Chapter 6

GENERATION AND VALIDATION OF RULES FOR A DATASET WITH MISSING ATTRIBUTE VALUES

6.1 Introduction

The rough set philosophy is founded on the assumption that there is some information (data, knowledge) associated with every object of the universe of discourse [110]. The input data files are usually in the form of a table known as a decision table or information table. In this table, each column represents one attribute representing some feature of the examples, and each row represents an example by all its attribute values. There are basically two types of attributes in a decision table one is called condition attributes and other is called decision attribute. Condition attributes are called independent variables of the decision table and decision attribute is called dependent variable [104]. On the basis of the values of condition attributes the decision is specified. In most of the cases each decision table has only one decision attribute, there may be any number of condition attributes in the decision table. The domain of each attribute may be either symbolic or numerical. We assume that all the attributes of input data are symbolic. Numerical attributes, after discretization, become symbolic as well and a symbolic attribute may be converted to a numeric attribute. For each example, there is a decision value associated with it. The set of all examples with the same decision value is called a concept [111].

However, in real life applications, some input data in the information table is usually missing or in other words we can say that decision tables are incompletely specified and some attribute values are frequently absent. Handling missing attribute values in rough set theory is a big challenge [79]. The concept of rule induction from incomplete data set is first given by Grzymala-Busse [54]. There are basically two main reasons for the attribute values to be missing; either they were “lost” means originally the attribute value
was known and due to some unknown reason it is erased or the “do not care condition” that is the values were not recorded since they were originally irrelevant [54]. The first rough set approach to missing attribute values, when missing values fall in the category of lost values, was described in 1997 in [55]. On the other hand, decision tables in which all missing attribute values are due to “do not care” condition, were described for the first time in [56].

In general, incomplete decision tables are described by characteristic relations, in a similar way as complete decision tables are described by indiscernibility relations [48, 51, 55]. In rough set theory, for complete decision tables, once the indiscernibility relation is fixed and the concept (a set of cases) is given, the lower and upper approximations become unique.

For incomplete decision tables, a given characteristic relation and concept, there are three different possibilities to define lower and upper approximations, called singleton, subset, and concept approximations. Singleton lower and upper approximations have been studied in [47-49, 55, 74, 163]. It is observed in [55] that singleton lower and upper approximations are not applicable in data mining. It has been proved that rules generated from concept lower and upper approximation are most significant.

### 6.2 Incomplete Decision Table

Sometimes, some of the condition attribute values may be missing from the information table available to us. When some of the condition attribute values are not specified in the table then that table is known as incomplete decision table [48]. All missing attribute values may be denoted either by “?” or by “*”, where in the lost values are denoted by “?” and “do not care” condition values are denoted by “*”. For each case, at least one attribute value must have been specified [50]. We consider a particular case, where some of the condition attribute values are lost but all decision attribute values are available in the information table. In table 6.1 the lost condition attribute values are donated by “?”.
<table>
<thead>
<tr>
<th>Cases</th>
<th>Condition Attributes</th>
<th>Decision Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blood Pressure</td>
<td>Chest Pain</td>
</tr>
<tr>
<td>1</td>
<td>High</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>?</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>?</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>High</td>
<td>?</td>
</tr>
<tr>
<td>5</td>
<td>?</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Normal</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 6.1 An Incomplete Decision Table

In the Decision table 6.1 there are three condition attributes and one decision attribute.
Condition attributes = \{Blood Pressure, Chest Pain, Cholesterol\}
Decision attribute = \{Heart Problem\}
All the possible values for the condition and decision attributes are given in table 6.2.

<table>
<thead>
<tr>
<th>Attributes</th>
<th>Nominal Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Condition Attributes</td>
<td>Blood Pressure, Chest Pain, Cholesterol</td>
</tr>
<tr>
<td>Decision Attribute</td>
<td>Heart Problem</td>
</tr>
<tr>
<td></td>
<td>Yes, No</td>
</tr>
<tr>
<td></td>
<td>High, Low</td>
</tr>
</tbody>
</table>

Table 6.2 Value Sets for the Attributes
6.3 Different Approaches to Handle Missing Attribute Values

There are following nine approaches to handle the missing attribute values [49]:

1. **Most Common Attribute Value.** It is one of the simplest methods to deal with missing attribute values. The CN2 algorithm [21] uses this idea. The value that occurs most frequently for a particular attribute is selected to be the value for all the unknown values of the attribute.

2. **Concept Most Common Attribute Value.** The most common attribute value method does not pay any attention to the relationship between the condition attribute and the decision values. The concept most common attribute value method is a restriction to the first method to the concept, that is frequency is considered amongst all examples with the same value for the decision attribute [74]. This time the value of the attribute, which occurs the most common within the concept is selected to be the value for all the unknown values of the attribute. This method is also called maximum relative frequency method, or maximum conditional probability method (given concept).

3. **C4.5.** This method is based on entropy and splitting the example with missing attribute values to all concepts [126].

4. **Method of Assigning All Possible Values of the Attribute.** In this method, an example with missing attribute value is replaced by a set of new examples, in which the missing attribute value is replaced by all possible values of the attribute [54]. If we have some examples with more than one unknown attribute value, we will do our substitution for one attribute first, and then do the substitution for the next attribute, etc., until all missing attribute values are replaced by all possible values of the attribute.

5. **Method of Assigning All Possible Values of the Attribute Restricted to the Given Concept.** The method of assigning all possible values of the attribute is not related with a concept [53]. This method is a restriction of the method of assigning all possible values
of the attribute to the concept as explained above with the difference that here one set of values shall be substituted for one concept..

This method is the simplest one just ignores the examples which have at least one unknown attribute value, and then use the rest of the table as input to the successive learning process.

7. Event-Covering Method. This method, described in [15], is also a probabilistic approach to fill in the unknown attribute values. By event-covering we mean covering or selecting a subset of statistically interdependent events in the outcome space of variable-pairs, disregarding whether or not the variables are statistically independent [47].

8. A Special LEM2 Algorithm. A special version of LEM2 that works for unknown attribute values omits the examples with unknown attribute values when building the block for that attribute [55]. Then, a set of rules is induced by using the original LEM2 method.

9. Method of Treating Missing Attribute Values as Special Values. In this method, the unknown attribute values are dealt by using a totally different approach: rather than trying to find some known attribute value as its value, the “unknown” itself is treated as a new value for the attribute that contain missing values.

6.4 Generation of Rules for Missing Attribute Values
Our strategy is to complete the incomplete decision table with the help of the Method of ‘Assigning All Possible Values of the Attribute’ discussed above i.e. the missing attribute values are replaced by all possible values of attribute from the domain of the attribute. In decision table 6.1 the missing attribute value of Blood Pressure is replaced by attribute value ‘High’ first and then with value ‘Normal’. Similarly for the attribute Chest Pain replacing the missing attribute value by ‘Yes’ first and then with the value ‘No’. For missing Cholesterol attribute first the value ‘High’ is used and then the value ‘Low’ is used. Table 6.3 is obtained after substituting Blood Pressure = ‘High’ in place of ‘?’ for Blood Pressure attribute. Similarly substituting the value Chest Pain = ‘Yes’ in place of ‘?’ for Chest Pain attribute and Cholesterol = ‘High’ by ‘?’ for Cholesterol attribute in table 6.1. After substituting these values in table 6.1 we get a complete decision table 6.3.
<table>
<thead>
<tr>
<th>Cases</th>
<th>Blood Pressure</th>
<th>Chest Pain</th>
<th>Cholesterol</th>
<th>Heart Problem(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>High</td>
<td>Yes</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>Yes</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>High</td>
<td>No</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>High</td>
<td>Yes</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>High</td>
<td>Yes</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>Normal</td>
<td>No</td>
<td>High</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 6.3  Complete Decision Table

We can change the nominal values of the table 6.3 into numerical values. Changing the nominal values of the table 6.3 into numerical values by substituting the following values and getting table 6.4.

High → 2  
Normal → 1  
Low → 0  

and  
Yes → 1  
No → 0
Cases  | Blood Pressure | Chest Pain | Cholesterol | Heart Problem(D) |
---|---|---|---|---|
1 | 2 | 1 | 2 | 1 |
2 | 2 | 1 | 2 | 1 |
3 | 2 | 0 | 2 | 0 |
4 | 2 | 1 | 2 | 1 |
5 | 2 | 1 | 0 | 0 |
6 | 1 | 0 | 2 | 0 |

Table 6.4 Numerical Values of Table 6.3

6.4.1 Finding Reduct and Core of Complete Information Table

Reduct and core of the above complete information table are generated by using Rose2 Software. Reduct is a minimal subset of attributes that enables the same classification for elements of the universe as the whole set of attributes, whereas core contains indispensible attributes. Core and Reduct of the table 6.4 are shown in fig 6.1 and fig 6.2.
Now we generate rules for table 6.4 with the help of Rose2 software.

Rule 1. (Blood_Pressure = 1) => (Heart_Problem = 0); [1, 1, 33.33%, 100.00%] [1, 0]

   [{6}, {}]

Rule 2. (Chest_pain = 0) => (Heart_Problem = 0); [2, 2, 66.67%, 100.00%] [2, 0]

   [{3, 6}, {}]

Rule 3. (Cholesterol = 0) => (Heart_Problem = 0); [1, 1, 33.33%, 100.00%] [1, 0]

   [{5}, {}]

Rule 4. (Chest_pain = 1) & (Cholesterol = 2) => (Heart_Problem = 1); [3, 3, 100.00%, 100.00%] [0, 3]

   [{}, {1, 2, 4}]

**END

Now in decision table 6.1 substituting the missing attribute values of attributes Blood Pressure and Chest Pain by Blood Pressure = ‘Normal’ and Chest Pain = ‘No’ and replacing the missing attribute value of Cholesterol = ‘Low’. Another complete information table 6.5 is obtained. After changing the nominal values of table 6.5 into numerical values table 6.6 is obtained.
<table>
<thead>
<tr>
<th>Cases</th>
<th>Condition Attributes</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blood Pressure</td>
<td>Chest Pain</td>
</tr>
<tr>
<td>1</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Normal</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Normal</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Normal</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Normal</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 6.5  Complete Information System

<table>
<thead>
<tr>
<th>Cases</th>
<th>Condition Attributes</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Blood Pressure</td>
<td>Chest Pain</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 6.6  Numerical Values of Information System

Core and Reduct of the information system of table 6.6 are shown in fig. 6.3 and fig. 6.4.
Fig 6.3  Core of the Information System or Table 6.6

Fig 6.4  Reduct of the Information System or Table 6.6
Generating the Rules for table 6.6

Rule 1. (Blood_Pressure = 1) & (Chest_pain = 0) => (Heart_Problem = 0); [2, 2, 66.67%, 100.00%] [2, 0] [{3, 6}, {}]

Rule 2. (Chest_pain = 0) & (Cholesterol = 0) => (Heart_Problem = 0); [2, 2, 66.67%, 100.00%] [2, 0] [{3, 6}, {}]

Rule 3. (Blood_Pressure = 2) => (Heart_Problem = 1); [2, 2, 66.67%, 100.00%] [0, 2] [{}, {1, 4}]

Rule 4. (Cholesterol = 2) => (Heart_Problem = 1); [2, 2, 66.67%, 100.00%] [0, 2] [{}, {1, 4}]

**END

Some rules are common for table 6.4 and table 6.6 and these rules are more important and significant than other existing rules for heart problem dataset. Common rules are:

Rule 1. (Blood_Pressure = 1) => (Heart_Problem = 0); [1, 1, 33.33%, 100.00%] [1, 0] [{6}, {}]

Rule 2. (Chest_pain = 0) => (Heart_Problem = 0); [2, 2, 66.67%, 100.00%] [2, 0] [{3, 6}, {}]

Rule 3. (Cholesterol = 2) => (Heart_Problem = 1); [2, 2, 66.67%, 100.00%] [0, 2] [{}, {1, 4}]

**END

After changing the Numerical values into Nominal values the rules are

Rule 1. (Blood_Pressure = Normal) => (Heart_Problem = No);

Rule 2. (Chest_pain = No) => (Heart_Problem = No);

Rule 3. (Cholesterol = High) => (Heart_Problem = Yes);