A comparison of sixteen classification strategies of rule induction from incomplete data using the MLEM2 algorithm

Venkata Siva Pavan Kumar Nelakurthi

Submitted to the Department of Electrical Engineering & Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

Thesis Committee:

Dr. Jerzy Grzymala-Busse: Chairperson

Dr. Guanghui (Richard) Wang: Member

Dr. Prasad Kulkarni: Member

05/03/2020

Date Defended

The Thesis Committee for Venkata Nelakurthi certifies that this is the approved version of the following thesis:

A comparison of sixteen classification strategies of rule induction from incomplete data using the MLEM2 algorithm

Thesis Committee:

Dr. Jerzy Grzymala-Busse: Chairperson

Dr. Guanghui (Richard) Wang: Member

Dr. Prasad Kulkarni: Member

05/08/2020

Date Defended

ABSTRACT

In data mining, rule induction is a process of extracting formal rules from decision tables, where the later are the tabulated observations, which typically consist of few attributes, i.e., independent variables and a decision, i.e., a dependent variable. Each tuple in the table is considered as a case, and there could be n number of cases for a table specifying each observation. The efficiency of the rule induction depends on how many cases are successfully characterized by the generated set of rules, i.e., ruleset. There are different rule induction algorithms, such as LEM1, LEM2, MLEM2. In the real world, datasets will be imperfect, inconsistent, and incomplete. MLEM2 is an efficient algorithm to deal with such sorts of data, but the quality of rule induction largely depends on the chosen classification strategy. We tried to compare the 16 classification strategies of rule induction using MLEM2 on incomplete data. For this, we implemented MLEM2 for inducing rulesets based on the selection of the type of approximation, i.e., singleton, subset or concept, and the value of alpha for calculating probabilistic approximations. A program called rule checker is used to calculate the error rate based on the classification strategy specified. To reduce the anomalies, we used ten-fold cross-validation to measure the error rate for each classification. Error rates for the above strategies are being calculated for different datasets, compared, and presented.

ACKNOWLEDGMENTS:

I would like to express my heartfelt thanks to my advisor and thesis committee chair, Dr.Jerzy Grzymala Busse, for his continued support and encouragement. The knowledge he imparted on me through his data mining lectures and during research discussions has been invaluable and helped me complete this thesis.

I extend my thanks to my parents for their constant support and encouragement in every aspect of my graduate education.

Not to forget, I am also thankful to all my friends and peers of The University of Kansas.

TABLE OF CONTENTS

Abstract iii

1. INTRODUCTION 1

1.1 Motivation 1

1.2 Organization of the document 4

2. BACKGROUND 6

2.1 Decision tables and decision rules 6

2.1.1 Information representation in decision tables 6

2.1.2 Decision rules 7

2.1.3 Handling incomplete attribute values in decision tables 8

2.2 Rough set theory 10

2.2.1 Preliminary concepts of rough set theory 10

2.2.2 Rule induction algorithms based on rough set theory 12

2.2.3 Classification system in LERS 14

2.2.4 Characteristic sets and characteristic relation for incomplete decision tables 15

2.2.5 Rough set theory in incomplete decision tables 17

2.2.6 MLEM2 algorithm 20

3. CLASSIFICATION STRATEGIES 22

3.1 Classification factors 22

3.2 Classification strategies 23

4. EXPERIMENTS 26

4.1 Experiment setup 26

4.2 Comparison of classification strategies 29

5. CONCLUSIONS 30

6. REFERENCES 31

List of Figures 36

List of Tables 37

Appendix A LEM2 algorithm 38

Appendix B Implementation details 39

CHAPTER 1: INTRODUCTION

In data mining different algorithms exist to process large amounts of data. However, real-life data are characterized by missing values, which renders the classification process difficult.

1.1 Motivation

The data mining algorithms are used for classifying critical and sensitive real-life data. Hence, accurate classification becomes essential. Many different algorithms were proposed to address this issue. Few of the algorithms that are commonly used for classification are following:

1. Linear lassifiers like naïve Bayes classifier :

Naïve bayes classifier is based on Bayes theorem. This classifier assumes that the presence of a certain feature in a clas is unrelated to presence od any other feature or that all the properties have independent contribution to probability [22].

2. Naerest Neighbour classifiers

The k-nearest-neighbors algorithm is a supervised classification technique that uses proximity as a proxy for 'similarity'. The algorithm takes a set of labelled points and uses them to learn how to label other points. To classify (label) a new point, it looks at the labelled points closest to that new point (those are its nearest neighbors) [21].

3. Support vector machines

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points [19].

4. Decision trees

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a set of data into smaller and smaller subsets while at the same time developing an associated decision tree. The final result is a tree with decision nodes and leaf nodes that can be used to classify new points [20].

5. Random forest

A Random Forest Classifier is a set of decision trees from randomly selected subset of training set. It **aggregates the votes from different decision trees** to decide the final class of the test object [18].

6. Neural networks

Neural networks are built of simple elements called neurons, which take in a real value, multiply it by a weight, and run it through a non-linear activation function. By constructing multiple layers of neurons, each of which receives part of the input variables, and then passes on its results to the next layers, the network can learn very complex functions. In theory, a neural network is capable of learning the shape of just any function, given enough computational power [17].

The technique of learning from inconsistent and incompete data , which is based on the concept of rough set theory called rule induction, is also one such example [25]. While most of the above algorithms are based on classification using feature selution, rule induction is primarily based on using rough-set theory which is better than using rule induction with feature selection [28].

Rule induction algorithms deal with missing examples in two ways. One is by preprocessing the incomplete decision table to make it complete and then performing rule induction on it. The other is by performing rule induction on incomplete data directly. MLEM2 algorithm that is based on rough set theory is one such rule induction algorithm that acts on missing examples directly. When an incomplete decision table is provided, rough set theory handles it by approximating the inconsistent or missing data with two approximations, lower and upper.

Effectiveness of MLEM2 is defined by different factors like matching factor, specificity, strength, and usage of certain and possible rules. Based on these factors, we can create sixteen different strategies to classify the ruleset. These rule sets can be compared by the error rates calculated using these strategies.

1.2 Organization of the document

This thesis is organized into six chapters that include a detailed background of rule induction, a description of the MLEM2 and sixteen classification strategies, the related experimentation, and results.

Chapter1 gives an overview of the motivation for the thesis. An overall idea about the topic and the main focus of the thesis has been explained.

Chapter 2 provides background information related to the thesis. It is divided into two parts. The first part explains some basic concepts related to rule induction. It also explains the concept of missing attribute values in decision tables and the different approaches to induce rules from these incomplete decision tables. The second part describes the rough-set approach and the related concepts in detail. Finally, it describes the MLEM2 algorithm with an example on how to induce rules from incomplete decision tables.

Chapter 3 introduces the classification factors used in rule induction and specifies the sixteen classification strategies that have been compared for incomplete data using the MLEM2 algorithm.

Chapter 4 shows the experimental results and the error rates of the classification strategies applied to different incomplete data sets using the MLEM2 algorithm. *Chapter 5* is for the conclusions of the thesis and suggestions for future areas of exploration.

CHAPTER 2: BACKGROUND

This chapter gives an overview of the fundamental concepts of rough set theory and the MLEM2 algorithm.

2.1 Decision tables and decision rules

In the field of data mining the information available about real-world data is represented in the form of a decision table. A decision table is a powerful and clear way of representing data.

2.1.1 Information representation in decision tables

A decision table describes a set of examples. Each example is presented in the form of a set of attribute values and a decision value. The attribute values provide information about the object, and the decision value provides a way of categorizing the object. Also, each example belongs to a class, known as the concept. Such a class is represented by a set of examples having the same value for a decision. A decision table describing cars is shown in Table 2.1.

Table 2.1: An example of the decision table

Case	Year	Mileage	Decision
1	1990	300000	Least Reliable
2	2000	150000	Less Reliable
3	2008	6000	Very Reliable

In the above decision table, 'Least Reliable', 'Less Reliable' and 'Very Reliable' are concepts about the car. Mileage and year are attributes that describe the car.

Decision tables are a precise yet compact way to model complicated situations. In mathematical terms, a decision table 'DT' is an information table of the form (U, C $\cup$ {d}) where d is a distinguished attribute called a decision, U is a set of objects, and elements of C are called condition attributes.

2.1.2 Decision rules

Decision rules classify the data in the decision table into different concepts. Usually decision rules are expressed in the form:

If (attribute1 = value1) & (attribute2 = value2) - > (decision = value)

The left-hand side represents the attribute-value pairs, and the right-hand side represents the concept. A case x is covered by a rule r if and only if every condition or attribute-value pair of r is satisfied by the corresponding attribute value for x. For example, from Table 2.1, the following rule can be extracted

If (Year = 2000) -> (decision = Less Reliable)

Since case 2 with the concept "Less Reliable" satisfies this rule, it is said to be covered by the rule.

Decision rules can be described using two important terms – completeness and consistency. The basic idea is to compute rules which are both complete and consistent. Such a rule set R is called the discriminant. Consider a concept C. This is completely covered by a rule set R if for every case x from C, there exists a rule such that r covers x. A rule set R is complete if every concept from the data set is completely covered by R. It is consistent with the dataset, if every case x is covered by R, where x is a member of the concept C indicated by R. In other words, a rule set R is consistent if and only if every rule $r \in R$ is consistent with the dataset.

Decision rule induction is the process by which rules are induced from the decision tables. It involves the extraction of high-level information from low-level data and is the most fundamental data mining technique. Examples of rule induction algorithms are LEM1 and LEM2 [6].

2.1.3 Handling incomplete attribute values in decision tables

Until recently, data represented in decision tables were assumed to be complete and consistent. In recent times, the need for new algorithms to deal with uncertain input, lost input or erroneous information has risen. Hence, handling data with incomplete values is an active topic of research in data mining. One of the general assumptions made in all these rule induction algorithms is that, in a particular case, at least one value should be present, which means we cannot induce a rule from a case that has all of its attributes missing [2].

In general, it is assumed that there are three specific types of missing data:

(1) The attribute value might be lost, i.e., either erased or missed out. These are called "lost conditions", represented by '?'.

(2) The attribute value might be irrelevant to the current classification. In other words, value is unnecessary. So, we do not care if they are missing or not. Hence, they are called “do not care” conditions. These missing attribute values are represented by ‘*’.

(3) The missing attribute values are assumed to belong to a particular concept. These are called “attribute-concept” values, represented by ‘-‘.

The theories that provide a solution to the missing attribute value problem follow two main approaches.

1) The input data is preprocessed, i.e., incomplete attribute values are filled up using some heuristics that the application demands. Some examples are: replacing the missing attribute value with the most frequently occurring value, replacing the missing attribute value with the attribute average, replacing the missing attribute value with the most frequently occurring value in that concept, etc. Hence, the data becomes complete after preprocessing. Now, the rule induction becomes the same as the traditional approach [5].

2) Inducing rules with the incompletely specified table itself. Examples for this approach are MLEM2 [23] and C4.5 algorithms [2, 24].

Table2.2 illustrates the three types of missing data using an example.

Table 2.2: An incomplete decision table: example 1

Case	Attributes			Decision
	Temperature	Headache	Nausea	Flu
1	High	-	No	Yes
2	Very_high	Yes	Yes	Yes
3	?	No	No	No
4	High	Yes	Yes	Yes
5	High	?	Yes	No
6	Normal	Yes	No	No
7	Normal	No	Yes	No
8	*	Yes	*	Yes

In the above table, there are three attributes namely 'Temperature', 'Headache', 'Nausea' and one decision 'Flu'. It has eight cases of which four have missing attribute values. Cases 3 and 5 have "lost conditions", case 8 has two "do not care" conditions and case 1 has an "attribute-concept" value. There are two concepts 'Yes' and 'No' for classification.

2.2 Rough set theory

Rough set theory, introduced by Z.Pawlak, brings a mathematical approach to data mining. It has evolved into a very powerful tool for analyzing the ambiguity of data elements and has served as the foundation for innumerable theories. It has also proved its usefulness in many real-life applications.

2.2.1 Preliminary concepts of rough set theory

The main idea behind rough-set theory is the indiscernibility relation between objects which can be explained as follows:

Any two objects in the universe can be assumed to be indiscernible if the available or known facts about them are similar. In other words, they are categorized based on the available information about them. The unknown information might show them to be different. But according to the facts available, they are classified to be similar [11, 12].

In mathematical terms, the indiscernibility relation can be explained as: A decision table is a pair (U, C) where U is a non-empty set of objects and C is a non-empty set of attributes [3, 14, 15]. For each subset of attributes $B \subseteq C$, the indiscernibility relation IND (B) can be defined as:

$$IND\,(B) = \{(x, y) \in U \times U : \forall c_i \in B, c_i(x) = c_i(y)\}$$

where x and y are two objects and $c_i(x)$ and $c_i(y)$ are the values of attribute c_i for x and y, respectively. According to rough set theory, with each set, a pair of precise sets called lower and upper approximation is associated. For any element x of U, the equivalence class of IND containing x will be denoted by $[x]_{IND}$. A lower approximation of X in (U, IND), denoted by $\underline{R}$X, is the set

$$\{x \in U \mid [x]_{IND} \subseteq X\}.$$

An upper approximation of X in (U, IND), denoted by $\overline{R}X$, is the set

$$\{x \in U \mid [x]_{IND} \cap X \neq \varnothing\}.$$

Lower approximation represents data that certainly belong to the set and hence the rules extracted using lower approximation are called certain rules. Upper approximation represents data that may possibly belong to the set and hence the rules induced this way are called possible rules [3, 16].

Let U be a non-empty set called the universe and let IND be the indiscernibility relation. An ordered pair (U, IND) is called an approximation space. Let X be a subset of U. X is defined in terms of the set in (U, IND). Any finite union of elementary sets in (U, IND) is called the definable set in (U, R). The lower approximation of X in (U, R) is the greatest definable set in (U, IND), contained in X. The upper approximation of X in (U, IND) is the least definable set in (U, IND), containing X [1].

The main idea behind the approximations is that if it looks impossible to completely categorize a set of objects using the available information, it can be approximated using the lower and upper approximations. The set that separates the lower and the upper approximation is the boundary region for the rough set. The main advantage of rough set theory is that it does not require any additional information about data, unlike a lot of statistical approaches. Thus, it is possible to find hidden information about data more efficiently [2].

2.2.2 Rule induction algorithms based on rough set theory

Rule induction is one of the most important techniques of data mining that extracts regularities from the real-life data. The critical task of rule induction is to induce a rule set R that is consistent and complete. That said, rule induction has its own challenges. For example, the input data can be affected by errors, may contain numeric attributes (which need to be converted to symbolic attributes before or during rule induction), may be incomplete due to missing values or inconsistent due to two cases having the same attribute values but a different decision value. There are two important kinds of rule induction algorithms, global and local. In global algorithms all the attribute values are considered where as in local algorithms the concept of attribute-value pairs come into picture. The two important rule induction algorithms based on rough set theory are LEM1 (Learning by Examples) and LEM2 [6].

Let B be a nonempty subset of a set C of all attributes. LEM1 is a global algorithm that is based on rough set theory and uses the indiscernibility concept IND (B). The family of all B-elementary sets will be denoted by B*. According to the definition of LEM1, for a decision *d* we say that *{d}* depends on *B* if and only if *B** <= *{d}** where {d}* is the family of elementary sets of d. A global covering of *{d}* is a subset *B* of C such that *{d}* depends on *B* and *B* is minimal in C. The main idea behind LEM1 is to compute a single global covering. The computed rules then undergo a process called dropping conditions. This is done by scanning through all the computed rules, dropping conditions one by one and then, checking if the consistency of the ruleset is affected by dropping the condition. If the rules with dropped conditions are consistent and cover the same number of cases, then the condition can be termed as a redundant condition and dropped forever.

On the other hand, LEM2 is a local algorithm, which takes into account attribute-value pairs and then converts them into a ruleset. LEM2 works better than

LEM1 because it works on lower and upper approximations separately and hence the input files for LEM2 are always consistent.

For an attribute-value pair (a, v) = t, a block of t, denoted by [t], is a set of all cases from the universe U which has attribute a and value v. Let B be a nonempty lower or upper approximation of a concept represented by a decision- value pair (d, w). Set B depends on a set T of attribute-value pairs t = (a, v) if and only if no proper subset T′ of T exists such that B depends on that subset T′.

$$\varnothing \neq [T] = \cap_{t \in T} [t] \subseteq B$$

Thus in LEM2, the first step is to compute a set of all attribute-value blocks [a, v]. After that, an iterative process is followed to identify the minimal complex. A detailed description of the LEM2 procedure is given in Appendix A [6].

2.2.3 Classification system in LERS

LERS (Learning from Examples based on Rough Sets) is a well-known data mining system that induces rules based on rough set theory and uses the already induced rules to further classify new data. LERS associates three basic parameters to evaluate the decision rules, namely strength, specificity, and support.

- *Strength* is defined as the total number of examples correctly classified by the rule during training. It is a measure of how well the rule performed.

- *Specificity* is the total number of conditions or attribute-value pairs on the left-hand side of the rule. Thus, rules with more attribute-value pairs are considered to be more specific.
- *Support* is defined as the sum of scores of all matching rules from the concept [8, 9].

These factors are explained in detail in the classification strategies below.

2.2.4 Characteristic sets and characteristic relations for incomplete decision tables

For decision tables with missing attribute values, modified definitions are needed to determine to which attribute-value block the cases with missing attribute values can be added. The following heuristics are followed:

- If for an attribute *a* there exists a case *x* such that the value of *a* for *x* is '?', i.e., the corresponding value is lost, then the case *x* should not be included in any block [(*a*, *v*)] for all values *v* of attribute *a*.
- If for an attribute *a* there exists a case *x* such that the value of a for *x* is '*' i.e., the corresponding value is a do not care condition, then the case *x* should be included in blocks [(*a*, *v*)] for all specified values *v* of attribute *a*.
- If for an attribute *a* there exists a case *x* such that the value is a '-', i.e., the corresponding value is an attribute concept value then the case *x* should be included in blocks [(*a*, *v*)] for all specified values *v* of attribute *a* belonging to that concept [2].

For example, the attribute-value block for Table 2.2 can be defined as:

[(Temperature, High)] = {1, 4, 5, 8},

[(Temperature, Very_high)] = {2, 8},

[(Temperature, Normal)] = {6, 7},

[(Headache, Yes)] = {1, 2, 4, 6, 8},

[(Headache, No)] = {3, 7},

[(Nausea, No)] = {1, 3, 6, 8},

[(Nausea, Yes)] = {2, 4, 5, 7, 8}.

A few important terms that need to be defined in rough set theory are characteristic sets and characteristic relations. The characteristic set $K_B(x)$ for a set B, is the intersection of blocks of attribute-value pairs (*a*, *v*) for all attributes *a* from *B* for which ρ(*x*, *a*) is specified and ρ(*x*, *a*) = *v*. For Table 2.2, the characteristic set can be defined for $B = A$ as:

For Case 1: $K_A(1)$ = {1, 4, 5, 8} ∩ {1, 3, 6, 8} = {1, 8},

For Case 2: $K_A(2)$ = {2, 8} ∩ {1, 2, 4, 6, 8} ∩ {2, 4, 5, 7, 8} = {2, 8},

For Case 3: $K_A(3)$ = {1, 2, 3, 4, 5, 6, 7, 8} ∩ {3, 7} ∩ {1, 3, 6, 8} = {3},

For Case 4: $K_A(4)$ = {1, 4, 5, 8} ∩ {1, 2, 4, 6, 8} ∩ {2, 4, 5, 7, 8} = {4, 8},

For Case 5: $K_A(5)$ = {1, 4, 5, 8}∩ U ∩ {2, 4, 5, 7, 8} = {4, 5, 8},

For Case 6: $K_A(6)$ = {6, 7} ∩ {1, 2, 4, 6, 8} ∩ {1, 3, 6, 8} = {6},

For Case 7: $K_A(7)$ = {6, 7, 8} ∩ {3, 7} ∩ {2, 4, 5, 7, 8} = {7},

For Case 8: $K_A(8)$ = {1, 4, 8}∩ {2,8}∩{1,2,4,6,8}={1,2,4,8}.

Thus, for a given interpretation of missing attribute values, the characteristic set $K_B(x)$ may be interpreted as the smallest set of cases that are indistinguishable from x [2, 10].

The characteristic relation R (B) is a relation on a set of all objects U defined for objects $x, y \in$ U as follows

$$(x, y) \in R\ (B) \text{ if and only if } y \in K_B(x)$$

The characteristic relation is known if we know all the characteristic sets x $\in$ U. For Table 2.2, the characteristic relation can be calculated as:

$R(A)$ = {(1, 1), (1, 8), (2, 2), (2, 8), (3, 3), (4, 4), (4, 8), (5, 4), (5, 5), (5, 8), (6, 6), (6, 8), (7, 7), (8, 2), (8, 4), (8, 6), (8, 8)}.

2.2.5 Rough set theory in incomplete decision tables

In rough set theory, incomplete decision tables are described using characteristic relations, in the same way, complete decision tables are described using the indiscernibility relation. For a complete decision table, once the indiscernibility relation is fixed and the concept is given, the lower and upper approximations are unique. For an incomplete decision table, for a given characteristic relation and concept there are three different ways to define lower and upper approximations, namely singleton, subset and concept.

In the case of singleton method, the singleton B-lower and B-upper approximations of X are defined as follows:

$$\underline{B}X = \{x \in U \mid K_B(x) \subseteq X\}$$

$$\overline{B}X = \{x \in U \mid K_B(x) \cap X \neq \emptyset\}$$

where $K_B(x)$ is the characteristic set and U is the set of all objects.

For example, in Table 2.2 the singleton A-lower and A-upper approximations of the two concepts {1, 2, 4, 8} and {3, 5, 6, 7} are:

$\underline{A}$ {1, 2, 4, 8} = {1, 2, 4, 8},

$\underline{A}$ {3, 5, 6, 7} = {3, 6, 7},

$\overline{A}$ {1, 2, 4, 8} = {1, 2, 4, 5, 8},

$\overline{A}$ {3, 5, 6, 7} = {3, 5, 6, 7}.

Singleton approximations are not very useful since they cannot be represented as union of attribute value blocks. So, the other approach uses characteristic relations instead of elementary sets. There are two different ways to do this. The first method is called subset approximations.

A subset B-lower approximation of X is defined as:

$$\underline{B}X = \cup\{K_B(x) \mid x \in U,\ K_B(x) \subseteq X\}$$

A subset B-upper approximation of X is defined as:

$$\overline{B}X = \cup\{K_B(x) \mid x \in U,\ K_B(x) \cap X \neq \emptyset\}$$

For example, for Table 2.2, the subset lower and upper approximations are given as follows:

$\underline{A}$ {1, 2, 4, 8} = {1, 2, 4, 8},

$\underline{A}$ {3, 5, 6, 7} = {3, 6, 7},

$\overline{A}$ {1, 2, 4, 8} = {1, 2, 4, 5, 8},

$\overline{A}$ {3, 5, 6, 7} = {3, 4, 5, 6, 7, 8}.

The second method is by changing the subset definition by replacing the universe U with the concept C. A concept *B*-lower approximation of the concept *X* is defined as:

$$\underline{B}X = \cup\{K_B(x) \mid x \in X, K_B(x) \subseteq X\}$$

A concept *B*-upper approximation of the concept *X* is defined as:

$$\overline{B}X = \cup\{K_B(x) \mid x \in X, K_B(x) \cap X \neq \emptyset\} = \cup\{K_B(x) \mid x \in X\}$$

For example, in Table 2.2, the concept lower approximations for B = A are:

$\underline{A}$ {1, 2, 4, 8} = {1, 2, 4, 8},

$\underline{A}$ {3, 5, 6, 7} = {3, 6, 7},

$\overline{A}$ {1, 2, 4, 8} = {1, 2, 4, 8},

$\overline{A}$ {3, 5, 6, 7} = {3, 4, 5, 6, 7, 8}.

For completely defined decision tables, singleton, subset, and concept lower and upper approximations are the same. As explained above, they are different for incomplete decision tables [2, 4, 10].

With these approximations, there is also probabilistic approximations which are a generalized form of singleton, subset and concept approximations.

For incomplete data, a B-concept probabilistic approximation can be defined as

$$\cup \{K_B(x) \mid x \in X, \Pr(X \mid K_B(x)) \geq \alpha\}$$

There is a lot of research existing for the comparison of classification systems using MLEM2 and probabilistic approximations [26, 27, 28].

2.2.6 MLEM2 algorithm

Most of the data mining algorithms are not designed to deal with numerical attributes. They take only symbolic (alphanumeric) attributes as inputs. So, to classify data containing numerical attributes, they have to be converted to symbolic attributes as a preprocessing step. Most of the data mining rule induction algorithms take the numerical data and preprocess it by a discretization process and then conduct the rule induction process. However, this approach doubles the processing time. So, the currently developed algorithms conduct rule induction and discretization at the same time. Examples for the latter approach are Modified LEM2 (MLEM2), C 4.5 etc [5, 9, 10].

The key focus in this thesis is the modified LEM2 (MLEM2) which is based on the LEM2 algorithm. MLEM2 classifies all attributes into two major types namely numerical and symbolic. Approximations are computed in different ways for numerical and symbolic attributes. The procedure is as follows: the first step is to sort all values of a numerical attribute. The next step is to compute the average of any two consecutive values of the sorted list. For each cut point x, MLEM2 works by creating two blocks, the first block containing the values smaller than x and the second block containing the values larger than x. The search space of MLEM2 includes both the blocks computed from symbolic attributes as well as the blocks computed from numerical attributes. From this point, the rule induction follows the same procedure as the LEM2 algorithm [9, 10].

With the background literature reviewed, the next chapter deals with the main theoretical aspects of the thesis.

CHAPTER 3: Classification Strategies

In this chapter we will see the factors of classification and sixteen classification strategies that have been compared in this work.

3.1 Classification factors

There are four factors for the classification of LERS:

Strength is a measure of how good the rule performed during training. It is the number of cases correctly classified by a single rule in training data. The higher the value is, the better.

Specificity is a measure of the completeness of a rule. It is the number of conditions (attribute-value pairs) of a rule. In other terms, a rule with a higher number of attribute-value pairs is more specific. Specificity can either be used or left out during experiments.

Matching_factor is a measure of matching of a case and a rule. It is the ratio of the number of matched attribute-value pairs of a rule with a case to the total number of attribute-value pairs of the rule. In these experiments, two versions of matching are used: using complete matching first, then partial matching if needed, and using both complete and partial matching at once. Therefore, matching-factor is always used in partial matching.

Support is related to a concept *C*. It is the sum of all matching rule on *C*. It is defined as follows:

$$\sum_{partially\ matching\ rules\ R\ describing\ C} Matching\_factor(R) * Strength(R) * Specificity(R)$$

The concept with the largest score will have precedence, and if there is a tie among concepts, the strongest rule determines the precedence.

In the above formula, any factor can be equal to one. For example, the value of specificity can be set to one, or we can say that the classification strategy does not use specificity. Also, during complete matching, the value of matching_factor will be one and vice versa.

3.2 Classification strategies

For the experiments, four combinations of certain and possible rules i.e. a) using only certain rules, b) using only possible rules, c) using certain rules first, then possible rules if necessary and d) using both certain and possible rules have been used. With these four options for choosing rule sets, two options for using specificity, and two options for matching_factor, the sixteen classification strategies that are used for the experiments are listed below:

1. Using only certain rules, specificity, complete matching then partial matching if necessary,
2. Using only certain rules, specificity and both complete matching and partial matching,
3. Using only certain rules and complete matching, then partial matching if

necessary and not using specificity,

4. Using only certain rules and both complete and partial matching, not using specificity,
5. Using only possible rules, specificity and complete matching, then partial matching if necessary,
6. Using only possible rules, specificity, both complete matching and partial marching,
7. Using only possible rules, not using specificity, using complete matching, then possible matching if necessary,
8. Using only possible rules and both complete and partial matching, not using specificity,
9. Using certain rules first, then possible rules if necessary, specificity, and complete matching, then partial matching if necessary,
10. Using certain rules first, then using possible rules if necessary, specificity and both complete and partial matching,
11. Using certain rules first, then using possible rules if necessary and complete matching if necessary, not using specificity,
12. Using certain rules first, then using possible rules if necessary and both compete and partial matching, not using specificity,
13. Using both certain and possible rules, specificity, using complete matching then partial matching if necessary,
14. Using both certain and possible rules, specificity, and both complete and partial matching.

15. Using both certain and possible rules, complete matching, then partial matching if necessary, not using specificity.
16. Using both certain and possible rules and both complete and partial matching, not using specificity.

CHAPTER 4: EXPERIMENTS

This chapter explains the experiments done on the rules induced by the valued tolerance relation.

4.1 Tools used for evaluation of the decision rules

The main parameter that is used for evaluating the decision rules is the error - rate. The validation of the decision rules is done by a process called ten-fold cross-validation. The rule checker program takes the rules induced by the algorithm (MLEM2) and computes the error rate. The algorithm takes Input data file as input and produces a report file, a file with rules, and error rate as outputs. These files with their descriptions and examples are as follows:

(a) Data input file – the file which is given as the input to the algorithm,

(b) Report file - This file gives the statistics if the results based on the rules and data. One such example report file is:

Fig 4.1 Sample report file

```
This report was created from: valuedresult and from: bank-5.d
The total number of examples is: 66
The total number of attributes is: 5
The total number of rules is: 25
The total number of conditions is: 100
The total number of examples that are not classified: 41
The total number of examples that are incorrectly classified: 0
The total number of examples that are not classified or are incorrectly classified:
41
Error rate: 62.12 percent

Concept(d, 1):
The total number of examples that are not classified: 8
The total number of examples that are incorrectly classified: 0
The total number of examples that are correctly classified: 25
The total number of examples in the concept: 33
```

```
Concept(d, 2):
The total number of examples that are not classified: 33
The total number of examples that are incorrectly classified: 0
The total number of examples that are correctly classified: 0
The total number of examples in the concept: 33
```

(c) Name of the new rule set file. This gives the rule file according to the LERS format. One such example rule file is:

Fig 4.2 Sample ruleset file

```
! This rule file was created from: input.txt.r and from: input.txt
! ________________________________________

2, 1, 1
(a3, 1) & (a4, 0) -> (d, low)
3, 1, 1
(a1, 2) & (a3, 2) & (a4, 1) -> (d, high)
3, 1, 1
(a2, 2) & (a3, 1) & (a4, 3) -> (d, high)
```

Where the three numbers before the rule specify the strength, specificity, and support, respectively. In addition to the above input files, the rule checker needs to be specified if the error rate has to be computed for the certain or possible rules. For this, the following inputs are to be provided after the generation of rules.

(a) If the rules are to be evaluated using conditional probability or strength,

(b) If support is to be used or not,

(c) If specificity is to be used or not,

(d) If the concept statistics need to be printed,

(g) If the case classifications need to be printed.

A typical example report generated with the concept statistics is given below:

Fig 4.3 Sample statistics

```
General Statistics

This report was created from: ru and from: input.txt
The total number of cases: 12
The total number of attributes: 4
The total number of rules: 3
The total number of conditions: 8
The total number of cases that are not classified: 0
    PARTIAL MATCHING
  The total number of cases that are incorrectly classified: 1
  The total number of cases that are correctly classified: 2
    COMPLETE MATCHING
  The total number of cases that are incorrectly classified: 4
  The total number of cases that are correctly classified: 5
    PARTIAL AND COMPLETE MATCHING
The total number of cases that are not classified or incorrectly classified: 5
Error rate: 41.67%

Concept ("d","low")
The total number of cases that are not classified: 0

    PARTIAL MATCHING
  The total number of cases that are incorrectly classified: 1
  The total number of cases that are correctly classified: 0

   COMPLETE MATCHING
  The total number of cases that are incorrectly classified: 4
  The total number of cases that are correctly classified: 1
The total number of cases in the concept: 6

Concept ("d","high")
The total number of cases that are not classified: 0

    PARTIAL MATCHING
  The total number of cases that are incorrectly classified: 0
  The total number of cases that are correctly classified: 2

   COMPLETE MATCHING
  The total number of cases that are incorrectly classified: 0
  The total number of cases that are correctly classified: 4
The total number of cases in the concept: 6
```

(c) Error rate calculator

The error rate calculated will be saved onto a file as a percentage value and also printed out among the above statistics.

4.2 Comparison of classification strategies:

From the error rates calculated as described above for different datasets have been tabled below for the comparison.

Table 4.1 Error rates

	Abalone	Ecoli	Ion	Pima	Yeast
1	11.10	35.14	32.00	67.00	8.00
2	11.00	35.67	33.46	67.31	8.12
3	11.01	36.01	32.14	67.36	8.34
4	11.21	35.43	33.61	67.56	8.34
5	12.34	40.01	42.15	71.24	7.36
6	10.13	43.27	37.48	67.97	7.34
7	12.34	36.76	36.73	66.54	8.93
8	13.01	35.98	40.47	72.34	10.05
9	11.12	36.74	33.61	68.37	8.60
10	10.36	43.12	32.14	69.31	10.4
11	11.21	39.01	42.15	73.12	8.60
12	11.84	40.21	46.41	72.67	7.35
13	12.31	38.60	39.16	65.19	9.41
14	13.12	41.14	44.47	66.78	8.00
15	11.01	37.43	33.61	67.31	8.12
16	11.00	36.12	31.31	67.57	10.10

CHAPTER 5: CONCLUSIONS

For the above experiments, we took five data sets of different sizes, and each set has been shuffled in the beginning to produce data sets for ten–fold cross-validation. The rule sets have been created using MLEM2 and then are used to calculate error rates using different factors like partial or complete matching, using certain and possible rules, and using specificity and strength.

From the above results, we can observe that for strategies (1,2,3,4), there is no significant effect by using partial or complete matching. There is also no significant difference observed by using specificity on the ruleset, this can be observed from the strategies 2 and 4, 6 and 8, 10 and 12 and 14 and 16. One factor that has a considerable impact on effectiveness is choosing between certain and possible rules. The options with possible rules tend to produce lower error rates compared to certain rules, and using both rules yielded lower error rates.

To summarize the above results, there is no single strategy that can be considered as the best one to use, and the effectiveness of these strategies can also be dependent on the size and type of data as well.

REFERENCES

[1] Jerzy Stefanowski, Alexis Tsoukias, *Incomplete Information tables and Rough Classification*, Computational Intelligence Journal, ed. By Randy Goebel and Russell Greiner, Volume 17, Number 3, Aug 2001, 545-566.

[2] Jerzy W. Grzymala-Busse, *Rough set Theory with Applications to Data Mining,* Real World Applications of Computational Intelligence, ed. by M. G. Negoita and B. Reusch, Springer Verlag, 2005, 221-244.

[3] Jerzy W. Grzymala-Busse, *On the unknown attribute values in Learning from Examples*, Proceedings of the ISMIS-91, 6th International Symposium on Methodologies for Intelligent Systems, Charlotte, North Carolina, Oct 16-19, 1991, 368-377.

[4] Jerzy W. Grzymala-Busse, *Three Approaches to Missing Attribute Values- A Rough Set Perspective,* Proceedings of the Workshop on Foundation of Data Mining, associated with the Fourth IEEE International Conference on Data Mining, Brighton, UK, November 1-4, 2004, 55–62.

[5] Jerzy W. Grzymala-Busse, Witold J. Grzymala-Busse, *Handling Missing Attribute Values,* The Data Mining and Knowledge Discovery Handbook 2005, ed. By Oded Miamon and Lior Rokach, Springer-Verlag, 2005, 37-57.

[6] Jerzy W. Grzymala-Busse, Witold J.Grzymala-Busse, *Rule Induction,* The Data Mining and Knowledge Discovery Handbook, ed. By Oded Maimon and Lior Rokach, Springer-Verlag, 2005, 277-294.

[7] Jerzy W. Grzymala-Busse, *EECS 837 Lecture Notes*, taken during Fall 2006.

[8] Temidayo.B.Ajayi, *Incomplete Data Mining, A Rough Set Approach,* Thesis document submitted to the Department of Electrical Engineering and Computer Science, University of Kansas, under the guidance of Jerzy W. Grzymala-Busse.

[9] Jerzy W. Grzymala-Busse, *A Comparison of Three Strategies to Rule Induction from Data with Numerical Attributes,* Proceedings of the International Workshop on Rough Sets in Knowledge Discovery (RSKD2003), associated with the European Joint Conferences on Theory and Practice of Software 2003, Warsaw, Poland, April 2003, 132-140.

[10] Jerzy W. Grzymala-Busse, Sachin Siddhaye, *Rough Set Approaches to Rule Induction from Incomplete Data,* Proceedings of the IPMU'2004, the 10th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Perugia, Italy, July , 2004, Volume 2, 923–930.

[11] Zdzislaw Pawlak, Jerzy Grzymala-Busse, Roman Slowinski, and Wojciech Ziarko, *Rough Sets,* Communications of the ACM November 1995, Volume 38, Issue 11 , 88 - 95.

[12] Zdzislaw Pawlak, *Rough Sets – Theoretical aspects of reasoning about data,* Boston, MA, Kluwer Academic Publishers, 1991, Boston Dordrecht, London.

[13] Jerzy Stefanowski and Alexis Tsoukias *On the Extension of Rough Sets under Incomplete Information,* ed by N. Zhong, A. Skowron, S. Ohsuga, RSFDGCrC, Springer-Verlag Berlin Heidelberg, 1999, 73-82.

[14] Jerzy Grzymala-Busse, *Rough Set Approach to Incomplete Data*, Artificial Intelligence and Soft Computing - ICAISC 2004, vol. 3070, 50-55.

[15] Zdzislaw Pawlak, *Inference Rules and Decision Rules*, Artificial Intelligence and Soft Computing - ICAISC 2004 7th International Conference, Zakopane, Poland, June 2004, 102-108.

[16] Zdzislaw Pawlak, *Rough sets and data analysis,* Fuzzy Systems Symposium, Soft Computing in Intelligent Systems and Information Processing, Dec 11-14, 1996, 1-6.

[17] Zeinali, Yasha and Story, Brett A. 'Competitive Probabilistic Neural Network'. 1 Jan. 2017 : 105 – 118.

[18] Biau, G., Scornet, E. A random forest guided tour. *TEST* **25,** 197–227 (2016).

[19] Suthaharan S. (2016) Support Vector Machine. In: Machine Learning Models and Algorithms for Big Data Classification. Integrated Series in Information Systems, vol 36. Springer, Boston, MA.

[20] Wei Liu, Sanjay Chawla, David A. Cieslak, and Nitesh V. Chawla Proceedings of the 2010 SIAM International Conference on Data Mining. 2010, 766-777.

[21] Wei Liu, Sanjay Chawla, David A. Cieslak, and Nitesh V. Chawla Proceedings of the 2010 SIAM International Conference on Data Mining. 2010, 766-777.

[22] McCallum, Andrew. "Graphical Models, Lecture2: Bayesian Network Represention" . Retrieved 22 October 2019.

[23] Patrick G. Clark, Cheng Gao and Jerzy W. Grzymala-Busse. A comparison of two MLEM2 rule induction algorithms applied to data with many missing attribute values. Proceedings of the DBKDA, the 8-th International Conference on Advances in Databases, Knowledge, and Data Applications, Lisbon, Portugal, June 26–30, 2016, 60–65.

[24] Jerzy W. Grzymala-Busse and Teresa Mroczek. A comparison of two approaches to discretization: multiple scanning and C4.5. Proceedings of the 6-th International Conference on Pattern Recognition and Machine Learning, Warsaw, Poland, June 30–July 2, 2015, Springer Verlag 2015, Lecture Notes in Computer Science 9124, 44–53.

[25] Patrick G. Clark, Cheng Gao and Jerzy W. Grzymala-Busse. A comparison of mining incomplete and inconsistent data. Proceedings of the ICIST 2016, 22nd International Conference on Information and Software Technologies, Druskininkai, Lithuania, October 13–15, 2016. Springer-Verlag, CCIS 639, 2016, 414–425.

[26] Patrick G. Clark and Jerzy W. Grzymala-Busse. A comparison of classification systems for rule sets induced from incomplete data by probabilistic approximations. Proceedings of the ALLDATA 2015, the First International Conference on Big Data, Small Data, Linked Data and Open Data, Barcelona, Spain, April 19–24, 2015, 46–51.

[27] Patrick G. Clark, Cheng Gao and Jerzy W. Grzymala-Busse. A comparison of

four classification systems using rule sets induced from incomplete data sets by local probabilistic approximations. Proceedings of the ISMIS 2017, the 23-rd International Symposium on Methodologies for Intelligent Systems, Warsaw, Poland, June 26–29, 2017, ed. by M. Kryszkiewicz et al., Springer Verlag, Berlin-Heidelberg, Lecture Notes in Artificial Intelligence 10352, 282–291.

[28] Jerzy W. Grzymala-Busse. An empirical comparison of rule induction using feature selection with the LEM2 algorithm. Proceedings of the IPMU 2012, the 14-th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Catania, Italy, July 9–13, 2012. Advances in Computational Intelligence 297, Springer Verlag, 270–279.

LIST OF FIGURES

Fig 4.1 Sample result file
Fig 4.2 Sample ruleset
Fig 4.3 Sample statistics

LIST OF TABLES

Table 2.1 An example of the decision table

Table 2.2 An incomplete decision table

Table 4.1 Error rates

APPENDIX A

LEM2 algorithm

The procedure for LEM2 algorithm is given below: [6]

```
Procedure LEM2
(input: a set B,
output: a single local covering $\mathbb{T}$ of set B);
begin
    G := B;
    $\mathbb{T}$ := Ø;
    while G ≠ Ø
        begin
        T := Ø;
        T(G) := {t | [t] ∩ G ≠ Ø};
        while T = Ø or [T] Ç B
            begin
                select a pair t ∈T(G) with the highest attribute priority,
                if a tie occurs, select a pair t ∈T(G) such that |[t] ∩ G| is
                maximum;
                if another tie occurs, select a pair t∈T(G) with the smallest
                cardinality of [t]; if a further tie occurs, select first pair;
                T := T ∪ {t};
                G := [t] ∩ G;
                T(G) := {t | [t] ∩ G ≠ Ø};
                T(G) := T(G) – T;
        end {while}
        for each t in T do
                if [T – {t}] ⊆ B then T: = T – {t};
        $\mathbb{T}$ := $\mathbb{T}$ ∪ {T};
        G := B – $\cup_{T \in T}[T]$
    end {while};
    for each T in $\mathbb{T}$ do
      if $\cup_{S \in \mathbb{T} - T}[S] = B$ then $\mathbb{T}$:= $\mathbb{T}$ – {T}
end {procedure}.
```

APPENDIX B

Implementation details:

For the experiments, a file containing dataset will be given as input to the program, which creates ten folds from the initial data, which will be used for validating the experiment. From each data set thus formed, the program will calculate the number of attributes and cases available to classify. Then the program will check if the data consists of missing values and numerical values, after performing discretization, attribute-value pairs are created for the entire collection of data. From this attribute-value pairs, using the alpha value give during initial inputs, the program will create lower and upper approximations. From these approximations, a file with rule sets will be created. These input data and output rule files are taken as inputs by the second program, which checks the rules and calculates error rate based on the factors that are being defined by the users. The error rates calculated for all the ten-folds will be written to an error.txt file with their average. This process is continued for all the datasets, and the resulting error rates are compared. This program was created using Java programming language.