Treatment of Missing Values for Association Rules

Arnaud Ragel and Bruno Crémilleux

ragel@info.unicaen.fr – cremilleux@info.unicaen.fr
Groupe de Recherche En Informatique Image et Instrumentation de Caen
CNRS UPRES-A 6072
Université de Caen - Esplanade de la paix - F 14032 CAEN CEDEX

Abstract. Agrawal *et al.* [2] have proposed a fast algorithm to explore very large databases with association rules [1]. In many real world applications missing values are often inevitable and we will show, in this case, that association rules give bad results. We propose an approach to increase their resistance against missing values. The main idea is to cut a database in several valid databases (vdb) for a rule, a vdb must not have any missing values. We redefine support and confidence of rules for vdb. These definitions are fully compatible with [2] which is the core of all known algorithms. Simulations show that this approach outperforms the classic approach by factors five to ten.
keywords: Association rules, Missing values, Knowledge Discovery.

1 Introduction

Data mining (knowledge discovery in databases) is a field of increasing interest combining databases, artificial intelligence and machine learning. The purpose of data mining is to facilitate understanding large amounts of data by discovering regularities or exceptions.

Discovery of association rules [1] is an interesting subfield of data mining. Historically, the motivation for searching association rules has come from the desire to analyze large amounts of supermarket basket data. For instance, an association rule $beer \longrightarrow chips(87)\%$ states that 87% of customers that have bought beer, also have bought chips. More generally, given a transaction database, where each data is a set of literals (called items), an association rule is an expression of the form $X \longrightarrow Y$, where X and Y are sets of items. The intuitive meaning of such a rule is that data of the database which contains X tends to contain Y. A rule $X \longrightarrow Y$ is evaluated by a support and a confidence. Its support is the percentage of data that contains both X and Y. Its confidence is the percentage of data that contains Y among those containing X. The problem of mining association rule is to find all rules that satisfy a user-specified minimum support and minimum confidence. Since Agrawal *et al.* [1], association rules have been the focus of considerable work on developing fast algorithms, e.g [2] [16] [15] [10] . At the present time, associations rules are used to explore millions of data [4].

Missing values have not been considered of interest since [1] [2] because association rules have been first developed to explore transaction databases where the problem of missing attribute values does not practically exist. However this problem becomes important if we try to find associations between values of different attributes in relational tables where missing values are often inevitable. Consequently all the algorithms derivated from [1] are ineffective in this context : very few rules are discovered in databases containing missing values (in Sect. 5.1 we will show the slump of the results). In this paper present an extension for [2], which is the core of all known algorithms, to mine inter-attribute association rules with missing attribute values. This approach is inspired by one of the simplest way to treat missing values in machine learning methods. It consists in evaluating a rule only with known values, i.e ignoring missing values. Even though this approach causes a huge waste of data for a lot of methods, e.g decision trees [9] [14], it is particularly well suited for association rules: a cancelled data for a rule, i.e data containing missing values for values tested by the rule, can be used by others rules. Then, even if the whole database is not used to evaluate a rule, the whole database is used to discover the ruleset. So a database is divided in several valid databases, where for a rule, a vdb must not have any missing values. Support and confidence must have been redefined to be evaluated in the different valid databases.

The remaining part of the paper is organized as follows. Section 2 briefly reviews the definition of association rules and the core of efficient algorithms. In Sect. 3, we will begin with a short presentation of the missing values and next we will set out problems posed by missing values for association rules. In Sect. 4 we will present our solution. In Sect. 5, we will present a simulation where we introduce, artificially, missing values to show the slump of the results with the usual approach and the robustness of the new one. Results on real world experiment (medical databases on Hodgkin disease) are presented too. Finally, we will conclude in Sect. 6 where we show how our extension can be useful to fill missing values for machine learning methods like decision trees.

2 Association Rules

We present a definition of association rules for relational databases to focus on missing values. Actually, association rules can be used on more general datasets (see [1]).

2.1 General Definitions

In databases, data are described by a same set of attributes, where each of them have a value which is chosen in a set of possible values, specific to each attribute. For example, in Fig. 1 (Section 3.2) the database has 4 attributes (X1, X2, X3, X4) and the possible values are a,b for X1, a,b,c for X2, a,b,c,d,e,f,g,h for X3 and c,d for X4.
Let B be a database with $R = \{I_1, I_2, ..., I_m\}$, the set of all associations attribute

values, called items. A data t is composed by a subset of R. For the precedent example, the set of all the items is {X1=a, X1=b, X2=a, X2=b, X2=c, X3=a, ..., X3=h, X4=c, X4=d}

An association rule is an implication of the form $X \longrightarrow Y$ with $X \subset R$, $Y \subset R$ and $X \cap Y = \varnothing$. A rule is evaluated by a support and a confidence where it has a support s in the database if s% of data contain XY and a confidence c if c% of data that contain X also contain Y.

Definition 1 (Support). The support of an itemset $X \subseteq R$ in a database B is:

$$Support(X) = \frac{\mid \{t \in B \mid X \subseteq t\} \mid}{\mid B \mid} \tag{1}$$

The support of a rule $X \longrightarrow Y$ in a database B is:

$$Support(X \longrightarrow Y) = Support(XY) \tag{2}$$

Definition 2 (Confidence). The confidence of a rule $X \longrightarrow Y$ in a database B is:

$$Confidence(X \longrightarrow Y) = \frac{Support(XY)}{Support(X)} \tag{3}$$

Remark 3. In the remaining of the paper we note, for an itemset $X \subseteq R$, B_X the subset of a database B containing X, i.e $B_X = \{t \in B \mid X \subseteq t\}$

The problem of mining association rules is, given a database B, to generate all association rules that have support and confidence greater than user-specified minimum support and minimum confidence respectively.
A confidence threshold is used to exclude rules that are not strong enough to be interesting. A support threshold is also given to remove rules that do not apply often enough.

The problem of mining association rules can be decomposed into two subproblems [1]:

- Find all combinations of items that have data support above minimum support. Call those combinations frequent itemsets.
- Use the frequent itemsets to generate the desired rules. The general idea is that if ABCD and ABD are frequent itemsets, then we can determine if a rule $ABD \longrightarrow C$ holds by computing the ratio r=support(ABCD)/support(ABD). The rule holds only if r is superior to minimum confidence.

The first step is responsible for most of the computation time (the second step just consists in a division of support between frequent itemsets).

An efficient breadth-first, or level-wise method for generating candidate sets, i.e potentially frequent itemsets, has been presented in [2] [3]. This method, called Apriori, is the core of all known algorithms.

2.2 Breadth-first Algorithms

The main idea is that if a set is not frequent, i.e with support lower than the user specified support, then its supersets can not be frequent. The algorithm starts on level 1 by evaluating the support of singleton itemsets. On level k, candidate itemsets X of size k are generated such that all subsets of X are frequent. After the support of the candidates on level k have been evaluated, new candidates for level k+1 are generated and evaluated. This is repeated until no new candidates can be generated. The efficiency of this approach is based on not generating and evaluating those candidate itemsets that cannot be frequent, given all smaller frequent itemsets. Further more, the whole database is not used for evaluating the support of candidate itemsets after the first pass because each itemset matches a subset of data: in later passes, the size of the subset can become much smaller than the database, thus saving much reading effort.

As an example, let L3 be {{1 2 3}, {1 2 4}, {1 3 4}, {1 3 5}, {2 3 4}}[1], the frequent itemsets of size 3. Only {1 2 3 4} is a possible itemset of length 4 (in {1 3 4 5}, the itemset {1 4 5} is not frequent, i.e is not in L3). The data matched by this itemset are the intersection between the data matched by {1 2 3} and {1 2 4}.

In the next section the problem posed by the missing values is studied.

3 Association Rules and Data with Missing Values

3.1 Data and Missing Values

The problem of the missing values has been widely studied in the literature [9] [12] [6]. A missing value hides a value of an attribute. There are various way in which missing values might occur. For example:

- values recorded are missing because they were too small or too large to be measured.
- values recorded are missing because they have been forgotten or they have been lost (input error).

In the first case the missing values are structured, i.e they hide a same category of values for an attribute. In the previous example they only affect large and small values. In the second case values are missing at random.

Now, let's show the weakness of the usual treatment of association rules when data contain missing values.

3.2 Usual Treatments

Association rules have been first developed to explore transaction databases where the problem of missing attribute values does not practically exist. This

[1] each number represent an attribute-value association

problem becomes important if we try to find associations between values of different attributes in relational tables. To our knowledge, no studies and no applications, to deal efficiently with missing values, have been made since. Consequently and actually, missing values are not correctly treated. In effect missing values are treated like a new possible value for each attribute. This is equivalent to the method presented in [11] for missing values in decision trees which is only appropriate when the missing values are informative, e.g values recorded are missing because they were too small or too large to be measured [7] [8].

Unfortunately, values are usually missing at random. In this case the value *?* does not have the same status as a proper attribute value because, for a same attribute, *?* is not equivalent between data, i.e the *?* hides several different values. The result of such a treatment, in this case, is very bad.
Example:

Imagine an attribute A with two possible values,a_1 and a_2, with $t_{a_1}\%$ for a_1 and $t_{a_2}\%$ a_2 ($t_{a_1} + t_{a_2} = 100\%$) . The item $A = a_1$ (resp.$A = a_2$) will have a support of $t_{a_1}\%$ (resp. $t_{a_2}\%$). If missing values are introduced with $vm_{a_1}\%$ for the value a_1 and $vm_{a_2}\%$ for the value a_2, the $A = a_1$ (resp.$A = a_2$) will have a support of $t_{a_1} - vm_{a_1}\%$ (resp. $t_{a_2} - vm_{a_2}\%$). Further more the item $A =?$ will have a support $vm_{a_1} + vm_{a_2}\%$.

This solution reduces support and confidence values. With the combination of items in an itemset, supports are reduced a lot and, so, can be under the user specified support. The result is a modification of the rule evaluations (support and confidence) or a loss of rules. In Sect. 5.1 a simulation with several database shows that with 15% of missing values, more than 90% of the rules are lost.

We give in Fig. 1 a very simple example which puts the problem to the fore quite well. We give two similar databases with 4 attributes: X1, X2, X3 and X4. In the second table we have introduced artificial random missing values.

Id	X1	X2	X3	X4
1	a	a	a	c
2	a	a	b	c
3	a	b	c	c
4	a	b	d	c
5	b	b	e	d
6	b	b	f	d
7	b	c	g	d
8	b	c	h	d

Database 1 (DB1)

Id	X1	X2	X3	X4
1	?	a	a	c
2	a	a	b	?
3	a	b	c	c
4	a	b	d	c
5	?	b	e	d
6	b	b	f	?
7	b	c	g	d
8	b	c	h	d

Database 2 (DB2)

Fig. 1. Same database, without and with missing values

If a minimum support of 40% and a minimum confidence of 80% are chosen we have, from DB1:
Itemsets with a support of 50%:

length 1: {X1=a}, {X1=b}, {X2=b}, {X4=c}, {X4=d}
length 2 (constructed from itemsets of length 1): {X1=a, X4=c}, {X1=b, X4=c}

As a result we have 4 rules:
If X1=a $\longrightarrow$ X4=c, support=50, confidence=100
If X1=b $\longrightarrow$ X4=d, support=50, confidence=100
If X4=c $\longrightarrow$ X1=a, support=50, confidence=100
If X4=d $\longrightarrow$ X1=b, support=50, confidence=100

In the second database, containing missing values, all the rules are lost because supports of the itemsets are under the minimum support of 40% (except for X2=b). If we look for the values of the precedent rules in DB2 we found a support of 25% and a confidence of 66%. If we choose a new minimum support of 25% and a new confidence support of 66% we retrieve the precedent rules but with a lot of new rules:
If X1=a $\longrightarrow$ X4=c, support=25, confidence=66,7
If X1=b $\longrightarrow$ X4=d, support=25, confidence=66,7
If X4=c $\longrightarrow$ X1=a, support=25, confidence=66,7
If X4=d $\longrightarrow$ X1=b, support=25, confidence=66,7

Here are the new rules:
If X2=c $\longrightarrow$ X1=b, support=25, confidence=100
If X2=c et X4=d $\longrightarrow$ X1=b, support=25, confidence=100
If X4=c $\longrightarrow$ X2=b, support=25, confidence=66,7
If X1=a $\longrightarrow$ X2=b, support=25, confidence=66,7
If X1=a et X4=c $\longrightarrow$ X2=b, support=25, confidence=100
If X4=d $\longrightarrow$ X2=c, support=25, confidence=66,7
If X1=b $\longrightarrow$ X2=c, support=25, confidence=66,7
If X1=b et X4=d $\longrightarrow$ X2=c, support=25, confidence=100
If X2=b et X4=c $\longrightarrow$ X1=a, support=25, confidence=100
If X2=c $\longrightarrow$ X1=b, support=25, confidence=100
If X1=a et X2=b $\longrightarrow$ X4=c, support=25, confidence=100
If X2=c $\longrightarrow$ X4=d, support=25, confidence=100
If X1=b et X2=c $\longrightarrow$ X4=d, support=25, confidence=100

To sum up, to discover useful rules in a database containing missing values, support and confidence could be reduced. It is a first problem because it is a hard problem to fix these minimum values. A second problem is that with these new thresholds, the useful rules are under evaluated and they are hidden by a lot of new inconsistent rules. In this example we know that only the four first rules are useful (because we have DB1 without missing values) but in real situations it is impossible to make a difference between useful and inconsistent rules when useful rules are under evaluated by the presence of missing values. Consequently association rules are not of interest in domains with missing values.

In the next section, we present our new approach where we avoid these precedent problems by partially ignoring missing values. With this new approach the rules are not under evaluated by the presence of missing values. Consequently rules can be used just as if there were not any missing values.

4 New Approach to Deal with Missing Values

As we do not know what random missing values mean (a *?* hides, for a same attribute, different values for each data) we will ignore them. In fact only certain data containing missing values will be partially disabled.

4.1 Definitions

Let B be a database, X an itemset, t a data and R a rule $X \longrightarrow Y$.

Definition 4 (Disabled data). t is disabled for X in B, if t contains missing values for at least one item i of X.

Definition 5. We note $Dis(X)$ the subset of B disabled for X.

Definition 6. We note B_X the subset of B containing X, i.e $B_X = \{t \in B \mid X \subseteq t\}$

Definition 7 (Valid database). The valid database (vdb) for an itemset X in B is: $vdb(X) = B - Dis(X)$
The valid database (vdb) for a rule R in B is: $vdb(R) = B - Dis(XY)$

In case of random missing values, vdb(X) is a sample without missing values of B. Consequently we can calculate the support of an itemset in its vdb and retrieved the support as if the database did not contain missing values. With this approach the whole database is not used to evaluate an itemset but the whole database is used to evaluate all itemsets because each itemset has a different vdb.

Example of disabled data:
Data 1: X1=a, X2=?, X3=a.
Data 1 is disabled for the itemset {X1=a & X2=c} because we do not know if the *?* hide a *c* or another value. If the real value is *c* the support must be increased but if it is another value the support must be decreased. As we can not conclude, the data is disabled. On the contrary data 1 is not disabled for the itemset {X1=a & X3=a}.
Data 1 is disabled for an itemset like {X1=b & X2=c} even if, whatever the value hidden by *?*, this itemset can not verify it. In this way, a vdb for an itemset Z, can be a good sample.

The main problem is now to calculate efficiently support and confidence in vdb for rules. In the next section we give new support and confidence definitions, fully compatible with [2], and we introduce a new parameter called *representativity*.

4.2 Support, Confidence and Representativity in Valid Databases

Notation: $\mid B \mid = Card(B)$

Support: We redefine support to calculate it ignoring disabled data.

$$Support(X) = \frac{\mid B_X \mid}{\mid B \mid} \quad \text{(usual definition)} \tag{4}$$

$$Support(X) = \frac{\mid B_X \mid}{\mid vdb(X) \mid} = \frac{\mid B_X \mid}{\mid B \mid - \mid Dis(X) \mid} \quad \text{(new definition)} \tag{5}$$

Figure 2 gives results from databases of the Fig. 1 for the same parameters of the previous example (minimum support of 40% and minimum confidence of 80%). We retrieved the results of the Sect. 3.2. No itemsets are erroneously discovered and they are not under evaluated.

Itemsets	data disabled	support
X1=a	1, 5	3/6=50%
X1=b	1, 5	3/6=50%
X2=b		4/5=50%
X4=c	2, 6	3/6=50%
X4=d	2, 6	3/6=50%
X1=a, X4=c	1, 2, 5, 6	2/4=50%
X1=b, X4=d	3, 4, 7, 8	2/4=50%

Fig. 2. Correction of supports for missing values

Confidence: A rule $X \longrightarrow Y$ holds with confidence of c if c% of data that contain X also contain Y.

$$Confidence(X \longrightarrow Y) = \frac{\mid B_{XY} \mid}{\mid B_X \mid} \quad \text{(usual definition)} \tag{6}$$

Confidence is quick and easy to calculate because $\mid B_X \mid$ and $\mid B_{XY} \mid$ have already been calculated in $Support(X)$ and $Support(XY)$. In effect:

$$\frac{Support(XY)}{Support(X)} = \frac{\frac{|B_{XY}|}{|B|}}{\frac{|B_X|}{|B|}} = \frac{\mid B_{XY} \mid}{\mid B_X \mid} \tag{7}$$

With the new approach it is not so easy. In a rule $X \longrightarrow Y$, a data D, can be not disabled for X and disabled for XY, i.e B_X can contain disabled data for XY. Then the ratio $\frac{|B_{XY}|}{|B_X|}$ is falsified by missing values of Y. We must recalculate the support of X in the valid database of XY. In fact we can use previous calculus:

$$Confidence(X \longrightarrow Y) = \frac{\mid B_{XY} \mid}{\mid B_X \mid - \mid Dis(Y) \cap B_X \mid} \quad \text{(new definition)} \tag{8}$$

$\mid B_{XY} \mid$ has already been calculated in $Support(XY)$,
$\mid B_X \mid$ has already been calculated in $Support(X)$,
$\mid Dis(Y) \cap B_X \mid$ requires to scan Dis(Y), which has already been constructed during the calculation of Support(Y), and to keep those which contain X.

Figure 3 shows the result for the database of the Fig. 1, containing missing values, according to a minimum confidence of 80%. With our new definitions, the presence of missing values is well suited. Exactly the same ruleset is retrieved where as with the usual approach all the rules were lost.

rule $X \longrightarrow Y$	data disabled for X	data disabled for Y	support	confidence
$X1 = a \longrightarrow X4 = c$	1,5	2,6	2/4=50	2/(4-2)=100
$X1 = b \longrightarrow X4 = d$	1,5	2,6	2/4=50	2/(4-2)=100
$X4 = c \longrightarrow X1 = a$	2,6	1,5	2/4=50	2/(4-2)=100
$X4 = d \longrightarrow X1 = d$	2,6	1,5	2/4=50	2/(4-2)=100

Fig. 3. Correction of confidence for missing values

Representativity: To obtain good results a vdb must be a good sample of the database. This is normally true if values are missing at random. To avoid rules to be discovered in a small vdb (overspecified rules) we introduce a new parameter called *Representativity*.

$$Representativity(X) = \frac{|vdb(X)|}{|B|} = \frac{|B| - |Dis(X)|}{|B|}$$

To be usable an itemset X must have a representativity greater than a user-specified minimum representativity.

4.3 Comments

These new definitions are fully compatible with [2] and quick to calculate: like in [2] where $B_{XY} = B_X \cap B_Y$, we have $Dis(XY) = Dis(X) \cup Dis(Y)$.

We have seen that our approach is well suited to cope with random missing values. We show now that it always deals with informative missing values (like with usual approach). To do that we have to redefine a disabled data:

Definition 8 (Disabled data). t is disabled for X in B, if t contains missing values for at least one item i of X, where i does not represent an association with a missing value.

Example
Data 1: X1=a, X2=?, X3=a.
Data 1 is not disabled for the itemset {X1=a & X2=?} (the missing value for X2 could be informative).
Data 1 is still disabled for {X1=a & X2=c} and {X1=b & X2=c} (See Sect. 4.1).

If a missing value is informative for an attribute, itemsets which test missing values will be frequent. In the other cases, they will disappear because they will decrease under the user specified support (like with usual approach).

In the next section we propose a simulation with real databases where we compare the two approaches.

5 Results

5.1 Simulation

The simulation consists applying the association rules program, developed from[2] (see Sect. 2), to a database (without missing values) to obtain a ruleset called *original ruleset*. Next 5% of random missing values per attribute are introduced and the program is rerun with [2] and our new approach. The second step is repeated with 10, 15, 20, 25 and 30% of missing values per attribute. We compare results regarding the number of retrieved rules, i.e rules which were present in the original ruleset, and the number of new rules, i.e rules which were not in the original ruleset.

This simulation is made with two databases. The first database has been generated with Synthetic Classification Data Sets (SCDS) program[2], developed by Gabor Melli. This program generates synthetic data sets which are particularly useful to test Knowledge Discovery from Database (KDD) algorithms.This database has 2000 data. Each data contains 8 attributes with 4 different possible values [3]. The second database is a classical one in machine learning community. Vote is composed by the Voting records drawn from the Congressional Quarterly Almanac, 98th Congress, 2nd session 1984, Volume XL: Congressional Quarterly Inc. Washington, D.C., 1985. This database has 435 data, each with 16 attributes with 3 different possible values.

For the scds database we have used a minimum support of 10% and a minimum confidence of 80%. For the vote database we have used a minimum support of 40% and a minimum support of 80%. In our simulation, conclusions of the rules, i.e Y in a rule $X \longrightarrow Y$, are items (not itemsets) to be more accurate rules.

Results are presented in Fig. 4. The new approach is more robust to the

[2] http://fas.sfu.ca/cs/people/GradStudents/melli/SCDS/
[3] The command line to generate this database is scds -v -O2000 -R3 -A8 -d4 -C5 -D2

Database SCDS, original ruleset: 2975 rules.

% of missing values	5	10	15	20	25	30
Usual approach						
Retrieved rules	394	123	68	33	0	0
Missing rules	2581	2852	2907	2942	2975	2975
New rules	0	1	0	1	0	0
New approach						
Retrieved rules	2816	2752	2603	2509	1773	1784
Missing rules	159	223	372	466	1202	1191
New rules	36	29	57	54	20	0

Database Vote, original ruleset: 216 rules.

% of missing values	5	10	15	20	25	30
Usual approach						
Retrieved rules	41	4	1	0	0	0
Missing rules	175	212	215	216	216	216
New rules	0	0	0	0	0	0
New approach						
Retrieved rules	211	145	178	155	137	121
Missing rules	5	71	38	61	79	95
New rules	48	13	46	100	54	30

Fig. 4. Comparison between usual and new approaches

missing values. An important point is that new rules discovered by the new approach are not false. In fact a vdb for a rule could not be exactly equivalent to the original database. Actually support and confidence are true with an epsilon error. With Vote (resp. SCDS) , new rules are present in a original ruleset with support of 38% (resp. 8%) and confidence of 78% (resp. 78%).

5.2 Medical Experimentation

We use association rules to discover useful relations in a database concerning the Hodgkin disease. The Hodgkin disease is a cancer. At the present time, there are several efficacious treatments according to the stage of disease. The consequences are important (leukemia ...) for last stages. The aim of this experiment is to try to define more accurately the different stages of the disease to apply the best treatment.

The database is collected by the lymphome group of the O.E.R.T.C (European Organization of Research and Treatment of Cancer). At the present time, it is managed by the François Baclesse center of Caen and had involved 2460 people since 1964. We have studied the H7 protocol concerning 832 people (1988-1993). There are 27 attributes and 11 of them have missing values (52%, 43%, 88%, 84%, 39%, 8%, 6%, 7%, 2%, 6% and 6.5%).

According to a minimum support of 50% and a minimum confidence of 80% we have found, with the usual treatment, 17133 rules. With our new treatment we have found 51286 rules (three times more). Previous simulations makes one think that new rules are useful. At present we work with experts to evaluate them. First results are of a great interest.

6 Conclusion

Association rules are very useful to explore very large databases. In consequence they are well adapted to explore real-world applications. In these domains missing values are often inevitable. In [2] which has proposed the core of all known association rules algorithms missing values have not been considered of interest: association rules have been first made to mine association from transaction databases where practically missing attribute values do not exist. Consequently actual association rules algorithms give very bad results with databases containing missing values (See Sect. 5): generally they can not be treated like a proper attribute value [9] [12] [6]. In this paper we have proposed a new way to deal efficiently with informative and random missing values, completely compatible with [2]. The results of Sect. 5 show that this approach is much more robust than the older one. In our real world experiment we have used this approach with a medical databases containing missing values to help experts: with the usual treatment we have found 17133 rules, and with the new one we have found 51286 rules.

In the future, an interesting application could be to explore data containing missing values and to use strong rules, i.e with high confidence, to fill missing values. This method could replace advantageously usual treatments of missing values in decision tree where missing values are filled with poor confidence and where they are often replaced by noise [14] [13] [6] [5].

References

1. R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In Proc. of the ACM SIGMOD Conference on Management of Data, Washington, D.C., p 207-216, May 1993.
2. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A. I. Verkamo. Fast Discovery of Association Rules. In Advances in Knowledge Discovery and Data Mining, Chapter 12, AAAI/MIT Press, 1996.
3. R. Agrawal, R. Srikant. Fast algorithms for mining association rules in large databases. In Proceedings of the Twentieth International Conference on Very Large Databases (VLDB'94), p. 487 - 499, September 1994.
4. K. Ali, S. Manganaris, R. Srikant: Partial Classification using Association Rules, in Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining, Newport Beach, California, August, 1997.
5. L. Breiman, J.H Friedman, R.A Olshen, C.J Stone. Classification and Regression Trees, Wadsworth International Group, Belmont, CA, The Wadsworth Statistics/Probability Series, 1984.

6. G. Celeux. Le traitement des données manquantes dans le logiciel SICLA. Technical reports number 102. INRIA - December 1988.
7. B. Crémilleux, C. Robert. A pruning method for decision trees in uncertain domains: applications in medicine. International Workshop on Intelligent Data Analysis in Medicine and Pharmacology, ECAI 1996, p 15-20, Budapest 1996.
8. B. Crémilleux, C. Robert. A theorical framework for decision trees in uncertain domains: application to medical data sets. 6th Conference in Artificial Intelligence in Medicine Europe (AIME 97), p 145-156, Grenoble 1997.
9. W.Z Liu, A.P White, S.G Thompson and M.A Bramer. Techniques for Dealing with Missing Values in Classification. In Second International Symposium on Intelligent Data Analysis, Birkbeck College, University of London, 4th-5th August 1997.
10. H. Mannila, H. Toivonen and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94), p. 181-192, Seattle, Washington, July 1994.
11. J.R Quinlan. Induction of decision trees. Machine learning, 1, p. 81-106, 1986.
12. J.R Quinlan. Unknown Attribute Values in Induction, in Segre A.M.(ed.), Proceedings of the Sixth International Workshop on Machine Learning, Morgan Kaufmann, Los Altos, CA, p. 164-168, 1989.
13. J.R Quinlan. C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA, 1993.
14. A. Ragel: Traitement des valeurs manquantes dans les arbres de décision. Technical reports, Les cahiers du GREYC. University of Caen, 1997.
15. A. Savasere, E. Omiecinski, S. Navathe. An efficient algorithm for mining association rules in large databases. In proceedings of the 21st International Conference on Very Databases (VLDB'95), p. 432 - 444, Zurich, Swizerland, 1995.
16. H. Toivonen. Sampling large databases for association rules. In Proc. of the 22nd Int'l Conference on Very Large Databases (VLDB'96), p. 134-145, Bombay, India, September 96