. *Other datasets*

We have carried out some experiments with other datasets (see Table 8) such as standard datasets obtained from the MLRepository [38]. Datasets with original missing attribute values were chosen. The table is a brief summary of their characteristics.

Similar results to the previous datasets are obtained (see Table 9). In all cases with missing values, the error with PostP is less than the error with C4.5. In addition, with a higher percentage of missing values, the error with PostP decreases faster than in C4.5. Furthermore, the standard deviation with our method is less than with C4.5.

7. Conclusions and future work

A new approach to working with missing attribute values has been presented. All stages of the TDIDT algorithms where the attribute value is necessary have been adapted to work with missing attribute values. The concept of the splitting criterion is formally defined and adapted to work with missing attribute values. We have developed a new approach to tackle the problem of assigning values to unknown values of an attribute from a Decision Theory point of view. This approach takes into account more information (attribute values and class values) than C4.5 and other analogous algorithms. This is the philosophy that underlies Bayes' Theorem. The attribute values are strongly related to the value class of a training case.

Every training case with a missing attribute value has two associated parameters: confidence and error.

When we apply Decision Trees to Data Mining, the ultimate aim is to make good predictions of the class of new observations. In this paper, we have developed a way to predict the class for new b and $b-$ observations in all Decision Tree variants. Therefore, we have presented a way to work with missing attribute values that includes statistical concepts in inductive learning algorithms.

Future work includes the analysis of the missing treatment methods in other data sets. In this work, missing values were inserted completely at random (MCAR), whereas in forthcoming work we will analyze the behavior of this method when missing values are not randomly distributed. In this case, there is a possibility of creating invalid knowledge. For an effective analysis, we will have to study not only the error rate, but also the quality of the knowledge induced by the learning system.

Other future research based on the inductive learning approach will take into account at least three aspects: risk, pruning, and boosting. Risk closely focuses on improving prediction (as an independent parameter or as a function with other parameters). We will investigate pruning to simplify the trees, using, in particular, the CIDIM method [31,32]. This method carries out pruning and looks at the best pair (attribute, splitting) from the point of view of information gain. Finally, we propose adapting other methods to obtain decision trees (by means of voting [39] and sampling [40]) for our case: missing attribute values.

It is worth mentioning that future work, from the statistical point of view of this approach, focuses on generalization taking into account two or more attributes. These attributes will be chosen according to their inter-relationships with the class value. These inter-relationships can be detected by linear regression or high correlation between the attributes. If the probability distribution of the attribute with missing attribute values and that of the class are known, following Little and Rubin [1], a further study of the attribute values can be made.

Acknowledgments

The authors would like to thank the anonymous referees whose comments greatly improved the readability of the paper.

References

[1] R.J.A. Little, D.B. Rubin, Statistical Analysis with Missing Data, John Wiley & Sons Inc, 1987.

[2] T.M. Mitchell, Machine Learning, MacGraw-Hill, 1997.

[3] M. Kryszkiewicz, Rough set approach to incomplete information systems, Inform. Sci. 112 (1998) 39–49.

[4] M. Kryszkiewicz, Rules in incomplete information systems, Inform. Sci. 113 (1999) 271–292.

[5] M. Kryszkiewicz, Rough set approach to rules generation from incomplete information systems, Encyclopedia Comput. Sci. Technol. 44 (2001) 319–346.

[6] J.W. Grzymala-Busse, Rough set strategies to data with missing attribute values, in: Proceedings of the Workshop on Foundations and New Directions in Data Mining, associated with the third IEEE International Conference on Data Mining, 19–22 November, Melbourne, FL, USA, 2003, pp. 56–63.

[7] J.W. Grzymala-Busse, S. Siddhaye, Rough set approaches to rule induction from incomplete data, in: Proceedings of the IPMU'2004, the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 4–9 July, Perugia, Italy, vol. 2, 2004, pp. 923–930.

[8] J.W. Grzymala-Busse, Three approaches to missing attribute values. A rough set perspective, in: Workshop on Foundations of Data Mining, associated with the fourth IEEE International Conference on Data Mining, 1–4 November, Brighton, UK, 2004.

[9] J.W. Grzymala-Busse, M. Hu, A comparison of several approaches to missing values in data mining, in: W. Ziarko, Y.Y. Yao (Eds.), Rough Sets and Current Trends in Computing, in: Lecture Notes in Computer Science, vol. 2005, Springer, 2001.

[10] M. Hu, S.M. Salvucci, M.P. Cohen, Evaluation of some popular imputation algorithms, in: Section on Survey Research Methods, American Statistical Association, 2000.

[11] J. Scheffer, Dealing with missing data, Res. Lett. Inf. Math. Sci. 3 (2002) 153–160.

[12] J.L. Schafer, Analysis of Incomplete Multivariate Data, Chapman & Hall, London, 1997.

[13] W. Liu, A. White, S. Thompson, M. Bramer, Techniques for dealing with missing values in classification, in: International Symposium on Intelligent Data Analysis, 1997.

[14] J. Quinlan, Unknown attribute values in induction, in: Proceedings of the Sixth International Machine Learning Workshop, Morgan Kaufmann, San Mateo, CA, 1989, pp. 164–168.

[15] E.B. Hunt, J. Marin, P.J. Stone, Experiments in Induction, Academic Press, New York, 1966.

[16] J. Quinlan, Discovering rules by induction from large collections of examples, in: D. Michie (Ed.), Expert Systems in the Micro Electronic Age, 1979.

[17] J. Quinlan, Learning efficient classification procedures, in: R.S. Michlaski, J.G. Carbonell, T.M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, Tioga Press, Palo Alto, CA, 1983.

[18] J. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.

[19] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.

[20] J. Quinlan, The effect of noise on concept learning, in: R.S. Michlaski, J.G. Carbonell, T.M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, vol. 2, Morgan Kaufmann, San Mateo, CA, 1986.

[21] B. Cestnik, I. Kononenko, ASSISTANT 86: A knowledge-elicitation tool for sophisticated users, in: I. Bratko, N. Lavrac (Eds.), Progress in Machine Learning, Sigma Press, Wilmslow, UK, 1987.

[22] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Gatos, CA, 1992.

[23] A. Farhangfar, L. Kurgan, W. Pedrycz, Experimental analysis of methods for imputation of missing values in databases, in: L. Priddy Kevin (Ed.), Intelligent Computing: Theory and Applications II, in: Proceedings of the SPIE, vol. 5421, 2004, pp. 172–182.

[24] K.J. Cios, L.A. Kurgan, Hybrid inductive machine learning: An overview of CLIP algorithms, in: L-C- Jain, J. Kacprzyk (Eds.), New Learning Paradigms in Soft Computing, Physica-Verlag, Springer, 2001, pp. 276–322.

[25] K.J. Cios, L.A. Kurgan, CLIP4: Hybrid inductive machine learning algorithm that generates inequality rules, Inform. Sci. 163 (1–3) (2004) 37–83.

[26] R.O. Duda, P.E. Hart, Pattern Classification and Scene Analysis, John Wiley, 1977.

[27] C.E. Shannon, A Mathematical theory of communication, Bell Syst. Tech. J. 27 (1948) 379–423.

[28] Z. Zheng, B.T. Low, Classifying unseen cases with many missing values, in: N. Zhong, L. Zhou (Eds.), PAKDD 99, in: LNAI, 1574, 1999, pp. 370–375.

[29] I. Kononenko, I. Bratko, E. Rokar, Experiments in automatic learning of medical diagnostic rules, ISSEK Workshop, Bled, 1984.

[30] I. Fortes, R. Morales-Bueno, Ll. Mora, F. Triguero, A decision theory approach to work with missing attribute values in inductive learning algorithms, in: Proc. of COMPSTAT2000 (14th Conference of the International Association for Statistical Computing), Utrecht, 2000.

[31] G. Ramos, R. Morales, Formalizacion de los Algoritmos TDIDT y CIDIM, Techn. Report LCC-ITI 99/01, Dept. Computer Science, Malaga University, 1999.

[32] G. Ramos, R. Morales, A. Villalba, CIDIM. Control of Induction of Sample Division Method), Una mejora de los algoritmos TDIDT, Techn. Report LCC-ITI 97/08, Dept. Computer Science, Malaga University, 2000.

[33] J.O. Berger, Statistical Decision Theory and Bayesian Analysis, Springer Verlag, New York, 1988.

[34] L. Hyalf, R.L. Rivest, Constructing optimal binary decision trees is NP-complete, Inf. Process. Lett. 5 (1) (1976) 15–17.

[35] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, J. Artificial Intelligence Res. 6 (1) (1997) 1–34.

[36] J.H. Friedman, A recursive partitioning decision rule for non-parametric classification, IEEE Trans. Comput. (1977) 404–408.

[37] R.C. Holte, Very simple classification rules perform well on most commonly used datasets, Mach. Learn. 11 (1993) 63–91.

[38] C. Blake, E. Keogh, C.J. Merz, UCI repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences, http://www.ics.uci.edu/mlearn/MLRepository.html, 1998.

[39] G. Ramos, Nuevos Desarrollos en Aprendizaje Inductivo, Tesis Doctoral, Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, 2001.

[40] G. Ramos, R. Morales, A new method for induction decision trees by sampling, Neurocolt Workshop on Applications of Learning Theory Bellaterra, Barcelona, 2000.