UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIA

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL DA SILVA ALBUQUERQUE

A ROUGH SETS-BASED RULE INDUCTION FOR NUMERICAL DATASETS

FORTALEZA

2019

RAFAEL DA SILVA ALBUQUERQUE

A ROUGH SETS-BASED RULE INDUCTION FOR NUMERICAL DATASETS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Computação do Centro de Ciência da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciências da computação. Área de Concentração: Lógica e Inteligência Artificial

Orientador: Prof. Dr. João Fernando Lima Alcântara

FORTALEZA

2019

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A313r Albuquerque, Rafael da Silva.
A Rough Sets-Based Rule Induction for Numerical Datasets / Rafael da Silva Albuquerque. – 2019.
58 f.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2019.
Orientação: Prof. Dr. João Fernando Lima Alcântara.

1. Rough Set. 2. Rule Induction. 3. Interpretability. 4. Numerical Data. I. Título.

CDD 005

RAFAEL DA SILVA ALBUQUERQUE

A ROUGH SETS-BASED RULE INDUCTION FOR NUMERICAL DATASETS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Computação do Centro de Ciência da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Ciências da computação. Área de Concentração: Lógica e Inteligência Artificial

Aprovada em: 12 de Março de 2019

BANCA EXAMINADORA

Prof. Dr. João Fernando Lima Alcântara (Orientador)
Universidade Federal do Ceará (MDCC/UFC)

Prof. Dr.ª Ana Teresa de Castro Martins
Universidade Federal do Ceará (MDCC/UFC)

Prof. Dr. César Lincoln Cavalcante Mattos
Universidade Federal do Ceará (UFC)

Prof. Dr. Ajalmar Rêgo Rocha Neto
Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)

À minha família, por sua capacidade de acreditar em mim e investir em mim.

ACKNOWLEDGEMENTS

To my parents and brothers, for the encouragement during this journey;

To Prof. Dr. João Fernando Lima Alcântara for guiding me in my master's dissertation and for his patience during this process;

I would like to thank Professors César Lincoln Cavalcante Mattos, Ajalmar Rêgo Rocha Neto and Ana Teresa for their participation in the Master's Examination Committee;

To my friends from LOGIA for all help during the development of this work;

And to the Cordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), for the financing of the master's degree research through scholarship.

“Education is a progressive discovery of our own ignorance.”

(Will Durant)

RESUMO

Tirar conclusões razoáveis a partir de dados do mundo real tem sido um desafio devido a diversos fatores relacionados à qualidade da informação. Para lidar com esses problemas, foi proposta a teoria dos conjuntos aproximados, que trata da inconsistência através da aproximação de conjuntos de dados. Entre as aplicações de conjuntos aproximados, destaca-se a sua utilização nos processos de aprendizagem, devido à sua capacidade de produzir modelos de classificação interpretáveis. Apesar do sucesso, alguns dos métodos baseados em conjuntos aproximados mais usados são projetados para trabalhar com dados de entrada categóricos. Essa opção de design pode limitar severamente sua aplicação a problemas do mundo real. Esses métodos também são inadequados para lidar com problemas de classificação binária. Utilizando métodos de discretização e árvores de decisão, conseguimos superar tais limitações e melhorar a qualidade da classificação dos métodos utilizados. Como resultado, desenvolvemos três abordagens. A primeira abordagem apresentada produz resultados interpretáveis considerando a opção de rejeição. A segunda abordagem faz uso de técnicas de fusão de crença para reduzir o número de objetos rejeitados na primeira abordagem. Por fim, a terceira abordagem faz uso de árvores de decisão para classificar todos os casos rejeitados.

Keywords: Conjunto áspero. Indução de regras. Interpretabilidade. Dados numéricos.

ABSTRACT

Drawing reasonable conclusions from real-world data has been a challenge owing to diverse factors related to the quality of information. In order to handle these problems, the rough sets theory, which deals with inconsistency through the approximation of data sets, was proposed. Among the applications of rough sets, their use in learning processes is highlighted due to their capacity to produce interpretable classification models. Despite their success, some of the most commonly used rough sets based methods are designed to work with categorical input data. This design choice can severely limit their application to real-world problems. Such methods are also inappropriate to handle binary classification problems. By using discretization methods and decision trees we were able to overcome such limitations and improve the classification quality of the methods used. As a result, we developed three approaches. The first presented approach produce interpretable results considering the rejection option. The second approach makes use of belief merging techniques to reduce the number of rejected objects in the first approach. At last, the third approach makes use of decision trees to classify all rejected cases.

Palavras-chave: Rough Set. Rule Induction. Interpretability. Numerical Data.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Working with real world data is a challenging task due to the presence of elements that may interfere with the quality of information. Quality and limitation of measuring instruments, interference of external elements and lack of control over the environment are some of the factors that may affect the quality of information. As a consequence, datasets may be inconsistent. According to (GRECO *et al.*, 2001), a dataset is called inconsistent when a portion of it is composed of objects satisfying a property, but with the same description of objects satisfying a different property. The presence of such kind of data may influence negatively the performance of data processes. More details are given in Example 1.

Because of the constant presence of uncertainties in the world, methods to work with imprecise information have become indispensable. Being able to handle inconsistent data makes information processes more reliable. Medical and predictive processes need to be reliable, so they can perform their functions in the right way. Also, as learning processes can be used to identify objects through patterns, the presence of inconsistent data also affects their performance. Thus, when learning processes work with inconsistent data, patterns that do not correspond to reality may arise, which may compromise the proper functioning of methods using it.

Example 1. We exhibit a set of objects describing cars from different brands. Such objects are characterized by attributes q_1, q_2, q_3 and the class attribute q_4, which specifies the brand of each object. Any set of objects defined by the same value of attribute class q_4, is called a concept.

Object (U)	Number of doors (q_1)	Horsepower (q_2)	Colour (q_3)	Make (q_4)
x_1	2	60	blue	Opel
x_2	2	100	black	Nissan
x_3	2	200	black	Ferrari
x_4	2	200	red	Ferrari
x_5	2	200	red	Opel

Table 1 – Example Datasets

In Table 1, information set is inconsistent because objects x_4 and x_5 are conflicting, i.e., both of them have the same values for q_1, q_2, and q_3, but different values for the decision class q_4. Thus, if we were asked to classify both objects x_4 and x_5, we would be uncertain about where to place each of them. Such kind of inconsistency may have been caused by noise during the measurement process, some limitation of the measurement instrument or by the limited

discriminatory power of attributes.

In order to deal with imprecise information, several methods such as fuzzy sets (ZADEH, 1997), soft sets (MAJI *et al.*, 2003), rough sets (PAWLAK, 1982), genetic algorithms (BEASLEY; CHU, 1996) and others compose what is known as soft computing. As observed in (ZADEH, 1996), methods in these areas work with imprecise information through the use of approximate calculations to provide imprecise but usable solutions to complex computational problems. These are some of the characteristics making soft computing methods able to deal with partial truths, inconsistent and incomplete information.

Among soft computing methods, rough sets have gained attention given their simplicity and effectiveness in dealing with inconsistent information (ZHANG *et al.*, 2016). Rough sets theory was proposed by Pawlak (PAWLAK, 1982). The rough sets theory is an extension of the classical set theory, considering that it is possible to have an imprecise definition for a set X, so defining it by approximations. These approximations are composed of elements that are certainly in X and elements that are possibly in X. Such approximations are called lower and upper approximation respectively and are denoted as $\underline{X}$ and $\overline{X}$ respectively. For a set X we say that a pair $(\underline{X},\overline{X})$ is its approximation space; when $\underline{X} = \overline{X}$ we say X is perfectly defined and in this case it is just like a classical set.

Backing to Example 1, we highlight approximate rough sets theory methods in some cases can avoid problems caused by inconsistent data. Through these approximations, we can isolate objects with ambiguous descriptions, such as x_4 and x_5. After isolating these objects, we can work with those objects whose description certainly matches the decision value.

Due to these characteristics of rough sets based methods, several methods and applications were developed (ZHANG *et al.*, 2016). Some of the features and applications of methods using rough sets are exploited in the next section.

1.1 Motivation

As described in (ZHANG *et al.*, 2016), rough sets theory has been widely used in machine learning, data mining, decision support, and analyses, etc. Some of the reasons explaining its success are due to its solid mathematical base and its easily interpretable results, as well as other good characteristics:

- The mathematical structure of rough sets is mature;
- It does not need any prior knowledge;

- Rough sets models are simple and easy to be calculated and
- Rough sets based methods are secure and robust.

Over the years, rough sets theory has been employed in several areas such as feature selection (HU *et al.*, 2008; INUIGUCHI, 2005), conflict analysis (SKOWRON *et al.*, 2006; PAWLAK, 2006), rule induction (ZHAO *et al.*, 2006; INUIGUCHI; MIYAJIMA, 2007) and decision-making process (GRECO *et al.*, 2000; GRECO *et al.*, 2001). Rough sets methods have also been used in several applications. Some of these applications are: intelligent industrial control (MROZEK, 1992), decision support systems (GOLAN; ZIARKO, 1995; PAWLAK; SLOWINSKI, 1994) and technical diagnosis of mechanical objects (SLOWINSKI; ZOPOUNIDIS, 1995; STEFANOWSKI *et al.*, 1992; NOWICKI *et al.*, 1992).

Of all the applications of rough sets theory, the rule induction methods (GRZYMALA-BUSSE, 1997; INUIGUCHI; MIYAJIMA, 2007) are one of the most relevant techniques of machine learning and data mining (STEFANOWSKI, 1998). Methods to represent knowledge by using rules are of great interest. As they can produce interpretable results, which are desirable, as they provide a better understanding of the reasons behind the obtained results.

Interpretable results are useful when dealing with critical systems and other situations, as they can be reviewed by a specialist. As an example of critical systems, we can quote to medical diagnostic and decision support systems. In both of these systems, erroneous results may cause an injury to a patient and to economic loss respectively. Therefore, interpretable results allow a specialist to review the result and intervene when necessary.

In short, rough sets based methods have some good qualities. However, although their qualities, they have a serious problem which restricts their application domain: rough sets based rule induction methods are not tailored to work with numerical data. More details are given in the next section.

1.2 Observed Problem

In spite of the good qualities of rough sets theory, rough sets based rule induction methods are not appropriate to work with numerical data, as can be observed in (HU *et al.*, 2005). The major reason for this limitation is that rough sets based methods commonly make use of equivalence classes, which are commonly used to cluster objects according to their descriptions. Although equivalence classes are applicable over numerical data, they are inappropriate to deal with this kind of information, as observed in Example 2.

In a numerical environment, given the wide variety of possible descriptions that an object may assume, it is unlikely that two objects have the exact same description. Such a characteristic causes equivalence classes to be unsuitable to work with numerical information, which considerably diminishes the application domain of rough sets based rule induction methods. The reasons why rough sets based rule induction methods are not suitable to work with numerical information can be viewed in more detail in Example 2.

Example 2. Now we present a dataset composed of a set of objects described by numerical attributes q_1, q_2, q_3, and a decision attribute q_4, specifying to which class each object is part. In this example, our objective is to observe some of the possible problems one can face when working with numerical information:

Object (U)	Height (q_1)	Weight (q_2)	Age (q_3)	Male (q_4)
x_1	151.7	47.8	63	1
x_2	139.7	36.4	63	0
x_3	136.5	31.8	65	0
x_4	156.8	53.0	41	1
x_5	154.4	41.2	51	0
x_6	163.8	62.9	35	1
x_7	149.2	38.2	32	0

Table 2 – Numerical Datasets Description

In Table 2, all objects have a different description from all others. Indeed, we can affirm it is highly unlikely that in a numerical dataset, two or more objects have exactly the same description. As a consequence, each equivalence class will have a unique object, which makes it hard to group objects satisfying the same decision value.

Over the years, several extensions of rough sets theory to approximate sets described by numerical information were presented (GRECO *et al.*, 2003; STEFANOWSKI, 1998; TRABELSI *et al.*, 2010; OHKI *et al.*, 2011; KRYSZKIEWICZ, 1999; TRABELSI *et al.*, 2011). These extensions significantly expanded the application domain of these methods. However, despite the advances, these extensions are still not enough to turn rough sets based rule induction methods applicable over numerical information.

Rough sets based rule induction methods aim to identify patterns characterizing sets of objects. However, given the diversity of descriptions that an object can assume in a numerical environment, finding a pattern is not an easy task. As observed in this section,

approximations are computed by using equivalence classes. Also, from Example 2, it is visible that equivalence classes are not appropriate to group objects described by numerical information. These observations make it clear that these methods are best suited for working with discrete information. Therefore, as will be discussed in the next chapter, finding a solution to this problem is one of the main points of this work.

1.3 Objective

In this work, we aim to find ways of making rough sets based induction methods suitable to numerical data, making these methods applicable on numerical data, would make it possible to apply it in a greater number of situations. The interest for this is due to the ability of these methods to work with inconsistent data. Making these methods more suitable to deal with real-world data.

We will focus on two rule induction methods. The first method, which is named *LERS* (Learning From Examples using Rough Sets) (GRZYMALA-BUSSE, 1997) is intended to deal with inconsistencies by using the principles of approximation of rough sets theory. In the second method, which was proposed by Masahiro Inuiguchi (here called *IM*) approach (INUIGUCHI; MIYAJIMA, 2007), the induction process consists of grouping all the characteristics satisfied by positive examples and not satisfied by negative examples.

Both *LERS* and *IM* were originally designed to perform binary classification. In order to generalize them to any arity, our approach will perform the induction process several times, one time for each class value. As a result, it gives us sets of rules identifying objects satisfying each of the decision values. However, as a drawback, an object can be either satisfied by two or more rules with different decision classes as a conclusion or even not satisfied by any of the resulting rules. In both cases, we are unable to classify it. In this work, these objects are called rejected. However, as rejection is not desirable in all classification problems we intend to develop some approaches that can reduce or completely eliminate the rejected cases.

Over the years several approaches considering rejection option have been presented. Such classification methods with rejection option have proved to be useful in several applications, mainly for critical systems as medical diagnostic systems, as misclassifying a sick patient as healthy is worse than the opposite. Some references considering classification problems with rejection option are (HERBEI; WEGKAMP, 2006), (CORTES *et al.*, 2016), (SOUSA *et al.*, 2014), (YUAN; WEGKAMP, 2010). As observed in (SOUSA *et al.*, 2014), rejection option may

be used to increase confidence in classification results, as in classification systems using rejection option, hard classification problems are rejected instead of taking the risk of misclassifying it.

In this work, our specific objectives are: apply rough sets based rule induction methods on numerical data, extend *LERS* and *IM* to classification problems with multiple classes and classify the rejected cases generated in problems with multiple classes.

1.4 Contributions

In order to achieve our goal and make rough sets based rule induction methods (*LERS* and *IM*) applicable over numerical data, we will apply discretization methods as a pre-processing step before the execution of the rule induction methods. In the discretization step, we apply the well known method MDLP (Minimum Description Length Principle) (RISSANEN, 1978).

For both rule induction methods, three methods were considered to performance evaluation in the presence of rejected cases: pessimistic, safe and optimistic: the pessimistic approach counts the cases rejected as error, the safe approach does not consider rejected objects during the test phase and optimistic approach only count as error objects not satisfying any decision rule. These approaches are described in more detail in Chapter 5.

In order to reduce the number of rejected objects, which were obtained as a result of the extension of both *LERS* and *IM* to more than binary classification problems, we describe a method for multiclass classification based on the partial satisfiability (PARRA; MACÍAS, 2007) of the resulting rules. This process is described in more detail in Section 4.2. Notwithstanding, the partial satisfiability is not enough to classify all rejected objects. Thus, although drastically reduced in number, rejected objects may still remain.

This disambiguation process (Section 4.2) based on the partial satisfiability can produce a classification value for a rejected object and no additional information can be applied to explain the reason behind the result. Thus, we say the disambiguation process is not interpretable. Therefore, the association of classification methods as *LERS* and *IM*, which are completely interpretable, with the disambiguation process, results in a semi-interpretable classification method, as not all objects are classified according to a logical rule. Although the resulting method is semi-interpretable, as can be seen in Section 5.1, we emphasize its application drastically reduces the number of rejected objects.

In addition, we propose an alternative solution to deal with the rejected objects: for each of them, we resort to a decision tree, (SAFAVIAN; LANDGREBE, 1991), a well known

interpretable classification method, to obtain its classification value. In this case, we not only eliminate every rejected object as also preserve the interpretability of the results for any object. Besides, as our experimental proceedings revealed in Chapter 5, we have even slightly improved the classification accuracy of the decision trees. In short, our contributions in this work are

- extension of both *LERS* and *IM* for arity > 2 (multiclass task),
- induction over numerical data using *LERS* an *IM*, and
- improvement in the classification accuracy of decision trees

Part of this work has been published in (ALBUQUERQUE *et al.*, 2018).

1.5 Structure

This work is organized as follows. Chapter 2 presents the basic concepts as rough sets, decision trees and discretization methods (MDLP). The two rough sets based rule induction techniques *LERS* and *IM* are shown in Chapter 3. Chapter 4 presents our contribution to make rule induction methods applicable over numerical data and also the method to classify rejected objects. The methodology and the results of the experiments are in Chapter 5. Conclusion and future works are in Chapter 6.

2 BACKGROUND

In this chapter we briefly discuss some methods and concepts used in this work. In particular, we present the theory of rough sets (PAWLAK, 1982), a classification method using decision trees (SAFAVIAN; LANDGREBE, 1991) and the discretization method MDLP (Minimal Description Principle) (RISSANEN, 1978). The methods presented in this chapter are used in association with each other to approach the problems previously mentioned (Section 1.2).

2.1 Rough Sets

When working with objects described by real world information, we must take into account that, due to noise and missing data, the description for these objects may not be in accordance with the real world. In addition, as this description becomes less precise, our ability to distinguish them tend to decrease. The presence of uncertain knowledge brings imprecision to the result of the methods applied over these information, which is not acceptable for critical systems, where failures may result in irreparable damage.

In this work, we refer to uncertainty/imprecision as a portion of knowledge that there are some doubts over. In rough sets theory, uncertainty is represented by a boundary set. Given these observations, rough sets were presented to deal with uncertain information by using sets approximation. The approximation process can be performed in two ways, which are called lower and upper approximation. Through these approximations, it is possible to minimize the influence of the uncertain information in the data process. We assume that the information describing objects is organized in what we call dataset:

Definition 1 (Information systems)**.** An information system S is a 4-tuple (U, A, V, f), where $U = \{x_1, x_2, \ldots, x_{|U|}\}$ is a non-empty finite set of individuals; $A = \{a_1, a_2, \ldots, a_{|A|}\}$ is a non-empty finite set of attributes such that $A = C \cup D$ and $C \cap D = \emptyset$, in which C and D denote the sets of condition attributes and decision attributes, respectively. $V = \{V_{a_1}, V_{a_2}, \ldots, V_{a_{|A|}}\}$ is a domain attributes, where each V_{a_i} is the domain of attribute a_i. Finally, $f = U \times A \rightarrow V$ is an information function such that $f(x, a) \in V_a$ for all $x \in U$ and for all $a \in A$.

Let $X \subseteq U$ a set of objects we want to represent using attribute set $P \subseteq C$. We consider that X is defined by a set of objects that have the same value for decision attribute.

Example 3. (RUTKOWSKI, 2008) Let $S = (U, C \cup D, V, f)$ be an information system such that $U = \{x_1, \ldots, x_{10}\}$, $C = \{q_1, q_2, q_3\}$, $D = \{q_4\}$, $V_{q_1} = \{2,3,4\}$, $v_{q_2} = \{60, 100, 200\}$, $v_{q_3} = \{\text{blue}, \text{black}, \text{red}\}$, $v_{q_4} = \{\text{Opel}, \text{Nissan}, \text{Ferrari}\}$ and f is defined as indicated below:

Object (U)	Number of doors (q_1)	Horsepower (q_2)	Colour (q_3)	Make (q_4)
x_1	2	60	blue	Opel
x_2	2	100	black	Nissan
x_3	2	200	black	Ferrari
x_4	2	200	red	Ferrari
x_5	2	200	red	Opel
x_6	3	100	red	Opel
x_7	3	100	red	Opel
x_8	3	200	black	Ferrari
x_9	4	100	blue	Nissan
x_{10}	4	100	blue	Nissan

Table 3 – Datasets Description

Note that, any subset X of examples from information system in Example 3 defined by objects with the same value for decision attribute is named a concept. The information system in example 3 consists of 3 concepts, they are defined by examples satisfying $(Make, Opel)$, $(Make, Nissan)$ and $(Make, Ferrari)$. In general, X cannot be well defined, as the system may present objects having a similar description and satisfying different values for decision attribute. Now we drive our attention to a very important definition in the rough sets theory: indiscernibility relation.

Definition 2 (Indiscernibility relation)**.** Let $(U, C \cup D, V, f)$ be an information system and $B \subseteq C$ a set of condition attributes. We obtain the indiscernibility relation

$$R_B = \{(x,y) \in U \times U \mid \forall a \in B, V_a(x) = V_a(y)\}$$

We note that indiscernibility relation is reflexive, symmetric and transitive, i.e., it is an equivalence relation. Thus, R_B divides U into a family of disjoint sets, which are called the equivalence classes of R_B:

Definition 3 (Equivalence classes)**.** Let $(U, C \cup D, V, f)$ be an information system, $B \subseteq C$ a set of condition attributes and R_B the associated indiscernibility relation. The equivalence class of R_B for each $x \in U$, denoted $[x]_B$, is given by

$$[x]_B = \{y \in U \mid (x,y) \in R_B\}$$

The equivalence class $[x]_B$, also called a basic granule of U, describes all objects which cannot be discerned from x according to R_B.

Example 4. Recalling Example 3, we have

$$[x_1]_C = \{x_1\} \qquad [x_4]_C = [x_5]_C = \{x_4, x_5\} \tag{2.1}$$

$$[x_2]_C = \{x_2\} \qquad [x_6]_C = [x_7]_C = \{x_6, x_7\} \tag{2.2}$$

$$[x_3]_C = \{x_3\} \qquad [x_9]_C = [x_{10}]_C = \{x_9, x_{10}\} \tag{2.3}$$

$$[x_8]_C = \{x_8\} \tag{2.4}$$

From set space U and indiscernibility relation R_B, we obtain the tuple (U, R_B), which is known as *Pawlak approximation space* (or simply approximation space). Furthermore, let $X \subseteq U$ represent a concept in (U, R_B), e.g., X is the set of all objects whose car make is Ferrari. Considering any real information system is usually imprecise, incomplete, uncertain and even contradictory, one can easily guess a fully description of X by the equivalence classes $[x]_B$ is not possible. This means that, determining if an object $x \in U$ belongs to X based on the knowledge of values of their features may not be unequivocal.

Example 5. In Example 3, let $X_F = \{x_3, x_4, x_8\}$ be the set of cars whose make is Ferrari. Note the description given by x_4 ($q_1 = 2$, $q_2 = 200$, $q_3 =$ red) does not suffice to characterize a Ferrari, as this is also a description provided by x_5 of an Opel.

From Example 5, we note that the information system in Example 3 is inconsistent since x_4 and x_5 are conflicting examples. With such motivations in mind, Pawlak introduced the lower and upper approximation operators with respect to an indiscernibility relation R_B:

Definition 4. [Lower and upper aproximations] Let $(U, C \cup D, V, f)$ be an information system, $X \subset U$, $B \subseteq C$ a set of condition attributes and R_B the associated indiscernibility relation. We define both the lower approximation of X with respect to R_B (denoted by $\underline{X}_B$) and the upper approximation of X with respect to R_B (denoted by $\overline{X}_B$) as follows:

- $\underline{X}_B = \{x \in U \mid [x]_B \subseteq X\}$
- $\overline{X}_B = \{x \in U \mid [x]_B \cap X \neq \emptyset\}$

From Definition 4, it is obvious that $\underline{X}_B \subseteq \overline{X}_B$. Furthermore, $\underline{X}_B$ is the set of the objects $x \in U$ of whom we can state they are certainly elements of X with respect to R_B, while,

$\overline{X}_B$ is the set of the objects $x \in U$ of whom we can state they are *possibly* elements of X with respect to R_B.

Considering again our running example, where $X_F = \{x_3, x_4, x_8\}$ is the set of cars whose make is Ferrari, we obtain

- $\underline{X_{FC}} = \{x_3, x_8\}$
- $\overline{X_{FC}} = \{x_3, x_4, x_5, x_8\}$

This means that, according to R_C, x_3 and x_8 are certainly the description of a Ferrari, whereas the attribute values in x_4 and x_5 are possibly the description of a Ferrari. Given these definitions for lower and upper approximation, we can now present what is known as positive, negative and boundary sets for X:

Definition 5. Let $(U, C \cup D, V, f)$ be an information system, $X \subset U$, $B \subseteq C$ a set of condition attributes. We define the positive, negative and boundary sets for X according to R_B, denoted respectively as $Pos_B(X)$, $Neg_B(X)$ and $Bon_B(X)$:

- $Pos_B(X) = \underline{X}_B$
- $Neg_B(X) = U - \underline{X}_B \cup Bon_B(X) = U - \overline{X}_B$
- $Bon_B(X) = \overline{X}_B - \underline{X}_B$

In particular, note the negative set $Neg_B(X)$ characterizes those objects that certainly are not in X according to R_B. Recall that in an information system S, each subset $X \subseteq U$ containing the same decision value is called a *concept*. Thus, uncertainty is associated with the boundary set $Bon_B(X)$. We say a concept X is *imprecise* (with respect to R_B) if $Bon_B(X) \neq \emptyset$. In our example, it is clear that $Pos_C(X_F) = \{x_3, x_8\}$, $Neg_C(X_F) = \{x_1, x_2, x_6, x_7, x_9, x_{10}\}$ and $Bon_C(X_F) = \{x_4, x_5\}$ (X_F is an imprecise concept with respect to R_C). These notions are summarized in Fig. 1.

A decision class is perfectly definable when its boundary region is empty, and so an information system is consistent when the boundary region for all concepts is empty. In case the information system is inconsistent due to uncertain information, it is possible to measure its reliability. The quality of approximations is presented in Definition 6.

Definition 6. [Quality of an approximation] Let $(U, C \cup D, V, f)$ be an information system, $X \subset U$, $B \subseteq C$ a set of condition attributes. We define the quality of the rough sets approximation for set X as $Q(X) = \frac{|\underline{X}_B|}{|\overline{X}_B|}$.

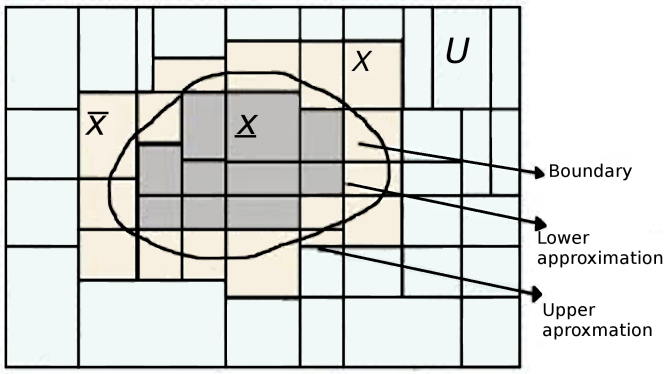

Figure 1 – Rough set representation

Although we usually consider the attribute set D of an information system is a singleton, it is also possible to conceive an information system with more than one decision attribute. In this case, the combination of decision attribute values can be seen as a single decision class, and we can proceed as usual. Thus, the definitions for positive, negative and boundary sets are not only definable over a single decision class, but it is also definable over multiple decision classes.

As presented in this chapter, rough sets theory uses indiscernibility relation to identify different sets of objects. Each set is characterized by a set of attribute values forming a unique combination for each set. Through the use of indiscernibility relation, two approaches for approximating sets were defined (lower/upper), which can be used to reduce the influence of uncertain information. The capacity of rough sets in approximating different sets of objects make it useful for classification and induction in applications as described in (TSUMOTO, 2003; TSUMOTO, 1998; LAW; AU, 2000).

2.2 Decision Tree

Decision trees are a family of inductive learning methods. Some of its advantages is that it produces interpretable models and the method itself is simple to understand. Some of the applications of decision tree methods are in regression and classification problems, (SAFAVIAN; LANDGREBE, 1991). Over the years some variations have been presented. Some of these variations are Bayesian decision trees(KOHAVI, 1996) and random forest (BREIMAN, 2001). In this work we are interested in applying decision trees to classification problems.

A decision tree is built by selecting cut points in attributes domain that best split a set of examples S into subsets $S_1, S_2, \ldots, S_k$, composed of objects satisfying different decision values, for all S_i and S_j such that $i \neq j$ we have $S_i \cap S_j = \emptyset$. To choose a cut point, the domain

of all description attributes is verified. The partitioning process is recursively applied until all objects in the same set have the same decision value or until splitting a set does not provide any information gain. From all possible cut points, the selected cut point is the one that minimizes the Gini index value, defined in the sequence.

Definition 7. [Gini Index](RAILEANU; STOFFEL, 2004) The Gini coefficient is a measure of how often an object is wrongly classified if it was randomly classified according to the proportions of elements satisfying different class value. Let $P_i = \{p_{i1}, p_{i2}, \ldots, p_{ik}\}$, where p_{ij} is the proportion of elements in S_i that are in class C_j.

$$Gini(P_i) = \sum_{j=1}^{k} p_{ij} \sum_{l \neq j} p_{il} = \sum_{j=1}^{k} p_{ij}(1 - p_{ij}) = \sum_{j=1}^{k} (p_{ij} - p_{ij}^2) = \sum_{j=1}^{k} p_{ij} - \sum_{j=1} p_{ij}^2 = 1 - \sum_{j=1}^{k} p_{ij}^2$$

Let $x \in S$ and $T = f(x, a_i) \in V_{a_i}$ be the selected cut point. Let S_i and S_j partitions of S such that, $\forall x \in S_i$, $f(x, a_i) = T$ and $\forall x \in S_j$, $f(x, a_i) \neq T$.

After S is partitioned, it is possible that $|S_i| \neq |S_j|$. Thus, Gini index must be weighted by the size of the set S_i, which results in $Gini(P_i) \times \frac{|S_i|}{|S|}$. As we are working with binary decision trees, S is always partitioned into only two sets. In this case, the Gini index associated with this partitioning is defined as

$$Gini(P_i) \times \frac{|S_i|}{|S|} + Gini(P_j) \times \frac{|S_j|}{|S|}.$$

When it is not possible to split S according to T, it is said that S is a leaf node. A node is said to be a leaf node when all objects in it have the same decision value or when after splitting it one of its subsets, S_i or S_j, is equal to S. When a leaf node is reached, the concluding classification value is determined by verifying the predominant decision value between objects in S_i.

Once the decision tree is completed, given an example x to be classified, the decision tree can be traversed by checking feature values of x against nodes conditions until it reaches a leaf node and so, attribute to it a decision value. In a decision tree, each path leading to leaf node can be interpreted as a logical rule concluding for each class which objects are part of it.

2.3 Bayesian Additive Regression Trees (BART)

Another decision tree based methods used in this work is the BART (Bayesian Additive Regression Trees), (KAPELNER; BLEICH, 2013). According to (CHIPMAN *et*

al., 2010), BART is a nonparametric Bayesian regression approach which uses dimensionally adaptive basis elements. We consider the problem of making inference about an unknown function f that predicts an output Y using a p dimensional vector of inputs $x = (x_1, ..., x_p)$ when

$$Y = f(x) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2). \tag{2.5}$$

To defined f, two approaches can be considered. The first approach consists in modeling f and the second consists in approximating $f(x) = E(Y|x)$, by a sum of m regression trees $f(x) = h(x) = \sum g_j$ where each g_j denotes a regression tree. Thus, 2.5 could be approximated by a sum-of-trees model

$$Y = h(x) + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2). \tag{2.6}$$

As observed in (CHIPMAN *et al.*, 2010), a sum-of-trees is fundamentally an additive model with multivariate components. Thus, BART uses a sum of trees to model or approximate $f(x) = E(Y|x)$. The idea behind it is to elaborate the sum-of-trees model 2.6 by imposing a prior that regularizes the fit by keeping the individual tree effects small. By weakening the g_j effects, BART ends up with a sum of trees, each of which explains a small and different portion of f.

To fit the sum-of-trees model, BART uses a tailored version of a Bayesian backfitting MCMC (Markov Chain Monte Carlo) that iteratively constructs and fits successive residuals. Inferences obtained from BART are based on successive iterations of the backfitting algorithm which are effectively an MCMC sample from the induced posterior over the sum-of-trees model space. For additional details see (CHIPMAN *et al.*, 2010). A single posterior mean estimate of $f(x) = E(Y|x)$ at any input value x is obtained by a simple average of these successive sum-of-trees model space.

Finally, by keeping track of the relative frequency with which x components appear in the sum-of-trees model iterations, BART can be used to identify which components are more important for explaining the variation of Y.

In the development of this work, we make use of decision trees in association with *LERS* (GRZYMALA-BUSSE, 1997) and *IM* (INUIGUCHI; MIYAJIMA, 2007) in order to solve some of the limitations presented by these methods. One of the reasons that motivated the choice of decision trees in this work was its simplicity and its characteristics of only one rule being satisfied at each time.

2.4 Minimum Description Length (MDLP)

The minimum description length (MDLP) principle is a formalization of Occan's razor. Given a dataset presenting some regularities, as any information set can be represented by a string of symbols, through MDLP principle, the best hypothesis explaining regularities in data is the one that achieves better data compression. Thus, the best hypothesis is the one that allow us to represent the data using the smallest number of symbols. MDLP was introduced by Jorma Rissanen in (RISSANEN, 1978).

One of the applications found for MDLP is for the discretization of continuous data. Such application showed-up to be of great interest, as most real-world classifiers involve continuous-valued attributes (FAYYAD; IRANI, 1993). The discretization method used here was primarily presented in (FAYYAD; IRANI, 1993). This algorithm addresses the use of entropy minimization heuristic to discretize the domain of continuous-valued attribute.

In the MDLP discretization process, given a numerical information system S and one of its description attributes $a_i \in C$, a value $T \in V_{a_i}$ (cut point) is selected to split S into $S_1 \subseteq S$ and $S_2 \subseteq S$, where $\forall x \in S_1,\ f(x,a_i) \leq T$ and $\forall x \in S_2,\ f(x,a_i) > T$. Let there k be classes $C_1,\ldots,C_k$ and let $P(C_i,S)$ be the proportion of examples in S that is part of class C_i. The class entropy (BICHSEL; SEITZ, 1989), of a subset of S is defined as:

$$Ent(S) = -\sum_{i=1}^{k} P(C_i,S)\log(P(C_i,S))$$

To measure the class entropy, after S is partitioned into S_1 and S_2, we take the weighted average of their resulting class entropy values.

Definition 8 (Class Entropy)**.** (BICHSEL; SEITZ, 1989) For an example set S, an attribute $a_i \in A$, and a cut value T: Let $S_1 \subset S$ be the subset of examples in S with a_i-values $\leq T$ and $S_2 = S - S_1$. The class information entropy of the partition induced by T, $E(a_i,T;S)$, is defined as

$$E(a_i,T;S) = \frac{|S_1|}{S}Ent(S_1) + \frac{|S_2|}{S}Ent(S_2)$$

Given Definition 8, a cut point T in attribute domain a_i is selected in order to minimize the value of $E(a_i,T;S)$. Such value T is used to produce a binary discretization of the domain of attribute a_i, which results in sets S_1 and S_2.

Once the training set is sorted according to the values of a_i, the algorithm is applied recursively, choosing a cut point T and producing a partition π_T at each time. However, it is

necessary to decide when to accept or reject a partition π_T. Let HT be the hypothesis that π_T induces if it were accepted. Thus, HT is the classifier that tests the value of a_i and classifies examples that have a value less than T according to examples in E. Similarly, NT represent the hypothesis obtained when π_T is rejected. In this case, E classifies examples according to classes in E without examining the value of a_i.

The problem of deciding to accept or reject π_T is a binary decision problem. In order to determine which decision to take, we resort to the calculation of the cost associated with each of these decisions. The cost associated with each decision is defined in the sequence. Values k_1 and k_2 are the numbers of classes in S_1 and S_2 respectively.

$$Cost(NT) = N.Ent(S) + k.Ent(S)$$

$$Cost(HT) = \log_2(N-1) + |S_1|.Ent(S_1) + |S_2|.Ent(S_2) + \log_2(3^k - 2) + k_1 Ent(S_1) + k_2 Ent(S_2).$$

The MDLP only accepts π_T (a partition) when $Cost(HT) > Cost(NT)$. The information gain associated with a point T in the domain of a conditional attribute a_i is defined as:

$$Gain(a_i, T; S) = Ent(S) - E(a_i, T; S) = Ent(S) - \frac{|S_1|}{N} Ent(S_1) - \frac{S_2}{N} Ent(S_2).$$

The above inequality, after dividing through by N reduces to

$$Gain(a_i, T; S) - \frac{\log_2(N-1)}{N} > \frac{\Delta(a_i, T; S)}{N}$$

where

$$\Delta(a_i, T; S) = \log_2(3^k - 2) - [kEnt(S) - k_1 Ent(S_1) - k_2 Ent(S_2)].$$

After these derivations, the adopted criterion to decide whether or not to accept a partition of S into S_1 and S_2 is

$$Gain(a_i, T; S) > \frac{\log_2(N-1)}{N} + \frac{\Delta(a_i, T; S)}{N}$$

Thus, we say a binary partition of N examples from S is accepted if and only if the above condition is true. This process is repeated for all continuous conditional attributes. For more details see (FAYYAD; IRANI, 1993).

Using the methods discussed in this chapter we present some solutions to the problems highlighted in the introduction (Section 1.2). By using rough sets theory we are able to

develop methods that can work with uncertain information. However, as discussed in section 4, methods based on the rough sets theory are not suitable for working with numerical data, in addition, these methods are focused on binary classification problems. To apply rough sets based rule induction methods on numerical data we use discretization methods such as MDLP. Other methods of classification as decision trees are used to solve some of the difficulties that arise when applying rough sets base classification methods in multiclass classification problems. Further details are given in Chapter 4.

Algoritmo 1: MDLP

Input : Numerical dataset S
Output : Returns a discretization for S

```
1 begin
2     DC = ∅;
3     for each a_i ∈ C do
4         DC = DC ∪ RecursiveSplit(S, a_i);
5     return DC;
```

In Algorithm 1, all description attributes related to a numerical dataset S are discretized one at each time. This discretization process is realized by applying Algorithm 2 over each of these attributes. In Algorithm 2, from lines 4 to 11 a cut point value T is selected in order to minimize the value of $E(a_i, T; S)$. Once a value for T has been selected, we proceed by recursively applying the discretization process to obtained sets $S_1, S_2 \subseteq S$ until the partitioning of S into S_1 and S_2 does not provide any additional information gain, according to condition in line 13. Once the condition in line 13 in Algorithm 2, is unsatisfied, a set of partitions of S considering description a_i is returned in line 17. Each discretization considering a different attribute a_i is stored in set DC in line 4 of the Algorithm 1.

One of the advantages in applying the discretization method MDLP, is that through its application we are able to apply rough sets based rule induction methods over numerical data. Such capacity turns it possible to extend the application of these methods over a large rang of applications and so bring some of the good qualities presented by rough sets based rule induction methods to these applications. However, although the MDLP turn it possible to apply rough sets based rule induction methods over numerical information, due to the discretization process it is unavoidable to lose information, which results in an increase in the amount of uncertain information. Despite this disadvantage, we hope to surpass it by making use of some approaches subsequently presented in Chapter 4.

Algoritmo 2: RecursiveSplit

Input : Numerical dataset S and one of its description attribute a_i
Output : Returns a discretization for a_i

begin
  $D = \emptyset$;
  $Var = \infty$;
  for *each* $x \in S$ **do**
    $S_1, S_2 = Split(S, f(x, a_i))$;
    $E(f(x, a_i), T; S) = \frac{|S_1|}{S} Ent(S_1) + \frac{|S_2|}{S} Ent(S_2)$;
    if $E(f(x, a_i), T; S) < Var$ **then**
      $SS_1 = S_1$;
      $SS_2 = S_2$;
      $T = f(x, a_i)$;
      $Var = E(f(x, a_i), T; S)$;
  $\Delta(f(x, a_i), T; S) = \log_2(3^k - 2) - [kEnt(S) - k_1 Ent(SS_1) - k_2 Ent(SS_2)]$;
  if $Gain(f(x, a_i), T; S) > \frac{\log_2(N-1)}{N} + \frac{\Delta(f(x, a_i), T; S)}{N}$ **then**
    return $D \cup RecursiveSplit(SS_1, a_i)$;
    return $D \cup RecursiveSplit(SS_2, a_i)$;
  else
    return S;
  return D;

3 RULE INDUCTION METHODS

Rule induction methods have been useful in several applications as medical diagnosis (TSUMOTO, 1998) and prediction systems (GOLAN; ZIARKO, 1995). Its diversity of applications led to several approaches being presented along the years. Some of the good qualities of these methods are their capacity to present interpretable results, which is desirable because it offers a better comprehension of the problem being worked. In this chapter, we present two of the most commonly used methods for inducing rules based on rough sets: *LERS* (Learning from Examples using Rough Sets) (GRZYMALA-BUSSE, 1997) and the algorithm proposed by M.Inuiguchi and T. Miyajima in (INUIGUCHI; MIYAJIMA, 2007) (we named it here *IM* approach). Both of these methods are suitable to work with uncertain and conflicting information, which make these methods useful for real world applications.

Rule induction methods aim to present a set of logical rules concluding the set of objects to which an object is part. By objects description, using inductive learning, it is possible to reach a general description of objects in each of the concepts. These objects may be cars, people or anything else that is desired to be classified. These methods can be applied in diagnosis systems to identify diseases through the symptoms presented by a patient and identify cars of different brands. The acquired information along this process allows us only to approximately identify examples from each concept, since not all objects from the same concept have the same description.

Knowledge in the form of rules induced by learning from information systems is easy for humans to comprehend. Besides that, such rules can be employed in the knowledge bases of rule-based expert systems to make inferences. One of the main advantages of inducing rules based on rough sets theory is that one does not need any preliminary information about uncertain, imprecise and inconsistent data (like prior probability in probability theory, grade of membership in fuzzy set theory).

3.1 Learning From Examples Using Rough Sets (*LERS*)

The rule induction system *LERS* (Learning from Examples using Rough Sets) described in (GRZYMALA-BUSSE, 1997), is a rough set based rule induction system aiming at producing logical rules describing objects that are part of a specific concept X. In order to accomplish this goal, *LERS* employs two algorithms: *LEM*1 and *LEM*2 (*LEM*1 and *LEM*2

stand for Learning from Examples Module version 1 and 2 respectively) (GRZYMALA-BUSSE, 1992). Both of these options produce minimal rules specifying a concept, but *LEM*2 is most used since it produces best results (GRZYMALA-BUSSE, 1992).

Given the concept $X \subseteq U$ we would like to learn from an information system $S = (U, C \cup D, V, f)$, *LERS* receives as input a description P of X; we consider P as being the lower $\underline{X}_C$ or upper $\overline{X}_C$ approximation of X with respect to C. It returns as output a minimal set $\mathscr{T}$, representing a nonempty collection of nonempty sets of attribute/value pairs describing X.

Now we proceed by defining the important notion of dependency. Let T be a set of attribute-value pairs of U, i.e., the elements of T are of the kind (c, v) such that $c \in C$ and $v = f(x, c)$ for some $x \in U$. By $[(c, v)]$ we mean the set $\{x \in P \mid f(x, c) = v\}$. From (GRZYMALA-BUSSE, 1997) we say set P depends on T if and only if

$$\emptyset \neq [T] = \bigcap_{t \in T} [t] \subseteq P.$$

Set T is named a *minimal complex* of P if and only if P depends on T and no proper subset T' of T exists such that P depends on T'. Thus $\mathscr{T}$ is a local covering of P if and only if the following conditions are satisfied:

- each member $T \in \mathscr{T}$ is a minimal complex of P,
- $\bigcap_{t \in \mathscr{T}} [T] = P$, and
- $\mathscr{T}$ is minimal, i.e., $\mathscr{T}$ has the smallest possible number of members.

In the sequel, we present the procedure *LEM*2 of *LERS* introduced in (GRZYMALA-BUSSE, 1992). Note it receives as input the set P (P is $\underline{X}_C$ or $\overline{X}_C$) and returns as output the set $\mathscr{T}$ representing the local covering of P. In Algorithm 3, it is possible to consider precedence of attributes (line 8). However, if this kind of information is not available in the information system, this step can be ignored.

The Algorithm 3 works as follows: In lines 6 to 11 we get a set T of attribute/value pairs whose corresponding $[T]$ is a subset of P, that is, in each iteration we add a new value pair t to T, turning $[T]$ more specific until it defines a proper subset of P. Observe from lines 12 to 14, the algorithm searches for value pairs t that are dispensable to define $[T]$. This guarantees that T is a minimal complex of P. Line 3 is true only when all objects in P are covered by some $T \in \mathscr{T}$. Line 17 to 19 search for dispensable minimal complex sets T of P. Thus, the resulting $\mathscr{T}$ will be minimal.

Recalling the information system presented in Example 3

Object (U)	Number of doors (q_1)	Horsepower (q_2)	Colour (q_3)	Make (q_4)
x_1	2	60	blue	Opel
x_2	2	100	black	Nissan
x_3	2	200	black	Ferrari
x_4	2	200	red	Ferrari
x_5	2	200	red	Opel
x_6	3	100	red	Opel
x_7	3	100	red	Opel
x_8	3	200	black	Ferrari
x_9	4	100	blue	Nissan
x_{10}	4	100	blue	Nissan

Table 3 – Datasets Description (Table 3 repeated from page 19)

Example 6. From Example 3. Suppose we want to induce rules to characterize the concept X_F ($Ferrari$). We can apply Algorithm 3 to induce rules from either $\underline{X_{FC}}$ or $\overline{X_{FC}}$. Making $P = \underline{X_{FC}} = \{x_3, x_8\}$ as the input of Algorithm 3, we obtain as output a certain rule describing a Ferrari:

$$(q_3, black) \wedge (q_2, 200) \rightarrow (q_4, Ferrari)$$

Making $P = \overline{X_{FC}} = \{x_3, s_4, x_5, x_8\}$ in Algorithm 3, we obtain as output a possible rule describing a Ferrari:

$$(q_2, 200) \rightarrow (q_4, Ferrari)$$

Thus, when considering lower approximation, for instance, if an object x satisfies $f(x, Horsepower) = 200$ and $f(x, Colour) = black$, we can understand that x satisfies decision class Ferrari.

From $\mathscr{T}$, one can easily build the corresponding rules (see Example 6). When the rules are induced from $\underline{X_C}$, they are named certain rules, while those rules induced from $\overline{X_C}$ are named possible rules.

As observed, the *LERS* method for inducing rules is a quite expensive computational method as ,during the rule construction, the process deciding which conditions will be added to the rule must verify among all examples the ones which satisfy the same condition. Thus, when working with large datasets, the computational time may be a limiting factor. We also

Algoritmo 3: LEM2

Input : P
Output : Returns a set of classification rules for P

begin
  $G = P$ $\mathscr{T} = \emptyset$
  while $G \neq \emptyset$ **do**
    $T = \emptyset$;
    $T(G) = \{t \mid [t] \cap G \neq \emptyset\}$;
    while $T = \emptyset$ *or* $[T] \not\subseteq P$ **do**
      select a pair $t \in T(G)$ *with the highest priority, if a tie occurs, select a pair* $t \in T(G)$ *such that* $\mid [t] \cup G \mid$ *is maximum; if another tie occurs, select a pair* $t \in T(G)$ *with the smaller cardinality of* $[t]$*; if a further tier occurs select first pair;*
      $T = T \cup \{t\}$;
      $G = [t] \cap G$;
      $T(G) = \{t \mid [t] \cap G \neq \emptyset\}$;
      $T(G) = T(G) - T$;
    for *each* $t \in T$ **do**
      if $[T - \{t\}] \subset P$ **then**
        $T = T - \{t\}$;
      $\mathscr{T} = \mathscr{T} \cup \{T\}$;
      $G = P - \cup_{T \in \mathscr{T}}[T]$;
  for *each* $T \in \mathscr{T}$ **do**
    if $\cup_{S \in \mathscr{T} - \{T\}}[S] = P$ **then**
      $\mathscr{T} = \mathscr{T} - \{T\}$;
  return $\mathscr{T}$

observe that, as this method uses equivalence classes, it is inappropriate to deal with numerical data, which reduces its applications to real world problems.

3.2 Masahiro Inuiguchi Approach (*IM*)

Another approach to inductively learn classification rules from an information system based on rough sets is presented by M. Inuiguchi and T. Miyajima in (INUIGUCHI; MIYAJIMA, 2007). Let us call it *IM* approach owing to the first letter of the author's surname. Given an information system $S = (U, C \cup D, V, f)$ and the concept $X \subseteq U$ we would like to learn, we resort to the two sets K^+ and K^- to select respectively positive and negative examples with respect to X. Various possibilities for K^+ and K^- were presented in (INUIGUCHI; MIYAJIMA, 2007). Here, we will consider two cases: $K^+ = \overline{X}_C$ and $K^- = \underline{X}_C$. As for K^-, we will assume $K^- = U - K^+$.

In their induction method, a discernibility matrix (SHAN; ZIARKO, 1995) is used

to identify attribute/value pairs discerning positive examples from those negative ones. Each discernibility matrix cell M_{ij} for K^+ and K^- is defined as the set

$$\{(c, f(x_i,c)) \mid f(x_i,c) \neq f(x_j,c)\}, x_i \in K^+, x_j \in K^-\}.$$

Regarding $(c, f(x_i,c))$ as an atomic formula (the value of attribute c is v), each matrix cell M_{ij} contains atomic formulae distinguishing a positive sample x_i from a negative sample x_j. Using this information, we can produce rules satisfied by positives samples, but unsatisfied by those negative ones. Considering a specific positive sample x_i and a negative one x_j, the rule differentiating x_i and x_j is given by

$$\bigvee M_{ij},$$

which is intended to mean the conjunction of all atomic formulas satisfied by x_i and unsatisfied by x_j. A rule differentiating a positive sample x_i from all negative ones is given by

$$\bigwedge_{j \in K^-} \bigvee M_{ij}.$$

Hence, the rule differentiating all positive sample from all negatives is

$$\bigvee_{i \in K^+} \bigwedge_{j \in K^-} \bigvee M_{ij}. \tag{3.1}$$

This formula is unsatisfied by all negative samples and is satisfied by at least one positive sample. Given as input the pair (K^+, K^-), this approach produces as output the formula exhibited in Equation (3.1) obtained from the associated discernibility matrix M_{ij}.

Example 7. Back to Example 3, let $K^+ = \{x_3, x_8\}$ and $K^- = \{x_1, x_2, x_4, x_5, x_6, x_7\}$ we obtain two certain rules describing Ferrari; one of them is shown below:

$$((q_2, 200) \vee (q_3, \mathit{black})) \wedge (q_2, 200) \wedge (q_3, \mathit{black}) \wedge ((q_1, 2) \vee (q_2, 200) \vee (q_3, \mathit{black}))) \rightarrow (q_4, \mathit{Ferrari})$$

Observe this rule can be simplified to $(q_2, 200) \wedge (q_3, \mathit{black}) \rightarrow (q_4, \mathit{Ferrari})$. Making $K^+ = \{x_3, x_4, x_5, x_8\}$ and $K^- = \{x_1, x_2, x_6, x_7\}$ we obtain four possible rules describing Ferrari; one of them is shown below:

$$(((q_2, 200) \vee (q_3, \mathit{black})) \wedge (q_2, 200) \wedge ((q_1, 2) \vee (q_2, 200) \vee (q_3, \mathit{black}))) \rightarrow (q_4, \mathit{Ferrari})$$

Also note that this can be simplified to $(q_2, 200) \rightarrow (q_4, \mathit{Ferrari})$. Thus, when considering lower approximation, for instance, if an object x satisfies $f(x, \mathit{Horsepower}) = 200$ and $f(x, \mathit{Colour}) = \mathit{black}$, we can understand that x satisfies decision class Ferrari. Note both approaches have obtained the same result, but it is a mere coincidence. Also note that as *LERS*, this method compares the attributes of the objects to discern between objects of different classes, thus it is also not appropriate to work with numerical information.

As can be seen in this chapter, both of induction methods make comparisons between objects. These comparisons are performed with the objective of identifying patterns that can be used to identify objects belonging to different sets. Comparisons like these performed in LERS and IM can be easily performed when working with categorical data, however, when we need to work with numerical data, due to the large number of combinations of values that an object can assume for its attributes, it is unlikely that two or more objects can be identified by the same pattern identified from these comparisons. We can also emphasize that both induction methods generate as results rules capable of identifying objects of only one set of objects, so they are better suited for binary classification problems. The limitations, solutions and their challenges are discussed in more detail in the next chapter.

4 RULE INDUCTION FROM NUMERICAL INFORMATION SYSTEMS

In this chapter we highlight some of the problems faced by the methods presented in the previous chapter and propose three approaches to deal with these problems. In the first approach (Section 4.1), we resort to the discretization method (MDLP) presented in (Section 2.4), as a pre-processing step before the application of the induction method. In the second approach (Section 4.2), we propose the use of belief merging to reduce the number of cases rejected in the first approach. The third approach (Section 4.3), use a decision tree to classify completely the rejected objects.

Some of the advantages of the presented induction methods are their ability to deal with incomplete, conflicting and ambiguous data through approximation processes. However, these methods are not appropriate to work with numeric data efficiently, since during the comparison process between two objects x and y, any difference between two numerical values describing x and y causes them to be in different decision classes, even when they satisfy the same decision value. Thus, the realization of the approximation of sets by equivalence relations is impracticable, since it is highly unlikely to find two objects with the exact same description. In this situation, each equivalence class would consist of only one object.

Object (U)	Height (q_1)	Weight (q_2)	Age (q_3)	Male (q_4)
x_1	151.7	47.8	63	1
x_2	139.7	36.4	63	0
x_3	136.5	31.8	65	0
x_4	156.8	53.0	41	1
x_5	154.4	41.2	51	0
x_6	163.8	62.9	35	1
x_7	149.2	38.2	32	0

Table 2 – Numerical Datasets Description (Table 2 repeated from page 14)

Example 8. Using equivalence classes to group objects from Table 2, we have

$$[x_1]_C = \{x_1\} \qquad [x_5]_C = \{x_5\} \tag{4.1}$$

$$[x_2]_C = \{x_2\} \qquad [x_6]_C = \{x_6\} \tag{4.2}$$

$$[x_3]_C = \{x_3\} \qquad [x_7]_C = \{x_7\} \tag{4.3}$$

$$[x_4]_C = \{x_4\} \tag{4.4}$$

as a result, all equivalence classes is composed by only one element.

4.1 Standard Approach

To overcome this difficulty and make induction methods based on rough sets suitable to numerical data, we propose the use for discretization methods as a prepossessing step prior to the application of induction methods. In this work, we resort to MDLP (Minimum Description Length Principle) (RISSANEN, 2014), a well known discretization method which is commonly used in classification methods as can be seen in (LINDEBERG; LI, 1997), (EVANS *et al.*, 2004) and (BEGUM *et al.*, 2013).

As a solution to this limitation, we present Algorithm 4. In this algorithm, the discretization method MDLP is applied over the information system S given as input. After that, an approximation (Lower or Upper) is computed for each set of objects satisfying a decision value. Through these approximations, a set of rules identifying objects satisfying the same decision value is produced.

Algoritmo 4: Rule Induction with MDLP

Input : $S = (U, C \cup D, V, f)$
Output : Returns a set classification rules

```
1 begin
2     result = ∅;
3     data = MDLP(S);
4     V = V_d(where D = {d});
5     for each v ∈ V do
6         X = {x | f(x,d) = v};
7         P = Approximation(X);
8         result = result ∪ Induction(P);
9     return result;
```

Example 9. Back to Example 3, suppose we want to classify the car object x_4 which has a description $(q_1, 2)$, $(q_2, 200)$, (q_3, red) and $(q_4, Ferrari)$. The available rules obtained as a result from Algorithm 4 for classification are presented in the sequel. Rules $F1$ and $F2$ classify objects to class $Ferrari$, rules $O1$, $O2$ and $O3$ classifies objects to $Opel$, and rules $N1$ and $N2$ classifies objects to class $Nissan$.

$F1 : (q_1, 2) \wedge (q_2, 200) \rightarrow (q_4, Ferrari)$

$F2 : (q_2, 200) \wedge (q_3, black) \rightarrow (q_4, Ferrari)$

$O1 : (q_1, 2) \wedge (q_3, red) \rightarrow (q_4, Opel)$

$O2 : (q_1, 3) \wedge (q_2, 100) \rightarrow (q_4, Opel)$

$O3 : (q_2, 60) \rightarrow (q_4, Opel)$

$N1 : (q_1, 4) \rightarrow (q_4, Nissan)$

$N2 : (q_2, 100) \wedge (q_3, black) \rightarrow (q_4, Nissan)$

Note, Algorithm 4 will not always be able to classify all objects. In some cases, an object may not satisfy any rule or does even satisfy rules of different decision classes, which is the case of object x_4 that satisfies $F1$ and $O1$. The objects that fit into these situations are called rejected. In this work, we consider the possibility of rejecting these cases, which is known as a rejection option and is widely used in critical systems. However, the presence of rejected cases negatively influences the performance of the induction methods, as the presented method is not able to classify these cases.

Definition 9 (Accuracy)**.** Let U_{TE} be the set of objects to be classified and C_{TE} the set of objects from U_{TE} which are correctly classified. The accuracy of the classification denoted as Ac, is defined as $Ac = \frac{|C_{TE}|}{|U_{TE}|}$.

We must also observe that, because we are using discretization methods, it is unavoidable to lose information/accuracy, which may lead to an increasing number of ambiguous and conflicting examples, thus, resulting in additional rejected cases.

4.2 Reviewed Approach

To deal with the rejected objects, we propose the use of the concept of partial satisfiability, which is used to verify the degree to which an object satisfies a rule. Thus, the partial satisfiability degree can be used to verify the degree to which an object satisfies each rule, thus, even if the object is a rejected one, it can be classified by the rule that presents the highest degree of satisfaction for the object we wish to classify.

Definition 10 (Partial satisfiability)**.** (KAREEM *et al.*, 2017) Let $\mathscr{L}$ be a language of propositional logic and $\kappa \in \mathscr{L}$, the antecedent of a classification rule, $\kappa = C_1 \wedge \ldots \wedge C_n$. In κ, every C_i can be a literal or disjunctive formula. The partial satisfiability of C_i for x, denoted as $x_{ps}(C_i)$ is equal to 1 if C_i is satisfied by x, otherwise, it is equal to 0. The partial satisfiability of κ for an object x, denoted as $x_{ps}(\kappa)$, is defined as $x_{ps}(\kappa) = \sum_{i=1}^{n} \frac{x_{ps}(C_i)}{n}$.

In the disambiguating process presented in Algorithm 5, all classification rules have its satisfiability degree verified for an object in the set of rejected objects.

Algoritmo 5: Disambiguation Process

```
   Input  : obj, ACRules
   Output : Returns a possible class for rejected case obj
1  begin
2      SS = ∅, MaxV = ∅;
3      for each SetRule ∈ ACRules do
4          SCI = ∅;
5          for each Rule ∈ SetRule do
6              SCI = SCI ∪ GetPartialSatisfaction(Rule, obj);
7          SS = SS ∪ SCI;
8          MaxV = MaxV ∪ GetMaxValue(SCI);
9      Max = GetMaxValue(MaxV);
10     Cnt = Count(Max, MaxV);
11     while Cnt > 1 do
12         MaxV = ∅;
13         GrSet = ∅;
14         for each SCI ∈ SS do
15             if Max = GetMaxValue(SCI) then
16                 GrSet = GrSet ∪ SCI;
17         for each SCI ∈ GrSet do
18             if Size(SCI) > 1 then
19                 RemoveMax(SCI);
20             else
21                 Remove(SCI);
22         if Size(SCI) > 1 then
23             for each SCI ∈ GrSet do
24                 MaxV = MaxV ∪ GetMaxValue(SCI);
25             Max = GetMaxValue(MaxV);
26             Cnt = Count(Max, MaxV);
27         else
28             return Rejected;
29     return ClassID(SCI, Max);
```

Now we present a brief explanation of how Algorithm 5 works. From line 3 to 7 we get the satisfaction degree of each rule in *SetRule* (line 5), which contains all classification rules of a class and store it in *SCI*. This process is repeated to each class (line 3). At line 8, we get the highest value from *SCI* and store it in set *MaxV* (line 8). At line 9 we get the highest value in *MaxV* and count how many times it appears in *MaxV*. If the condition in line 11 is false, the

disambiguation process is completed and the class ID associated with the set containing such value is returned. On the other hand, if the condition is true, we proceed with the disambiguation process. From line 14 to 21, we remove every set SCI not containing value Max from SS. We also remove the Max value from the remaining sets. From 22 to 26 we update the Max value and verify how many sets contain it (line 25, 26). If just one set contains the value Max, the process is finished; otherwise, if more than one set contains it, the process continues from line 12.

Example 10. Continuing Example 9, we note that x_4 satisfies both rules $F1$ and $O2$, so it is a rejected case. In order to classify this case, we resort to the disambiguation method, which makes use of satisfiability degree to classification rules. During the disambiguation process, we obtain the arrays $F : [1, 1/2]$, $O : [1, 0, 0]$ and $N : [0, 0]$ showing the satisfiability degree for rules from classes $Ferrari$, $Opel$ and $Nissan$ respectively. As we can observe, both F and O have value 1 which delivers us to a tie, however the second greatest value in F is $1/2$ and in O it is 0, so we select class $Ferrari$ as the class for object x_4.

Given an object x_i, the satisfiability degree of all rules classifying to the same decision class are grouped into a multiset. Thus, we have a multiset for each of the decision classes. These multisets can be used to verify to which class belongs the rule with the highest degree of satisfitability and in this way classify the object x_i. However, this method does not guarantee that all objects can be classified. In the case of a tie, we consider the second highest value of each multiset, from these values, we select the one that has the highest value. In the case of successive draws, the same process is repeated considering the next largest value in each multiset. However, if successive draws occur between two arrays of different sizes, we perform the tie-breaking process until a multiset is selected or until the smallest multiset does not have a next largest value, in this case, we select class related to the rules in the largest multiset. If successive draws occur between multisets of the same size, we repeat the tie breaking process until a decision value is established, or until neither multiset has a next higher value. If using this process it was not possible to define a decision value, we consider that the object remains ambiguous. In this process, only the tied classes are considered.

Observing that, the disambiguation process give us as result only the classification value for a rejected object and none additional information that could be used to explain the reason behind the result, we say the disambiguation process is not interpretable. Thus, the association of classification methods as $LERS$ and IM, which are completely interpretable, with the disambiguation process, results in a semi-interpretable classification method. In short, we

say that a classification method is semi-interpretable when the classification approach alternates between logical rules (which are interpretable) and a mathematical approach (not interpretable). Although the resulting approach is not able to classify all objects, its classification power is higher than the standard approach of *LERS* and *IM*, which produce a significantly higher number of rejected objects. The reason for this difference in accuracy is explained by the number of rejected objects, which is significantly higher in standard approaches than when using the disambiguation process.

The characteristic of being semi-interpretable brings an innovative aspect for classification methods, as these methods produce interpretable results while preserving the classification power obtained by using the disambiguation process. When compared to decision tree classification methods, which are completely interpretable and always provides a classification result, an approach with the characteristic of leaving some objects as unclassified while classifying others may be better suitable for critical systems as a medical diagnostic system, as a wrong classification may result in irreparable damage. Therefore, as well as standard approaches for LERS and IM it still necessary a specialist to perform the classification for more complex cases.

It is also notable the advantage of being able to produce interpretable results when compared to methods like Neural Networks (NNs), Support Vector Machine (SVM) and others that are usually called black boxes, as observed in (CORTEZ; EMBRECHTS, 2013). Although some times these methods are said to be not interpretable, in the recent years many works have been presented in order to extract knowledge from SVM (MARTIN-BARRAGAN *et al.*, 2014), Neural Networks (CRAVEN; SHAVLIK, 1994) and others. However, the results obtained when using rough set based methods are directly and easily interpretable.

Given these observations and the results shown in Section 5.1, it is visible the improvement obtained by using the partial satisfiability measure instead of either *LERS* or *IM*. Anyway, in the next section, we will focus on an alternative approach, which resorts to binary classification trees to classify rejected objects. Its advantage is it eliminates every rejected object, and all results are interpretable.

4.3 Complete Approach

Although the number of rejected cases has decreased by the application of the disambiguation method, some rejected cases still persist. Although the rejection option has some advantages such as minimizing risks, it is not always desired. In some problems, the cost of not

responding may be greater than that of responding wrong. It may be particularly true for low risk problems or well known problems.

We know an object may be rejected in two situations: when it does not satisfy any of the decision rules or when it satisfies decision rules of different decision classes. Objects in the first situation are rejected because of the lack of rules to identify some data patterns. Such rejected cases are not uncommon, as datasets used in the induction process may not contain examples of all data patterns. Objects in the second situation are rejected due to the low restrictive power of some rules, as satisfied rules do not exclude the possibility of rules classifying objects to different classes from being satisfied, thus generating a conflict between satisfying rules.

Definition 11 (Conflicting rules)**.** Let $Ru_i \in R_i$ and $Ru_j \in R_j$ be two different rules classifying objects to different decision classes. Also, let C_{Ru_i} and C_{Ru_j} be its respective sets of conditional attributes. We say that Ru_i and Ru_j are conflicting with each other in the following situations.

$$CaseI : C_{Ru_i} \subseteq C_{Ru_j} \text{ or } C_{Ru_j} \subseteq C_{Ru_i}$$
$$CaseII : C_{Ru_i} - C_{Ru_j} \neq \emptyset \text{ or } C_{Ru_j} - C_{Ru_i} \neq \emptyset$$

To solve such classification problems, we resort to decision trees to classify rejected cases. The classification process of the rejected cases are performed during the test phase (on demand). Thus, after trying to classify a rejected case through rules obtained by *LERS* and *IM*, the rejected case is classified by the decision tree. This process is described in Algorithm 6.

Algoritmo 6: Complete Approach

```
   Input  : Rejected object x
   Output : Decision value for x
1  begin
2      Count = 0;
3      for each R_i ∈ R do
4          for each r_j ∈ R_i do
5              if x_p s(r_j) = 1 then
6                  Val = Conclusion(r_j);
7                  Count = Count + 1;
8      if Count = 0 or Count > 1 then
9          Val = ClassifyTree(x);
10     return Val;
```

Although this approach does not show the exact conditions satisfied by a rejected case x. Using the set R_{Tree} of rules composing the decision tree and the set R of rules given by

LERS and *IM*, it is possible to produce new rules that classify cases rejected due to conflicting rules.

Backing to Example 9, we note that rules $F1$ and $O1$ are conflicting, as *CaseII* in Definition 11 is satisfied. Once a conflict between Ru_i and Ru_j is identified, a new rule $Ru_{i,j}$, is produced by combining the conditions of Ru_i and Ru_j. The new resulting rule $Ru_{i,j}$, is a conjunctive rule whose conclusion is equal to the conclusion of Ru_i or Ru_j, therefore is undetermined. More details are given in Example 11.

Example 11. As an example, we show two conflicting rules $F1$ and $O1$. Note that both $F1$ and $O1$ can be simultaneously satisfied.

$$F1 : (q_1, 2) \wedge (q_2, 200) \rightarrow (q_4, Ferrari)$$
$$O1 : (q_1, 2) \wedge (q_3, red) \rightarrow (q_4, Opel)$$

The resulting rule $Ru_{F1,O1}$ obtained through $F1$ and $O1$ presents an undetermined conclusion. The resulting rule for this case is as follows

$$Ru_{F1,O1} = (q_1, 2) \wedge (q_2, 200) \wedge (q_3, red) \rightarrow ((q_4, Ferrari) \vee (q_4, Opel)).$$

Note that all possible patterns simultaneously satisfying rules classifying objects to different decision classes are caught by these new rules. In our approach, to classify rejected cases, rules induced by decision trees are used to turn these new built rules more restrictive and assign a decision value to it.

Let R_{Tree} and R_{Con}, be the set of rules composing a decision tree and the set of rules $R_{i,j}$ respectively. After rules composing these sets are known, a new rule for each pair of compatible rules in $R_{Tree} \times R_{Con}$ is produced. Let $Rut_i \in R_{Tree}$ and $Ruc_i \in R_{Con}$, we say that Rut_i and Ruc_i are compatible if both of them can be simultaneously satisfied. Two rules can be simultaneously satisfied if one of the cases presented in Definition 11 is satisfied.

The new rules generated by using Rut_i and Ruc_i are produced by adding to Ruc_i, conditions that are in Rut_i and missing in Ruc_i. The set of new rules produced in this way is used to classify all objects that would be rejected when rules concluding different decision values is satisfied.

Example 12. Rules obtained as a result from decision tree when given as input the information system presented in Example 3 are

$T1 : (q_2, 200) \rightarrow (q_4, Ferrari)$

$T2 : \neg(q_2, 200) \wedge (q_1, 3) \rightarrow (q_4, Opel)$

$T3 : \neg(q_2, 200) \wedge \neg(q_1, 3) \wedge (q_2, 60) \rightarrow (q_4, Opel)$

$T4 : \neg(q_2, 200) \wedge \neg(q_1, 3) \wedge \neg(q_2, 60) \rightarrow (q_4, Nissan)$.

The only rule in $R_{Tree} = \{T1, T2, T3, T4\}$ that is compatible with $Ru_{F1,O1}$ is $T1$, as the first situation of Definition 11 is satisfied. Thus, the chosen decision value for $Ru_{F1,O1}$ obtained in Example 11 is $(q_4, Ferrari)$.

During the classification process, all objects are first tested using these new set of rules. If none of these rules classify one of the cases, this case is tested using the rules previously obtained from *LERS* or *IM* approach. At least, if one case is rejected in one of these two phases, it is submitted to the decision tree. We would like to emphasize that this approach is equivalent to Algorithm 6. As both of then produce the same result.

This approach classifies all cases that would be rejected by the rules provided by *LERS* and *IM*. In the next chapter, we will show the addition of the new set of rules to overcome the results obtained by the decision trees in most of the datasets when using the lower approximation. Such improvement gains evidence that the rough sets based rule induction methods classify accordingly some additional cases when compared to classification decision trees.

As shown in this chapter, rough sets based rule induction methods have some limitations. Some of the limitations of these methods are their inability to work with numerical data and perform classification in multiclass problems (only binary classification problems). To address these difficulties we use discretization methods such as the MDLP to make it possible to apply these induction methods to numerical data.

Also, to apply these methods to multiclass classification problems we have chosen to induce different sets of rules and we have adopted the rejection option when it is not possible to categorically classify an object according to the available rules. To classify the rejected cases two approaches were presented: reviewed approach (Section 4.2), which makes use of partial satisfiability measure and the complete approach (Section 4.3), which uses decision trees. As the reviewed approach is not interpretable, we say that this is a semi-interpretable approach, this approach preserves the rejection option. Unlike the reviewed approach, the complete approach is able to classify all objects by a logical rule, for this reason, we say that this is a completely interpretable approach, so it does not present the rejection option.

Each of the approaches presented to classify rejected cases may be more indicated in different cases, as the approach using partial satisfaction has the option of rejection, this approach would be more suitable for critical systems. However, if the problem is well known and does not present great risks, the approach using decision trees is the most appropriate one.

5 EXPERIMENTS

In this section, we consider the proceedings used through the performance evaluation of the rule induction methods. In our experiments, we resort to two approaches: one using lower approximation and another one using upper approximation. In all performed tests the discretization method MDLP (RISSANEN, 1978) was applied as a preprocessing step before the application of the induction methods. To carry out the experiments, we used seven numerical datasets listed in Table 4 with their respective number of objects, number of attributes and number of concepts. Five of these datasets are real datasets, they are: Iris, Diabetes, Wine, Sonar and Glass. Almost all of them can be found in the Machine Learning Repository at the University of California (LICHMAN, 2013). The exception is Diabetes from the Pima Indians (LAMAHAMADEH, 2013). The Test-I is a synthetic and completely consistent dataset. Using this dataset two experiments were performed, one using it without any modification and other after making 5% of it inconsistent, which we called Test-II. This synthetic dataset is used to visualize the impact of inconsistent information in the performance of the methods used.

In our experiments, each dataset has been split into two disjoint sets as follows: 70% for training and 30% for testing. Four variations of Algorithm 4 (*LERS* with lower approximation, *LERS* with upper approximation, *IM* with lower approximation and *IM* with upper approximation) have been trained/tested. For each variant, we executed the experiments one hundred times and took the average of each combination. The result of each execution consists of the percentage of objects from testing set correctly classified to the approach quoted. In each execution, a new pair of training/testing data was generated. To the evaluation of the results we considered two approaches named standard, which has the rejection option; reviewed, which tries to classify the rejected objects and the complete approach, which was described in (Section 4.3).

The datasets used here were chosen because they are well known in the literature and are commonly used for performance evaluation of classification systems. Also, the number of executions (one hundred) is performed in order to minimize the influence of test cases that favors a good or a bad result. Thus, running the experiments a sufficiently large number of times is enough to provide us a better notion of how the method behaves in the general case.

Definition 12 (Classification acuracy)**.** Let U_{TE} be the test set and U_{CR} the set of objects correctly classified. The accuracy of the classification methods denoted as A_C is defined as $A_C = \frac{|U_{CR}|}{|U_{TE}|}$.

In the standard approach, each test case can be evaluated by three strategies. The evaluation strategies to the standard approach are named pessimistic, optimistic and safe. To precisely specify these approaches, following we present a formal definition for ambiguous objects.

Definition 13 (Rejected objects)**.** Let $R = \{R_1, R_2, \ldots, R_n\}$ the set of classification rules and $R_i \subseteq R$ the set of rules identifying objects satisfying the same decision value of an object x. Also, let $r_k \in R_i$ be one of these rules. The set of rejected objects is defined as

$$\{x | \nexists r_k \in R_i, x_{ps}(r_k) = 1\} \cup \{x | \exists r_k \exists r_l (r_k \in R_i, r_l \in R_j), i \neq j, x_{ps}(r_k) = x_{ps}(r_l) = 1\}.$$

The classification process using the pessimistic, optimistic and safe approach consider as correctly classified only the following set of objects:

- Pessimistic approach: $\{x | \exists r_k \in R_i, x_{ps}(r_k) = 1 \wedge \forall r_k \in \overline{R_i}, x_{ps}(r_k) = 0\}$.
- Optimistic approach: $\{x | \exists r_k \in R_i, x_{ps}(r_k) = 1\}$

At least, the safe approach does not consider rejected objects in accuracy calculation. Thus, every rejected example is excluded from the testing set. Let U_{TE} be the test set and U_{DU} the set of rejected objects presented in Definition 13. The set of objects used to measure the accuracy of the classification in the safe approach is $\{x | x \in U_{TE} \wedge x \notin U_{DU}\}$.

The reviewed approach is similar to the pessimistic approach, as it also applies pessimistic, safe and optimistic approaches. However, differently from the standard approach, we use the partial satisfiability measure, that is used to classify as many rejected objects as possible. This approach also considers the rejection option for unclassified objects. However, although its classification power is greater when compared to the previously presented approaches, the obtained results from partial satisfiability are not interpretable. Thus, we say the reviewed approach is a semi-interpretable approach.

Now we present the results for each approach used in Algorithm 4. In Tables 5, 7 and 9 (resp. Tables 6, 8 and 10) results when using lower approximation (resp. upper approximation) are presented. For each table, results involving the accuracy defined in 12 (followed by the respective standard deviation) and the percentage of objects rejected for each variant of the algorithm proposed are presented. We would like to highlight that during these tests, classification rules were induced even when the datasets have only two decision values. In this way, when it is

said that an object does not belong to a class, this does not mean that the same object belongs to the remaining class.

	cases	number of attributes	concepts
Iris	150	4	3
Diabetes	768	8	2
Wine	178	13	3
Sonar	208	61	2
Glass	214	10	6
Test-I	100	4	3

Table 4 – Datasets Description

5.1 Results: Standard Approach

	LERS	Rejected %	*IM*	Rejected %
Iris	**87.7** ± 2.6	9.1 ±5.5	58.4 ±6.3	37.8 ±6.6
Diabetes	39.3 ±3.1	50.9 ±3.7	**44.2** ± 3.1	46.1 ±3.5
Wine	**89.5** ± 1.9	9.1 ±4.7	48.5 ±5.5	39.4 ±6.1
Sonar	**62.3** ± 5.1	27.7 ±6.2	11.4 ±3.5	86.5 ±3.7
Glass	**33.5** ± 6.9	58.8 ±6.9	19.8 ±5.2	69.9 ±6.8
Test-I	**97.4** ± 1.5	1.6 ±2.4	65.0 ±6.6	32.8 ±6.4
Test-II	**94.3** ± 1.3	5.4 ±3.3	62.8 ±6.9	35.1 ±7.5

Table 5 – Algorithm 4 with Lower Approximation and Pessimistic Approach

	LERS	Rejected %	*IM*	Rejected %
Iris	**92.4** ± 2.2	4.6 ±3.6	65.6 ±5.5	29.9 ±5.9
Diabetes	31.7 ±3.5	63.1 ±4.0	**39.1** ± 2.6	53.5 ±3.5
Wine	**88.6** ± 1.5	10.7 ±4.2	46.4 ±6.2	45.7 ±6.7
Sonar	**61.1** ± 4.7	26.9 ±5.3	11.5 ±3.6	86.4 ±3.7
Glass	**31.9** ± 6.9	60.1 ±7.3	23.7 ±5.3	62.6 ±7.9
Test-I	**97.9** ± 1.5	1.0 ±2.2	65.6 ±7.5	32.8 ±7.5
Test-II	**94.4** ± 2.6	4.7 ±2.9	63.2 ±7.4	34.4 ±7.5

Table 6 – Algorithm 4 with Upper Approximation and Pessimistic Approach

As can be observed, in general *LERS* approach gives us better results for both lower and upper approximation when compared to *IM* approach, the exceptions are the results for dataset Diabetes in when using the lower approximation. One of the reasons for the difference between performance is the number of rejected objects produced by each approach, which in general is higher in the *IM* approach than in *LERS*. The exception is in Diabetes when using the

lower approximation, which is the only occasions where *IM* accuracy outperforms *LERS*.

It is also possible to note that both approaches present a significantly lower accuracy for Sonar when using *IM* in both lower/upper approximation. The reason behind it becomes clear when looking at the respective results in Table 11. By these observations, we conclude that these results are due to the low discriminatory power of obtained rules, which results in a high number of rules from different classes being satisfied by the same objects. This result makes clear that in a binary classification problem a negative answer to rules from a class does not mean a positive answer to the rules of the other class.

	LERS	**Rejected %**	***IM***	**Rejected %**
Iris	**96.5** ± 4.9	9.1 ±5.5	94.1 ±3.7	37.8 ±6.6
Diabetes	80.2 ±2.8	50.9 ±3.7	**82.1** ± 3.6	46.1 ±3.5
Wine	**89.5** ± 1.9	9.1 ±4.7	80.1 ±6.3	39.4 ±6.1
Sonar	**86.3** ± 6.1	27.7 ±6.2	85.0 ±9.3	86.5 ±3.7
Glass	**81.8** ± 5.7	58.8 ±6.9	66.4 ±10.0	69.9 ±6.8
Test-I	**99.0** ± 2.8	1.6 ±2.4	96.8 ±4.4	32.8 ±6.4
Test-II	**99.8** ± 3.3	5.4 ±3.3	97.1 ±4.0	35.1 ±7.5

Table 7 – Algorithm 4 with Lower Approximation and Safe Approach

	LERS	**Rejected %**	***IM***	**Rejected %**
Iris	**96.9** ± 3.6	4.6 ±3.6	93.8 ±3.6	29.9 ±5.9
Diabetes	**86.2** ± 3.0	63.1 ±4.0	84.3 ±3.2	53.5 ±3.5
Wine	**99.2** ± 4.3	10.7 ±4.2	85.5 ±5.7	45.7 ±6.7
Sonar	83.7 ±5.4	26.9 ±5.3	**85.2** ± 11.6	86.4 ±3.7
Glass	**80.5** ± 5.7	60.1 ±7.3	64.1 ±9.8	62.6 ±7.9
Test-I	**99.0** ± 2.6	1.0 ±2.2	97.7 ±3.4	32.8 ±7.5
Test-II	**99.1** ± 3.4	4.7 ±2.9	96.5 ±4.3	34.4 ±7.5

Table 8 – Algorithm 4 with Upper Approximation and Safe Approach

From the observation of the results presented in Table 7 and Table 8 it is possible to conclude that the results presented by *LERS* are similar for both lower and upper approximation, as well as results presented by *IM* approach. It is also possible to observe that Safe approach preset better results than the pessimistic approach, as rejected objects are not considered in accuracy calculation.

	LERS	Rejected %	*IM*	Rejected %
Iris	**88.4** ± 4.5	8.4 ±5.0	60.1 ±6.1	36.1 ±6.5
Diabetes	41.0 ±2.8	49.1 ±3.7	**51.0** ± 3.0	39.2 ±3.3
Wine	**94.6** ± 3.5	3.9 ±3.0	72.1 ±5.4	15.7 ±4.9
Sonar	80.3 ±5.0	9.7 ±4.2	**98.0** ± 1.2	0.0 ±0.0
Glass	**36.5** ± 5.6	55.8 ±6.8	20.7 ±5.2	69.1 ±6.7
Test-I	**98.8** ± 1.8	0.2 ±1.1	65.9 ±6.7	31.8 ±6.3
Test-II	**95.3** ± 3.2	4.4 ±3.2	62.8 ±6.9	35.1 ±7.5

Table 9 – Algorithm 4 with Lower Approximation and Optimistic Approach

	LERS	Rejected %	*IM*	Rejected %
Iris	**96.7** ± 3.2	0.3 ±1.9	65.6 ±5.5	29.9 ±5.9
Diabetes	**94.5** ± 1.7	0.3 ±0.4	92.6 ±1.8	0.0 ±0.0
Wine	**93.7** ± 4.0	5.5 ±3.7	74.8 ±5.3	17.3 ±4.7
Sonar	80.3 ±4.6	7.7 ±3.4	**98.0** ± 1.3	0.0 ±0.0
Glass	**85.8** ± 3.9	6.2 ±2.8	63.1 ±4.7	23.1 ±5.4
Test-I	**99.0** ± 1.5	0.0 ±0.6	66.6 ±7.2	31.8 ±7.2
Test-II	**97.1** ± 3.2	2.0 ±2.2	66.4 ±7.2	31.2 ±7.1

Table 10 – Algorithm 4 with Upper Approximation and Optimistic Approach

In the Optimistic approach, it is expected that an object satisfies at least one condition of classification rules corresponding to its corresponding class. However, from Tables 9, we observe that for some datasets as Diabetes and Glass, the performance of the classification methods is relatively poor. This behavior can be explained by a relatively high number of rejected objects, that are satisfying rules classifying objects to different classes.

5.2 Results: Reviewed Approach

	LERS	Rejected %	*IM*	Rejected %
Iris	**94.6** ± 2.8	2.1 ±1.9	93.2 ±2.7	3.0 ±2.3
Diabetes	72.2 ±2.7	17.9 ±2.6	**72.6** ± 2.8	17.7 ±2.3
Wine	**95.6** ± 3.2	3.0 ±2.6	84.4 ±5.5	3.4 ±3.5
Sonar	79.7 ±5.3	10.3 ±4.2	**83.3** ± 4.0	14.7 ±4.1
Glass	58.2 ±8.2	34.1 ±8.4	**65.9** ± 7.1	23.9 ±6.1
Test-I	**97.4** ± 2.8	0.7 ±6.1	97.2 ±3.1	0.5 ±1.2
Test-II	**95.7** ± 3.2	4.0 ±3.2	94.9 ±3.6	3.1 ±2.5

Table 11 – Algorithm 4 with Lower Approximation and Reviewed Approach

	LERS	Rejected %	*IM*	Rejected %
Iris	**95.2** ± 3.0	1.8 ±2.4	94.5 ±2.6	1.0 ±1.6
Diabetes	69.8 ±4.1	24.9 ±3.9	**75.8** ± 2.4	16.7 ±2.1
Wine	**95.7** ± 3.3	3.5 ±3.0	89.0 ± 4.0	3.1 ±2.8
Sonar	78.8 ±4.3	9.2 ±3.2	**82.7** ± 4.6	15.3 ±4.5
Glass	55.2 ±7.6	36.7 ±8.1	**64.3** ± 6.9	21.9 ±7.3
Test-I	97.9 ±2.6	1.0 ±2.2	**98.0** ± 2.6	0.3 ±1.0
Test-II	**94.9** ± 3.1	4.2 ±2.5	94.5 ±3.4	3.1 ±2.5

Table 12 – Algorithm 4 with Upper Approximation and Reviewed Approach

From results presented in Table 11 and Table 12 we highlight that the accuracy presented for Algorithm 4 when using both lower and upper approximation is better than the results presented by pessimistic approach and similar to some results presented by safe approach. The reasons behind the improvement in results obtained using the reviewed approach are explained by the use of partial satisfiability measure, which is used to classify rejected objects. The influence of the use of partial satisfiability is also observed in the relatively low number of rejected objects in comparison to the other approaches.

Through the observed results, the impact of the partial satisfiability is remarkable, once it provides an improvement in accuracy for both *LERS* and *IM* approaches in both lower and upper approximation, as the number of rejected objects is low when compared to the other approaches.

5.3 Results: Complete Approach

To establish a comparison between conventional rule induction methods that also provide interpretable results, we provide results about the performance of both classification decision trees (SAFAVIAN; LANDGREBE, 1991) and BART decision tree (RISSANEN, 1978). We also show results referring to the approach exhibited in Section 4.3. In the following tables, the accuracy in percentage and standard deviation are displayed for each of the approaches used. As well as in the previous experiments, we ran the experiments one hundred times and took the average.

	Decision Tree	BART Decision Tree
Iris	**95.1** ± 2.7	93.4 ±3.3
Diabetes	72.5 ±2.5	**76.2** ± 2.2
Wine	93.8 ±2.7	**95.35** ± 3.2
Sonar	77.6 ±4.8	**79.8** ± 4.9
Glass	**73.6** ± 5.4	52.9 ±5.4
Test-I	99.1 ±1.4	**99.5** ± 1.1
Test-II	95.2 ±2.9	**96.5** ± 3.2

Table 13 – Base line

	LOWER		UPPER	
	LERS	***IM***	***LERS***	***IM***
Iris	**94.5** ± 2.6	94.1 ±2.7	**96.4** ± 2.2	94.5 ±2.6
Diabetes	72.6 ±2.6	**73.1** ± 2.3	**76.0** ± 2.7	73.7 ±2.2
Wine	**96.4** ± 2.5	83.7 ±5.2	**96.9** ± 2.4	87.5 ±3.8
Sonar	**80.5** ± 4.6	78.0 ±4.4	**78.8** ± 4.6	78.2 ±4.6
Glass	**75.4** ± 5.3	73.7 ±5.6	**75.0** ± 4.7	70.2 ±5.5
Test-I	**98.8** ± 1.8	97.7 ±3.0	**99.0** ± 1.5	98.4 ±2.3
Test-II	**95.6** ± 2.9	94.0 ±3.5	**95.3** ± 3.3	94.2 ±3.2

Table 14 – Complete approach

From the results in Table 14, we note a performance gain is obtained by our approach (Section 4.3) when compared to previous approaches (Standard and Reviewed). Besides that, it is important to note that this new approach does not provide a rejection option. Although it does not present the rejection option, its results are similar to the results obtained when using the Safe approach.

We also observe a performance gain in both *LERS* and *IM* when comparing results in Tables 13 and 14. Such improvement obtained by using decision tree leads to surpass the performance of the decision tree in most cases and also get close to the performance of BART decision tree.

5.4 Discussion

From our results, we see that when using the standard approach with lower approximation/MDLP, in general, we obtain slightly better results than with upper approximation/MDLP. A possible justification is when using lower approximation, we are taking less ambiguous cases; thus, the resulting rules will be unsatisfied by a greater number of cases from the other classes and satisfied by a greater number of cases from its own class.

Both safe and reviewed approach produces similar results (the only exception is the Glass dataset) while the pessimistic approach has significantly lower accuracy, as rejected objects count as a misclassification. This indicates that the disambiguation method classifies successfully the majority of rejected cases.

Also note that, as the safe approach considers a subset of the original test set (it excludes all cases satisfying rules from other classes and all cases not satisfying any rule), it is not unusual to obtain some results like that in Table 8 for Wine (99.2% of accuracy) or Diabetes (86.2% of accuracy), which is uncommon. The reason for these results is after eliminating all rejected elements, the remaining cases have been evaluated successfully. Notwithstanding, the number of rejected elements obtained in Algorithm 4 for pessimistic and safe approaches are relatively high.

When comparing the results obtained by the standard, reviewed and complete approaches it is visible the performance improvement of the complete approach when compared to the others. The complete approach has even achieved better results than the decision tree itself.

6 CONCLUSIONS AND FUTURE WORK

One of the main advantages of using a rough set based rule induction method is the interpretability of its resulting rules and its ability to cope with uncertainty in data. Unfortunately, most of the preexisting methods have been designed to deal with categorical data. The exceptions are (GRZYMALA-BUSSE, 2010) which induce rules in the exact same way as *LEM2* (Learning from Examples Module 2) (GRZYMALA-BUSSE, 1992), and the method (ZHAO *et al.*, 2006), which is based on similarity relations. Both these approaches do not have the rejection option (Definition 13).

In this work, we explored ways to make rough sets based rule induction methods with rejection option applicable over any numerical dataset. We considered two rule induction methods from rough sets theory *LERS* (Learning From Examples Using Rough Sets) (GRZYMALA-BUSSE, 1997) and *IM* (first letter of the authors name) (INUIGUCHI; MIYAJIMA, 2007) approach, and extend them to work with numerical data by applying to the discretization method MDLP (Minimum Description Length Principle) (RISSANEN, 1978). However, in reason of the discretization method, it is unavoidable to lose information/accuracy, which may lead to an increasing number of rejected objects. Then we take into account the notion of partial satisfiability as a post processing step to classify some of the rejected objects in terms of the satisfiability degree to which a rejected object satisfies each rule, so reducing the number of rejected cases.

Although the partial satisfiability is not enough to eliminate all rejected objects, it reduces their number as can be seen in Tables 11 and 12 when compared to Tables 5,6,7 and 8. However, as rejection option may not be desirable in all classification problems, we introduce the Complete Approach in Section 4.3, to classify completely all object given as input. Such an approach simply resorts to a decision tree to obtain the decision value of the rejected objects left by the Standard Approach. In short, in this work we realized the following contributions:

- Extension of both *LERS* and *IM* for arity > 2 (multiclass task)
- Induction over numerical data using *LERS* an *IM*
- Improvement in the classification accuracy of decision trees

As future work, we intend to improve both the discretization strategies and induction methods to reduce the number of rejected objects in Standard and Reviewed approaches. As for extensions of the rule induction methods, we plan to make them suitable to deal with numerical data straightforwardly. Then the results will be potentially more accurate as the information loss

inherited from the discretization will be surmounted.

Besides that, we also intend to present an alternative approach for classification of data example matching patterns that rules induced by *LERS* and *IM* are not able to identify. Instead of employing decision trees to deal with rejected objects as in the Complete Approach, we can resort to BART (Bayesian Additive Regression Tree) decision trees (KAPELNER; BLEICH, 2013). We guess if the resulting method will improve the accuracy of the BART decision tree, as well as the Complete Approach has improved the accuracy of the decision trees.

REFERENCES

ALBUQUERQUE, R. d. S. A. S.; ALCÂNTARA, J. F. L. A. L.; GOMES, J. P. P. G. P. A rough sets-based rule induction for numerical datasets. In: IEEE. **2018 7th Brazilian Conference on Intelligent Systems (BRACIS)**. [*S. l.*], 2018. p. 510–515.

BEASLEY, J. E.; CHU, P. C. A genetic algorithm for the set covering problem. **European journal of operational research**, Elsevier, v. 94, n. 2, p. 392–404, 1996.

BEGUM, N.; HU, B.; RAKTHANMANON, T.; KEOGH, E. Towards a minimum description length based stopping criterion for semi-supervised time series classification. In: IEEE. **2013 IEEE 14th International Conference on Information Reuse & Integration (IRI)**. [*S. l.*], 2013. p. 333–340.

BICHSEL, M.; SEITZ, P. Minimum class entropy: A maximum information approach to layered networks. **Neural Networks**, Elsevier, v. 2, n. 2, p. 133–141, 1989.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001.

CHIPMAN, H. A.; GEORGE, E. I.; MCCULLOCH, R. E. *et al.* Bart: Bayesian additive regression trees. **The Annals of Applied Statistics**, Institute of Mathematical Statistics, v. 4, n. 1, p. 266–298, 2010.

CORTES, C.; DESALVO, G.; MOHRI, M. Learning with rejection. In: SPRINGER. **International Conference on Algorithmic Learning Theory**. [*S. l.*], 2016. p. 67–82.

CORTEZ, P.; EMBRECHTS, M. J. Using sensitivity analysis and visualization techniques to open black box data mining models. **Information Sciences**, Elsevier, v. 225, p. 1–17, 2013.

CRAVEN, M. W.; SHAVLIK, J. W. Using sampling and queries to extract rules from trained neural networks. In: **Machine Learning Proceedings 1994**. [*S. l.*]: Elsevier, 1994. p. 37–45.

EVANS, S.; BARNETT, B.; BUSH, S. F.; SAULNIER, G. J. Minimum description length principles for detection and classification of ftp exploits. In: IEEE. **IEEE MILCOM 2004. Military Communications Conference, 2004.** [*S. l.*], 2004. v. 1, p. 473–479.

FAYYAD, U.; IRANI, K. Multi-interval discretization of continuous-valued attributes for classification learning. 1993.

GOLAN, R. H.; ZIARKO, W. A methodology for stock market analysis utilizing rough set theory. In: IEEE. **Computational Intelligence for Financial Engineering, 1995., Proceedings of the IEEE/IAFE 1995**. [*S. l.*], 1995. p. 32–40.

GRECO, S.; INUIGUCHI, M.; SŁOWIŃSKI, R. Rough sets and gradual decision rules. In: SPRINGER. **International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing**. [*S. l.*], 2003. p. 156–164.

GRECO, S.; MATARAZZO, B.; SLOWINSKI, R. Rough set approach to decisions under risk. In: SPRINGER. **International Conference on Rough Sets and Current Trends in Computing**. [*S. l.*], 2000. p. 160–169.

GRECO, S.; MATARAZZO, B.; SLOWINSKI, R. Rough sets theory for multicriteria decision analysis. **European journal of operational research**, Elsevier, [*S. l.*], v. 129, n. 1, p. 1–47, 2001.

GRZYMALA-BUSSE, J. W. Lers-a system for learning from examples based on rough sets. In: **Intelligent decision support**. [*S. l.*]: Springer, 1992. p. 3–18.

GRZYMALA-BUSSE, J. W. A new version of the rule induction system lers. **Fundamenta Informaticae**, IOS Press, v. 31, n. 1, p. 27–39, 1997.

GRZYMALA-BUSSE, J. W. Mining numerical data–a rough set approach. In: **Transactions on Rough Sets XI**. [*S. l.*]: Springer, 2010. p. 1–13.

HERBEI, R.; WEGKAMP, M. H. Classification with reject option. **Canadian Journal of Statistics**, Wiley Online Library, v. 34, n. 4, p. 709–721, 2006.

HU, Q.; YU, D.; LIU, J.; WU, C. Neighborhood rough set based heterogeneous feature subset selection. **Information sciences**, Elsevier, v. 178, n. 18, p. 3577–3594, 2008.

HU, Q.; YU, D.; XIE, Z. Hybrid attribute reduction for classification based on a fuzzy rough set technique. In: SIAM. **Proceedings of the 2005 SIAM International Conference on Data Mining**. [*S. l.*], 2005. p. 195–204.

INUIGUCHI, M. Several approaches to attribute reduction in variable precision rough set model. In: SPRINGER. **International Conference on Modeling Decisions for Artificial Intelligence**. [*S. l.*], 2005. p. 215–226.

INUIGUCHI, M.; MIYAJIMA, T. Rough set based rule induction from two decision tables. **European Journal of Operational Research**, Elsevier, v. 181, n. 3, p. 1540–1553, 2007.

KAPELNER, A.; BLEICH, J. bartmachine: Machine learning with bayesian additive regression trees. **arXiv preprint arXiv:1312.2171**, 2013.

KAREEM, S. A.; POZOS-PARRA, P.; WILSON, N. An application of belief merging for the diagnosis of oral cancer. **Applied Soft Computing**, Elsevier, 2017.

KOHAVI, R. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In: CITESEER. **Kdd**. [*S. l.*], 1996. v. 96, p. 202–207.

KRYSZKIEWICZ, M. Rules in incomplete information systems. **Information sciences**, Elsevier, v. 113, n. 3-4, p. 271–292, 1999.

LAMAHAMADEH. **UCI Pima-Indians-Diabetes-DataSet**. 2013. Https://github.com/LamaHamadeh/Pima-Indians-Diabetes-DataSet-UCI. Acesso em: 08 jul. 2018.

LAW, R.; AU, N. Relationship modeling in tourism shopping: a decision rules induction approach. **Tourism Management**, Elsevier, v. 21, n. 3, p. 241–249, 2000.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. Http://archive.ics.uci.edu/ml. Acesso em: 08 jul. 2018.

LINDEBERG, T.; LI, M.-X. Segmentation and classification of edges using minimum description length approximation and complementary junction cues. **Computer Vision and Image Understanding**, Elsevier, v. 67, n. 1, p. 88–98, 1997.

MAJI, P.; BISWAS, R.; ROY, A. Soft set theory. **Computers & Mathematics with Applications**, Pergamon, v. 45, n. 4-5, p. 555–562, 2003.

MARTIN-BARRAGAN, B.; LILLO, R.; ROMO, J. Interpretable support vector machines for functional data. **European Journal of Operational Research**, Elsevier, v. 232, n. 1, p. 146–155, 2014.

MROZEK, A. Rough sets in computer implementation of rule-based control of industrial processes. In: **Intelligent Decision Support**. [*S. l.*]: Springer, 1992. p. 19–31.

NOWICKI, R.; SŁOWIŃSKI, R.; STEFANOWSKI, J. Rough sets analysis of diagnostic capacity of vibroacoustic symptoms. **Computers & Mathematics with Applications**, Pergamon, v. 24, n. 7, p. 109–123, 1992.

OHKI, M.; INUIGUCHI, M.; HARADA, T. Decision rule visualization for knowledge discovery by means of rough set approach. In: IEEE. **Granular Computing (GrC), 2011 IEEE International Conference on**. [*S. l.*], 2011. p. 502–507.

PARRA, P. P.; MACÍAS, V. B. Partial satisfiability-based merging. In: SPRINGER. **Mexican International Conference on Artificial Intelligence**. [*S. l.*], 2007. p. 225–235.

PAWLAK, Z. Rough sets. **International journal of computer & information sciences**, Springer, v. 11, n. 5, p. 341–356, 1982.

PAWLAK, Z. Conflicts and negotations. In: SPRINGER. **International Conference on Rough Sets and Knowledge Technology**. [*S. l.*], 2006. p. 12–27.

PAWLAK, Z.; SLOWINSKI, R. Decision analysis using rough sets. **International Transactions in Operational Research**, Wiley Online Library, v. 1, n. 1, p. 107–114, 1994.

RAILEANU, L. E.; STOFFEL, K. Theoretical comparison between the gini index and information gain criteria. **Annals of Mathematics and Artificial Intelligence**, Springer, v. 41, n. 1, p. 77–93, 2004.

RISSANEN, J. Modeling by shortest data description. **Automatica**, Elsevier, v. 14, n. 5, p. 465–471, 1978.

RISSANEN, J. Minimum description length principle. **Wiley StatsRef: Statistics Reference Online**, Wiley Online Library, 2014.

RUTKOWSKI, L. **Computational intelligence: methods and techniques**. [*S. l.*]: Springer Science & Business Media, 2008.

SAFAVIAN, S. R.; LANDGREBE, D. A survey of decision tree classifier methodology. **IEEE transactions on systems, man, and cybernetics**, IEEE, v. 21, n. 3, p. 660–674, 1991.

SHAN, N.; ZIARKO, W. Data-based acquisition and incremental modification of classification rules. **Computational Intelligence**, Wiley Online Library, v. 11, n. 2, p. 357–370, 1995.

SKOWRON, A.; RAMANNA, S.; PETERS, J. F. Conflict analysis and information systems: a rough set approach. In: SPRINGER. **International Conference on Rough Sets and Knowledge Technology**. [*S. l.*], 2006. p. 233–240.

SLOWINSKI, R.; ZOPOUNIDIS, C. Application of the rough set approach to evaluation of bankruptcy risk. **Intelligent Systems in Accounting, Finance and Management**, Wiley Online Library, v. 4, n. 1, p. 27–41, 1995.

SOUSA, R.; NETO, A. R. da R.; CARDOSO, J. S.; BARRETO, G. A. Classification with reject option using the self-organizing map. In: SPRINGER. **International Conference on Artificial Neural Networks**. [*S. l.*], 2014. p. 105–112.

STEFANOWSKI, J. On rough set based approaches to induction of decision rules. **Rough sets in knowledge discovery**, v. 1, n. 1, p. 500–529, 1998.

STEFANOWSKI, J.; SŁOWIŃSKI, R.; NOWICKI, R. The rough sets approach to knowledge analysis for classification support in technical diagnostics of mechanical objects. In: SPRINGER. **International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems**. [*S. l.*], 1992. p. 556–565.

TRABELSI, S.; ELOUEDI, Z.; LINGRAS, P. Rule discovery process based on rough sets under the belief function framework. In: SPRINGER. **International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems**. [*S. l.*], 2010. p. 726–736.

TRABELSI, S.; ELOUEDI, Z.; LINGRAS, P. Classification systems based on rough sets under the belief function framework. **International Journal of Approximate Reasoning**, Elsevier, v. 52, n. 9, p. 1409–1432, 2011.

TSUMOTO, S. Modelling medical diagnostic rules based on rough sets. In: SPRINGER. **International Conference on Rough Sets and Current Trends in Computing**. [*S. l.*], 1998. p. 475–482.

TSUMOTO, S. Automated extraction of hierarchical decision rules from clinical databases using rough set model. **Expert systems with Applications**, Elsevier, v. 24, n. 2, p. 189–197, 2003.

YUAN, M.; WEGKAMP, M. Classification methods with reject option based on convex risk minimization. **Journal of Machine Learning Research**, v. 11, n. Jan, p. 111–130, 2010.

ZADEH, L. A. Soft computing and fuzzy logic. In: **Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh**. [*S. l.*]: World Scientific, 1996. p. 796–804.

ZADEH, L. A. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. **Fuzzy sets and systems**, Elsevier, v. 90, n. 2, p. 111–127, 1997.

ZHANG, Q.; XIE, Q.; WANG, G. A survey on rough set theory and its applications. **CAAI Transactions on Intelligence Technology**, Elsevier, v. 1, n. 4, p. 323–333, 2016.

ZHAO, S.-Y.; NG, W. W.; TSANG, E. C.; YEUNG, D. S.; CHEN, D.-G. Rule induction from numerical data based on rough sets theory. In: IEEE. **Machine Learning and Cybernetics, 2006 International Conference on**. [*S. l.*], 2006. p. 2294–2299.