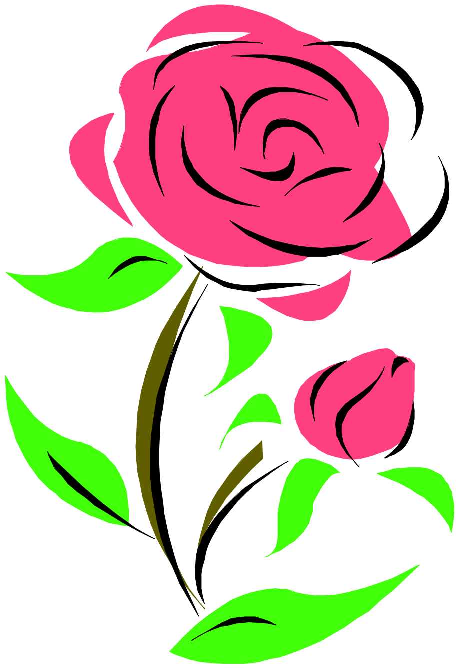
2.0

User's Guide



ROSE 2

Rough Set Data Explorer

for Windows 95/98/NT

Some details may differ from the final version of software.

1

Introduction

Human knowledge based on experience (e.g. concerning decision making in a specific field) is often recorded in a structure called an *information system*. The information system contains information about particular cases (objects, states, observations, events, …) in terms of their *attributes* (features, variables, characteristics, symptoms, …). The set of attributes is consisted of two kinds of attributes. First group (called *condition* attributes) concerns results of some tests or measurements, data from observation, anamnesis, symptoms of cases, states etc. The other group (called *decision* attributes) concerns some expert's decisions, diagnoses, classified results of a treatment etc. The question about *cause-effect dependencies* between these two groups of attributes (e.g. in a form of decision rules) is the most interesting in the analysis of information systems. The other essential problem refers to a possible *reduction* of all superfluous attributes and cases in the information system, a representing dependencies between values of attributes in the form of *decision rules.*

The answer to these question may be more difficult if the attributes may have different nature, i.e. some of them may have a *qualitative* character (i.e. their domain consists of nominal or ordered linguistic codes) while other may be of *quantitative* character (i.e. may be defined on numerical interval). Such situation often exists in practical applications. In medical practice, for instance, a set of attributes usually consists of a mixture of qualitative of quantitative ones. However, very often the values of quantitative attributes are not interpreted by physicians as precise numbers, but they are compared to some established norms. Thus, they treat them as results e.g., at high, average and small level. In other words, the quantitative attributes are often translated for practical interpretation into some qualitative terms.

Additional difficulties in the analysis of such information systems may refer to possible *ambiguity* in the definition of objects' classification caused by *inconsistencies* in

objects' description. The inconsistency means that some objects are described by the same values of condition attributes and are assigned to different values of decision attributes. It generally prevents the precise definition of objects classification, and looking for dependencies between values of condition and decision attributes.

The most convenient tool for the analysis of such data sets is the ***rough sets approach***, introduced by Pawlak in [1]. This approach is particularly useful in cases of ambiguous data and when the data are not convenient for traditional, ordinary statistical methods.

Using the rough sets approach to analyse information system leads among other possible results to:

- evaluation of importance of condition attributes for definition of objects' classification by values of chosen decision attributes.,
- reduction of all redundant objects and attributes in the information systems so as to get the minimum subsets of attributes ensuring a satisfactory approximation of objects' classification,
- creation of models of the most representative objects for particular decision classes (i.e. classes of a classification defined by values of decision attributes),
- representation of the dependencies between the minimum subset of condition attributes and decision ones in a form of a set of decision rules.

The set of decision rules and the information about the most significant attributes for objects' classification may be treated as *representation* of the *knowledge* acquired by a specialist on all known cases/objects contained in his information system.

It must be stressed that this form of the representation is free of all redundancies, so typical for real data bases, which cover a lot of important factors of experience.

Moreover, the results obtained by the rough sets approach are expressed in the *form similar* to *human, natural language*. So, these representations are very easy to understand by the user/analyst and allow to build the justification or explanation for derived data analysis conclusions. It gives also the possibility for the analyst to control the data analysis process in a simple way. Such possibilities are not typically offered by traditional data analysis techniques. The rough set approach is characterized by low computational efforts and a possibility of interaction with a user in a dialogue mode. Results are not adulterated by subjective indirect evaluations or arbitrary chosen operators.

The existing results of the applications have proved the suitability of the rough sets approach to the analysis of several information systems in different domains (cf. [2]-[7]).

The microcomputer program ROSE is a software implementation of the rough set approach with some extensions.

In the next section basic notions of the rough sets theory are presented. Then, in section 3, the detailed guide for a user is given. A computational example of a usage of ROSE is discussed in the last section.

2

Basic concepts of the rough sets theory and rule induction technique

A short presentation of basic notions of the rough sets theory created by Pawlak is given in this section. More information can be found in references listed in this manual.

The observation, that we cannot distinguish objects on the basis of imprecise information about them is the starting point of the rough sets theory. In other words, imprecise information causes indiscernibility of objects in terms of available data. Indiscernibility of objects generally prevents the precise classification of objects. Indiscernibility relation is used to define two main operations on data: lower and upper approximation of a set. Approximation space and approximation of a set or a family of sets (particularly classification) in this space, are two next concepts of the rough sets theory with a great practical importance. Using lower and upper approximation of a set (or a classification) we can define an accuracy and a quality of approximation. These are numbers from interval [0,1] which define how exactly we can describe the examined set (or classification) of objects using available information. The concept of an information system is used to construct the approximation space. It enables representation of data in a useful form of a table.

2.1. Information system

By an information system we understand the 4-tuple $S=\langle U,Q,V,\rho\rangle$, where *U* is a finite set of *objects*, *Q* is a finite set of *attributes*, $V=\bigcup_{q\in Q} V_q$ and V_q is a *domain* of the attribute *q*, and $r: U\times Q\rightarrow V$ is a total function such that $\rho(x,q)\in V_q$ for every *qÎ Q, xÎ U*, called *information function*. Any pair *(q,v), qÎ Q,* $v\in V_q$ is called *descriptor* in S.

The information system is in fact a finite data table, columns of which are labelled by attributes, rows are labelled by objects and the entry in column *q* and row *x* has the value $\rho(x,q)$. Each row in the table represents the information about an object in *S*.

For example, a data file concerning patients suffering from a certain disease is an information system (cf. [2]-[7]). An example of an information system is presented in Table 1.

Q U	p	Q	r	s
x_1	1	0	1	0
x_2	2	1	0	1
x_3	0	1	0	0
x_4	2	0	0	1
x_5	0	1	2	0
x_6	0	1	0	0
x_7	2	1	0	1
x_8	1	0	1	0
x_9	0	1	2	0
x_{10}	0	1	1	0

Table 1. Exemplary information system

2.2. Indiscernibility relation

Let $S=\langle U,Q,V,\rho\rangle$ be an information system and let *PÍ Q, x,yÎ U*. We say that *x* and *y* are *indiscernible* by the set of attributes *P* in *S* (denotation $x\,\tilde{P}\,y$) if $r(x,q)=r(y,q)$ for every *qÎ P*. Equivalence classes of relation $\tilde{P}$ are called *P-elementary sets* in *S*. *Q-elementary sets* are called *atoms* in *S*. Atoms and {p}-elementary sets in the information system from Table 1. are as follows:

{p}-elementary sets	**atoms**	
$X_1=\{x_3, x_5, x_6, x_9, x_{10}\}$	$Z_1=\{x_1, x_8\}$	$Z_5=\{x_4\}$
$X_2=\{x_1, x_8\}$	$Z_2=\{x_2, x_7\}$	$Z_6=\{x_{10}\}$
$X_3=\{x_2, x_4, x_7\}$	$Z_3=\{x_3, x_6\}$	
	$Z_4=\{x_5, x_9\}$	

The family of all equivalence classes of relation $\tilde{P}$ on *U* is denoted by P^*. $\mathrm{Des}_p(X)$ denotes the description of equivalence (P-elementary set) $X \in P^*$;

i.e.: $Des_p(X) = \{(q,v) \mid r(x,q) = v, \underset{x \in X}{\forall}, \underset{q \in P}{\forall}\}$.

2.3. Approximations of sets

In order to evaluate, how well the set $\{\mathrm{Des}_p(X), \quad X \in P^*\}$ describes objects of a certain set Pawlak [1] has introduced the following concepts.

Let P⊆Q and Y⊆U. The *P-lower approximation* of Y, denoted by $\underline{P}Y$, and the *P-upper approximation* of Y, denoted by $\overline{P}Y$, are defined as:

$$\underline{P}Y = \bigcup X \quad \{X \in P^* \text{ and } X \subseteq Y\}$$

$$\overline{P}Y = \bigcup X \quad \{X \in P^* \text{ and } X \cap Y \neq \varnothing\}$$

The *P-boundary* (*P doubtful region of classification*) is defined as

$$\mathrm{Bn}_p(Y) = \overline{P}Y - \underline{P}Y.$$

Set $\underline{P}Y$ is the set of all elements of *U* which can be certainly classified as elements of *Y*, employing the set of .attributes *P*; Set $\overline{P}Y$ is the set of elements of *U* which can be possibly classified as elements of *Y*, using the set of attributes *P*. The set $Bn_p(Y)$ is the set of elements which cannot be certainly classified to *Y* using the set of attributes *P*.

With every subset Y⊆U, we can associate an *accuracy of approximation* of set Y by P in S, or in short, accuracy of Y, defined as .

$$m_p(Y) = \frac{card(\underline{P}Y)}{card(\overline{P}Y)}$$

Let us consider the information system presented in Table 1. Let $Y=\{x_1, x_4, x_5, x_7, x_9\}$ and $P=Q=\{p, q, r, s\}$. The approximations are:

$\underline{Q}Y=Z_4 + Z_5 = \{x_4, x_5, x_9\}$,

$\overline{Q}Y=Z_1 + Z_2 + Z_4 + Z_5 = \{x_1, x_2, x_4, x_5, x_7, x_8, x_9\}$,

$\mathrm{Bn}_Q(Y)=Z_1 + Z_2 = \{x_1, x_8\}+\{x_2, x_7\}$

and the accuracy is $\mu_Q(Y)=3/7=0.429$

For better understanding, we consider other example which is more typical for medical applications. Let us assume that Table 2. represents a medical information system describing 10 patients $x_1, ..., x_{10}$. Attributes c_1 i c_2 denote results of some medical tests – c_1 in three degree scale and c_2 in two degree scale. Let *d* denote a classification of patients done by a physician; value 0 means healthy ones and value 1 means ill ones.

Q / U	c₁	c₂	d
x_1	0	0	0
x_2	0	0	0
x_3	0	1	1
x_4	0	1	0
x_5	1	0	1
x_6	1	1	1
x_7	2	0	0
x_8	2	0	1
x_9	2	1	0
x_{10}	2	1	0

Table 2. An example of medical information system.

Equivalence classes created on a base of a subset of attributes $C=(c_1, c_2)$, it means $\{c_1, c_2\}$-elementary sets are as follows $\{x_1, x_2\}$, $\{x_3, x_4\}$, $\{x_5\}$, $\{x_6\}$, $\{x_7, x_8\}$, $\{x_9, x_{10}\}$. The physician's classification generates two {d} elementary sets:

- $\{x_1, x_2, x_4, x_7, x_9, x_{10}\}$ - *healthy patients*
- $\{x_3, x_5, x_6, x_8\}$ - *ill patients.*

Let *Y* denotes the set of ill patients, then $\underline{C}Y = \{x_5\} \cup \{x_6\}$, and $\overline{C}Y=\{x_5\}\cup\{x_6\}\cup\{x_3, x_4\}\cup\{x_7, x_8\}$. The accuracy of approximation *Y* is $\mu_C(Y) = 2/6 = 0.33$.

It is easy to notice that patients x_3 and x_4 represents an ambiguity in the information system. It is impossible to distinguish them basing only on values of attributes c_1 i c_2 (x_3 and x_4 are indiscernible by the set *C*), however the physician assigned x_3 to ill patients and x_4 to healthy ones. The pair x_7 and x_8 represents a similar case.

2.4. Rough classification

Let *S* be an information system, *PÍQ*, and let *X*={*Y*$_1$, *Y*$_2$, ...,*Y*$_n$} be a *classification* of *U*, i.e. $Y_i \cap Y_j = \varnothing, \quad \forall i, j \le n, i \ne j$ and $\bigcup_{i=1}^{n} Y_i = U$. Y_i are called *classes* of *X*. By P-lower (P-upper) approximation of *X* in S we mean sets

$\underline{P}X = \{\underline{P}Y_1, \underline{P}Y_2, \ldots, \underline{P}Y_n\}$ and $\overline{P}X = \{\overline{P}Y_1, \overline{P}Y_2, \ldots, \overline{P}Y_n\}$ respectively. The coefficient

$$\gamma_P(X) = \frac{\sum_{i=1}^{n} \mathrm{card}(\underline{P}Y_i)}{\mathrm{card}(U)}$$

is called the *quality of approximation of classification X* by set of attributes, or in short, *quality of classification X*. It expresses the ratio of all P-correctly classified objects to all objects in the system.

2.5. Reduction of attributes

We say that the set of attributes *RÍQ depends* on the set of attributes *PÍQ* in *S* (denotation $P \to R$) if $\tilde{P} \subseteq \tilde{R}$. Discovering dependencies between attributes enables the reduction of the set of attributes. Subset *PÍQ* is *independent* in *S* if for every *P'Ì P*, $\tilde{P}' \supset \tilde{P}$; otherwise subset *PÍQ* is *dependent* in *S*.

If we consider the information system from Table 1., we can find the exemplary dependency {q,r,s}→p. Such the dependencies may be easy checked looking at the {q,r,s}-elementary and {p}-elementary sets:

{q,r,s}-elementary sets	{p}-elementary sets
$\{x_1, x_8\}$	$\{x_3, x_5, x_6, x_9, x_{10}\}$
$\{x_2, x_7\}$	$\{x_1, x_8\}$
$\{x_3, x_6\}$	$\{x_2, x_4, x_7\}$
$\{x_5, x_9\}$	
$\{x_4\}$	
$\{x_{10}\}$	

Notice, that each {q,r,s}-elementary set is a subset of one of {p}-elementary sets, so $\{q,r,s\} \subset \{\tilde{p}\}$. Other examples of dependencies are {p,q,r}→s and p→s.

In practical applications we are interested in reducing those attributes which are redundant in *S* (i.e. we are interested in obtaining so called reducts). Subset *PÍQ* is a reduct of *Q* in *S* iff *P* is the greatest independent set in *Q*.

To find a reduct one can also use the quality of approximation of classification γP(*X)*.

The least minimal subset which ensures the same quality of classification as the set of all attributes is a *reduct* in S. It is sometimes called a *minimal set of attributes.*

Let us notice that an information system may have more than one reduct/minimal set. Intersection of all reducts/minimal sets is called the *core*.

The core is a collection of the most significant attributes for the classification in the system.

Let us notice, that in the information system from Table 1 are two reducts {p,q,r}, {q,r,s} and a core {q,r}.

2.6. Decision rules

The *ROSE* system generates decision rules using a modified version of the LEM2 algorithm \cite{Grzymala}. This algorithm consists of a group of induction algorithms, which are focused on inducing either the so called *discriminant description* of each decision class or a description very close to it. Considering a set of objects or learning examples, a rule description of a decision class is said to be *discriminant* (according to \cite{mich83}) when it is complete (each *positive example* - i.e. belonging to the decision class - must be recognized as belonging to the class) and *consistent* (each *negative example* - i.e. belonging to any of the other classes - cannot be recognized as belonging to the class). The discriminant description is usually assumed to be minimal, i.e. removing any of its rules would result in the description which is no longer complete.

A crucial point in these algorithms is the way of performing the search within the space of possible rules in order to select 'good' rules and combines them into the required subset. Since explicit characterization and exhaustive exploration of this space lead to combinatorial difficulties, *heuristic approaches*, avoiding these characterization and exploration steps, are usually implemented. All these algorithms follow the same general *greedy* scheme:

```
begin
  while all the positive examples are not covered do
  begin
        construct a "best" rule;
        remove the positive examples covered by this rule
  end
  take the disjunction of these rules as the required description
end
```

where *"best"* is defined according to the *preference criterion* considered.

The minimal set of rules is usually applied for classification aims.

ROSE can be also used as a kind of the Classification Support System based on the set of decision rules. The classification problem consists in determining the assignment of an object, described by a set of attributes, to one of a priori known decision classes. The classification support is based on the use of decision rules derived from a set of learning examples. The description of the classified object is matched against the condition parts of the decision rules.

The matching may lead to one of the three situations:

i. the object matches one or several rules indicating the same decision class,
ii. the object matches one or several rules indicating different decision classes,
iii. the object does not match any of the rules.

ROSE supports the user in all these situations. In case (i), the recommendation is univocal. In the case of ambiguous matching (ii), the user is informed about the total strength of all matching rules with respect to each suggested decision class. Case (iii) is handled by looking for the rules 'nearest' (according to same distance measure) to the description of the classified object.

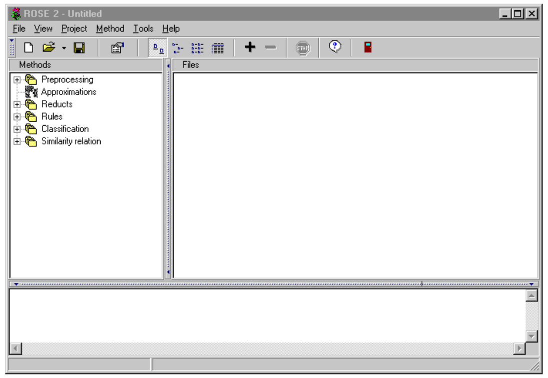
3

Example of use

In this chapter an example of information system is described together with its simple analysis using *ROSE*. This description is a practical illustration of a usage of this program presented in the next chapter. Assumption is made that *ROSE* is already installed and ready for use (for instructions how to do this see paragraph 4.2).

3.1 Running *ROSE*

If you didn't enter any customized settings during program setup, you will find *ROSE* icon in the **Start** menu in group **Programs | ROSE2**. Please click on this icon. A momen later you will see main *ROSE* window on your desktop. It should resemble the one bellow.



Now you have *ROSE* running with an empty untitled project. *ROSE* window is divided into 4 parts: topmost *menubar* and *toolbar*, middle *Methods tree* and *Project window* and bottom *Console window*. There is also a *status line* in the bottom.

If you are familiar with use of Microsoft Windows Explorer™ you will find *ROSE* easy to use.

Following is step by step description of an exemplary data analysis using some of *ROSE* features.

3.2 Information system

Let us consider an information system built from 17 objects and 8 condition attributes (denoted by A1-A8). The set of objects is divided into 3 classes (decision attributes denoted by Dec). This system is shown below.

	A_1	**A_2**	**A_3**	**A_4**	**A_5**	**A_6**	**A_7**	**A_8**	**Dec**
x_1	2	2	3	1	2	2	2	1	1
x_2	2	1	3	1	2	3	1	2	1
x_3	2	2	2	1	2	3	2	1	2
x_4	1	2	1	1	1	1	2	1	3
x_5	2	2	2	1	2	2	2	1	1
x_6	3	1	3	1	2	3	1	2	2
x_7	1	1	2	2	2	1	1	2	1
x_8	3	2	3	1	2	2	2	1	2
x_9	2	1	3	1	2	1	1	2	2
x_{10}	1	3	1	2	1	2	1	2	3
x_{11}	1	2	2	1	1	1	2	1	1
x_{12}	1	2	2	2	1	2	1	1	3
x_{13}	3	2	3	1	2	2	2	1	2
x_{14}	1	3	4	1	1	2	2	1	3
x_{15}	2	2	3	1	2	2	2	1	2
x_{16}	1	3	1	2	1	2	1	2	1
x_{17}	1	2	2	1	1	1	2	1	1

Table 3. Exemplary information system

3.3 Data file

As the first step you should create a file containing data from information system.

You have now two choices – you may use preadsheet-like editor (see chapter ?.?) or any plain text editor. For this example we will focus on the second option.

There is a simple text editor built into the ROSE environment. If you want to use it, select **Tools | Text** editor menu command or simply press **Ctrl** and **E** on the keyboard.

Now enter text shown below (you may also open a file *example.isf*, which can be found in *EXAMPLES* subdirectory). Detailed information about file format can be found in Appendix A.

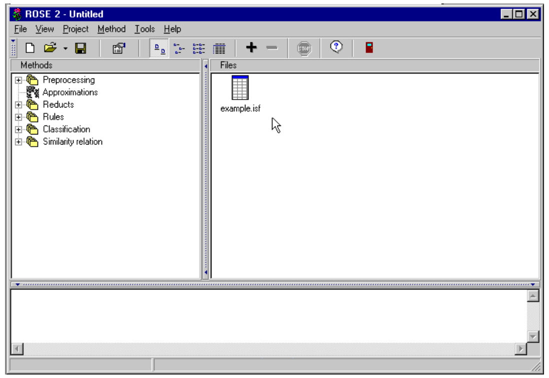
```
**ATTRIBUTES
A1: (numbercoded)
A2: (numbercoded)
A3: (numbercoded)
A4: (numbercoded)
A5: (numbercoded)
A6: (numbercoded)
A7: (numbercoded)
A8: (numbercoded)
Dec: [1,2,3]
decision: Dec
**EXAMPLES
2,2,3,1,2,2,2,1,1
2,1,3,1,2,3,1,2,1
2,2,2,1,2,3,2,1,2
1,2,1,1,1,1,2,1,3
2,2,2,1,2,2,2,1,1
3,1,3,1,2,3,1,2,2
1,1,2,2,2,1,1,2,1
3,2,3,1,2,2,2,1,2
2,1,3,1,2,1,1,2,2
1,3,1,2,1,2,1,2,3
1,2,2,1,1,1,2,1,1
1,2,2,2,1,2,1,1,3
3,2,3,1,2,2,2,1,2
1,3,4,1,1,2,2,1,3
2,2,3,1,2,2,2,1,2
1,3,1,2,1,2,1,2,1
1,2,2,1,1,1,2,1,1
**END
```

Now, save the file as example.isf. At this moment you are able to add this file to your project.

3.4 ROSE project

Use **Project | Add file to Project** menu command or press **Ins** on the keyboard. You will see a typical file open dialog. Please select your newly created file.

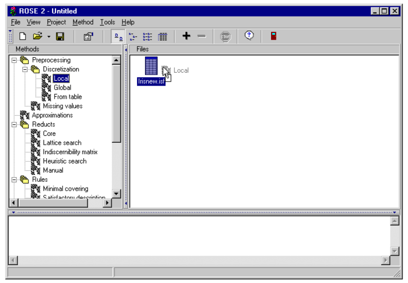
A new icon should appear in the right pane of *ROSE*.



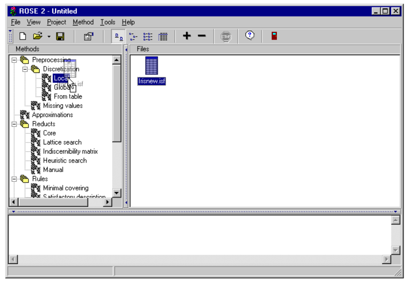
What you have now is a project containing one data file. For future use save it to disk using **File | Save project** menu command or press **Ctrl** and **S** on the keyboard.

Now you are able to use analysis methods with your data. There are several methods of invoking them:

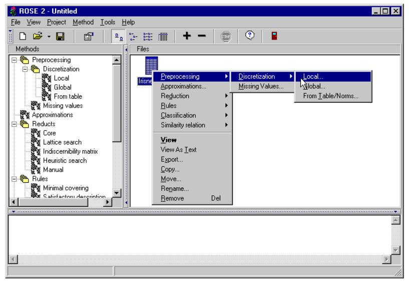
- dragging a method from the tree in the left-hand pane to the data file in the right-hand pane,



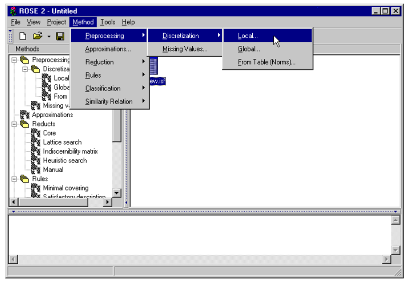
- dragging a data file from right-hand pane to the method in the left-hand pane,



- right-clicking on the data file and selecting method from the popup menu,



- using **Method** menu command.



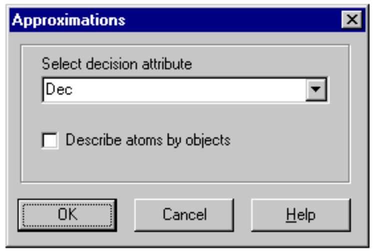
Because our example is simple we will skip the preprocessing phase and start with the basic rough set analysis methods.

3.5 Searching for atoms

One of the first steps of data analysis using rough set theory is searching for approximations of decision classes (atoms). To do this you should use **Approximations**

method in one of the earlier mentioned ways. If you are not familiar with extensive mouse operations please use **Method** menu command.

Now you will see an input dialog for searching for atoms.


Approximations

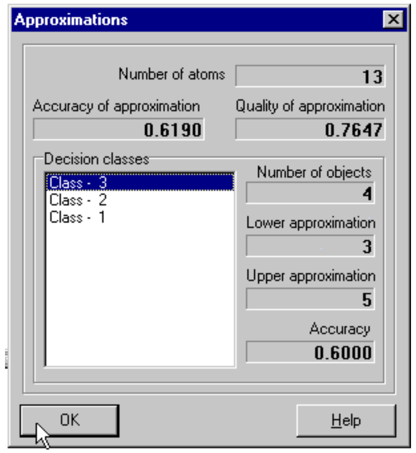
Select decision attribute: Dec

☐ Describe atoms by objects

Here you can select decision attribute and decide whether you want atoms to be described by objects (in the result file you will see what objects constitute an atom).

For now just press OK.

Briefly you will see external module execution dialog and then, when the computations are completed a result dialog:


Approximations

Number of atoms: 13

Accuracy of approximation: 0.6190

Quality of approximation: 0.7647

Decision classes:
Class - 3
Class - 2
Class - 1

Number of objects: 4

Lower approximation: 3

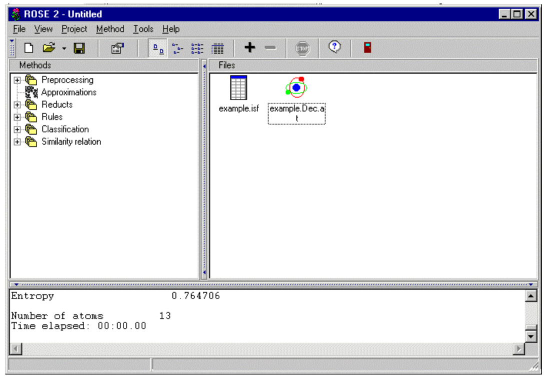
Upper approximation: 5

Accuracy: 0.6000

As you can see, there are 13 atoms in our exemplary information system, quality of approximation equals 0.7647, and accuracy of approximation equals 0.619. Now you can select one of decision classes that is of interest to you by clicking the mouse on its name (e.g. Class 3). For example Class 3 contains 4 objects, with lower approximation containing 3 and upper approximation 5 objects. Accuracy equals 0.6.

There is some new contents in the *Console window*. It includes now all the output from the external module that was schown in the execution window.

You may have noticed that a new icon appeared in the right-hand pane. It represents the result file containing detailed information about approximations.



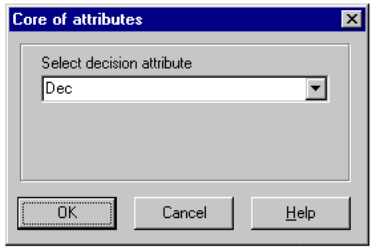
```
Entropy                0.764706

Number of atoms        13
Time elapsed: 00:00.00
```

If you want to see it all you may open it by right-clicking and selecting **View as text** from a pop-up menu. Double clicking the icon or selecting **View** command will show the same result dialog as mentioned earlier.

3.6 Core

Next step of an analysis is to generate core of attributes in order to find the most meaningful of them. Please use **Core** method from the **Reducts** group.

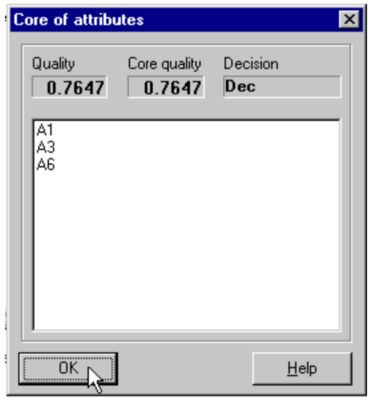
You will see a decision attribute selection dialog similar to shown earlier.



Select decision attribute: Dec

Because in our example we have only one decision attribute just press the OK button.

When core is generated the following dialog will appear:



Quality	Core quality	Decision
0.7647	0.7647	Dec

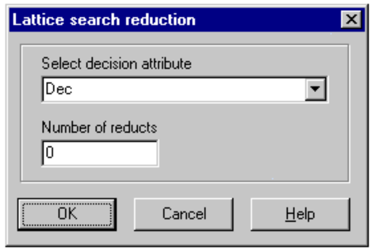
A1
A3
A6

As you can see attributes A1, A3 and A6 belong to the core of attributes. Because quality of core is the same as the quality of the whole set of attributes, it means, there is only one reduct, and it contains the same attributes as the core.

There also appeared a new icon in the project window associated with the core search results. The remarks are the same as with the approximations results.

3.7 Reducts

Although we stated that there is only one reduct, for instructional purpose we will look for reducts using one of the available methods. Please use **Lattice search** method from the **Reducts** group. You will be asked for decision attribute and reducts limit in the following dialog:

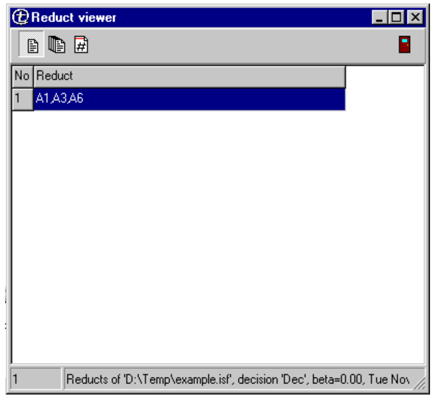


Select decision attribute: Dec

Number of reducts: 0

Because there can be many reducts, and finding all of them can be time consuming, you may want to set a limit on the number of reducts. When you enter 0, all reducts will be found.

When calculations are done you will see a reduct viewer window open.



No	Reduct
1	A1,A3,A6

1 Reducts of 'D:\Temp\example.isf', decision 'Dec', beta=0.00, Tue Nov

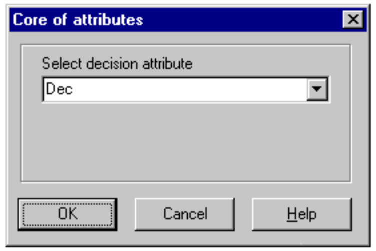
You may view reducts in the single view (all in one column), attribute view (in a grid with attributes in columns) and as an attribute statistics (occurrence of attributes in all reducts).

There is also a new icon in the right-hand pane, which enables viewing results in the already described way.

3.8 Rules

One of the most important elements of data analysis is rule generation. We will use one of the available methods – minimal covering rules (LEM2). Please select **Minimal covering** method from the **Rules** group.

You will be asked to select a decision attribute.



Select decision attribute: Dec

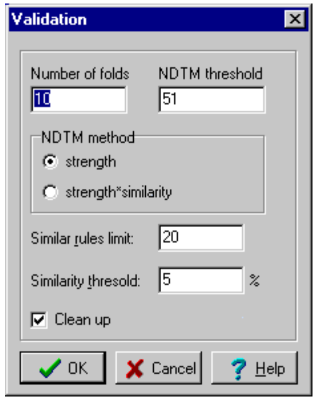
When rule generation is finished a new icon will appear in the project window. Currently you are able to view the results only in text mode.

3.9 Classification

Usually data analyst wants to know what are the generated rules worth, i.e. how good they can classify objects.

In ROSE system you are able to do the classification test using any of the available methods for rule generation. So we will test the same method as above.

Please select **Minimal covering validation** method from the **Classification** group. At first you will sea the same dialog as at the rule generation method. But then a new dialog appears:



Number of folds: 10
NDTM threshold: 51
NDTM method: [✓ strength] strength*similarity
Similar rules limit: 20
Similarity thresold: 5 %
[✓ Clean up]

ROSE system uses L-metric type classifier, which has several parameters:

- NDTM threshold – threshold used if object matches one or several rules indicating different decision classes to determine majority class.

- NDTM method – method used in non-deterministic matching to determine which method should be used to comparing matched rules (one of two methods – strength of rules or strength multiply by similarity factor - can be choosen).
- Similarity rules limit – maximal number of similar rules used to classifiaction of one object.
- Similarity threshold – minimal value of similarity factor of rule, which can be used to classifiaction of one object.

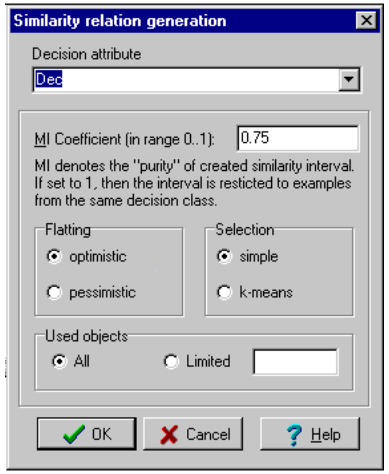
When finished click OK. When computations are finished result file is opened in the editor.

3.10 Similarity relation

One of the extensions of the rough set theory in *ROSE* system is similarity relation.

First of all you have to create similarity relation based on your data. Please use **Relation generation** method from **Similarity relation** group.

You will see the following dialog:


Similarity relation generation

Decision attribute: Dec

MI Coefficient (in range 0..1): 0.75

MI denotes the "purity" of created similarity interval. If set to 1, then the interval is resticted to examples from the same decision class.

Flatting: [✓ optimistic] pessimistic

Selection: [✓ simple] k-means

Used objects: [✓ All] Limited

In *'Decision attribute'* box you should select a decision attribute, which will be used as an active one during calculation of similarity relation.

The other parameters affect the calculation process. The '*MI*' coefficient determines "purities" of created similarity intervals. If it is set to 1.0, then created intervals can only contain objects from the same class. If it smaller than 1.0, then the similarity intervals can contain objects from different classes. The default value is 0.75.

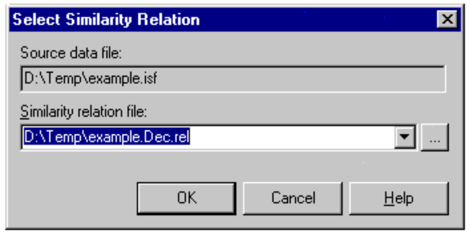
The '*Flatting*' parameter determines how the similarity intervals are processed to ensure their monotonicity. If you select the '*optimistic*' option, then intervals will be extended if necessary, otherwise – if you select the '*pessimistic*' option, then they will be narrowed if it is necessary to satisfy the monotonicity.

The '*Selection*' and '*Used objects*' parameters determine the way of selecting objects, which are in turn used as starting points of the similarity intervals. If '*Used objects*' is set to '*All*', then all objects from the data set are used as starting points of similarity intervals, and the setting of '*Selection*' has no effect. If you set '*Used objects*' to '*Limited*' and enter the number of objects, then the selection is based on the value of '*Selection*'. Is '*Selection*' is set to 'simple', then every *k*-th (where *k* in the number of objects entered by the user) object is used as a starting point of the interval. Otherwise the *k*-means algorithm is used and the centroids of generated clusters are used as starting points of similarity intervals.

When you clik OK a new icon will appear in the project window.

Only when you have a relation file in your project you will be able to use similarity based methods.

For example use **Similarity approximations** from the **Similarity relation** group. You will see dialog shown bellow:


Select Similarity Relation

Source data file: D:\Temp\example.isf

Similarity relation file: D:\Temp\example.Dec.rel

The '*Source data file*' field contains a name of the data file, for which approximations will be calculated. It is a read-only field, supplied for information and cannot be changed.

In the '*Similarity relation file*' box you should select an appropriate file with similarity relation, i.e. the file created in the previous step. You can choose this file among all similarity relation files contained in current project or browse for the file yourself by clicking the '...' button.

4

Use of ROSE

4.1. Basic information

ROSE is a microcomputer software designed to analyse data by means of the rough set theory. It is runing on microcomputers using Intel x86 family processors and compatible, working under 32-bit graphical user interface operating systems, such as Windows 95/98™ or Windows NT™. The microcomputer with at least Pentium processor is recomended.

ROSE has a modular architecture – it consists of integrated environment and external executable modules. Some of them are used only by the computational engine, others (like data editor) may be run standalone. Please note that for correct work system must know the location of its modules. Please see *Options* section.

4.2 Program installation

Program is distributed on ISO9660 compatible **CD-ROM**. In order to install software on your computer you should execute *setup.exe* found in the INSTALL folder of a CD-ROM. Setup will ask you for destination folder (Program Files\Rose2 is default) and will copy all necessary files to your hard disk, including some system libraries and examples. Also registry entries will be made. When setup is finished you will find Rose folder in start menu.

Uninstallation is possible using Windows Control Panel and selecting Add/Remove Programs.

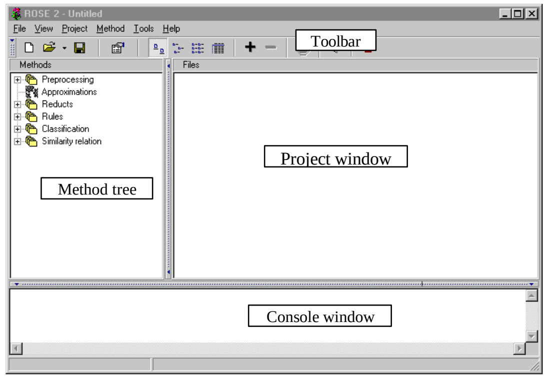
4.3 Files used by *ROSE*

During work with *ROSE* software you will encounter several file types. These include:

- *ROSE* project files have *ros* extension. These files contain information about your projects.
- data files with *isf* extension. It is one of more important files. It contains data to be analysed by the system and must be provided by the user. It is plain text file although it has to comply with file format described further in Appendix A.
- atom files with *atm* extension, described in Appendix B.
- core files with *cor* extension.
- reduct files with *rdf* extension.
- rule files with *rlf* extensions, described in Appendix C.
- log files with *log* extension – they contain information about the execution of modules.
- similarity relation files with *rel* extension – containing information about similarity relation.
- discretization norm files with *dis* extension – containing information about discretization norms.

4.4 Program conventions

If you ever used any Windows program, especially if you use Windows Explorer, you should be quite familiar with using *ROSE*. When you run ROSE you will see following window appear on your desktop.



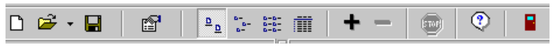
It is divided into four main parts:

- menu bar with toolbar – here you select commands using mouse or shortcut keys,
- method tree – this is expandable tree showing all currently available data analysis methods; if there is a folder icon and plus sign by the group name it may be expanded by clicking on it; may be turned off.
- project window – here you organize all files in your project – input data files, intermediate data files and results,
- console window – shows results of execution of external engine modulesl; may be turned off.

All three windows are divided by split-bars, i.e. they can be resized.

All program functions can be accessed by using the menu, some of them by using toolbar buttons or shortcut keys and of course keyboard.

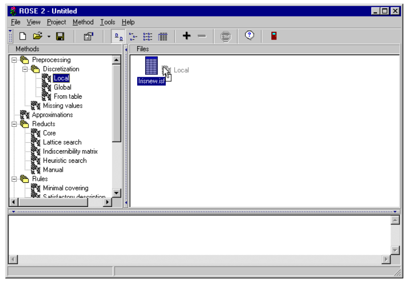
If you wonder what is the meaning of the toolbar buttons, here you can see it:



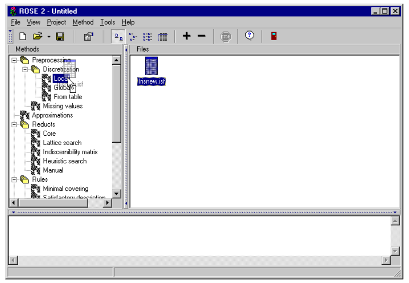
From the left to right: *new project, open project (with MRU list), save project, project properties, large icons, small icons, list, details, add to project, remove from project, stop external execution, help, exit.*

There are four possibilities to use a method with a data file:

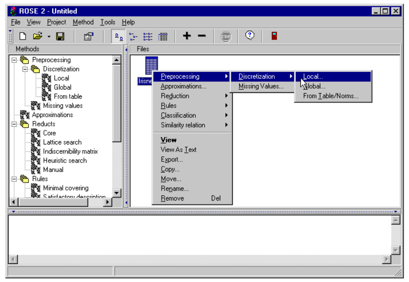
- dragging a method from the method tree to the data file in project window,



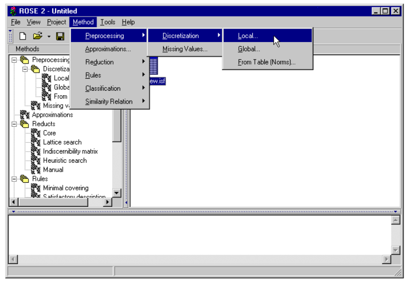
- dragging a data file from project window to the method tree,



- right-clicking on the data file and selecting method from the popup menu,



- using **Method** menu command (a data file must be selected).



4.5 Menu layout

```
-- File
|-- New Project
     |-- Open Project
     |-- Save Project
     |-- Save Project As
     |-- Recent Projects
     |-- Exit
-- View
     |-- --File View Type
                |-- Large icons
                |-- Small icons
                |-- List
                |-- Details
     |-- Show file extensions
     |-- Output
     |-- Method
     |-- Refresh
-- Project
     |-- Add file to project
     |-- Import file to project
     |-- Remove file from project
     |-- Export file
     |-- View file
     |-- View file as text
     |-- Copy file
     |-- Move file
     |-- Rename file
     |-- Properties
```

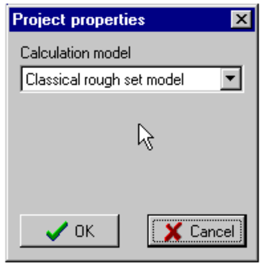
```
-- Method
     |-- -- Preprocessing
          |-- Discretization
               |-- Local
               |-- Global
               |-- From norms
          |-- Missing values
     |-- Approximations
     |-- Reduction
          |-- Core
          |-- Lattice search
          |-- Indiscernibility matrix
          |-- Heuristic search
          |-- Manual
     |-- Rules
          |-- Minimal covering
          |-- Extended minimal covering
          |-- Satisfactory description
     |-- Classification
          |-- Minimal covering validation
          |-- Satisfactory description validation
     |-- Similarity relation
          |-- Similarity relation
          |-- Similarity approximations
          |-- Similarity minimal covering
-- Tools
     |-- Text editor
     |-- Data file editor
     |-- Options
-- Help
     |-- Index
     |-- About
```

Using projects

All data analysis in ROSE system must be done using projects. These are special files containing some information about data you analyse, and your preferences.

First thing you should do is create data file in ISF format (see Appendix A). Then you can create a new project. All project functions are gathered in **File** and **Project** menu. Here you can create new project (see section 3.3), open an existing project or set project options. Because opening projects is as easy as openning any files in windows applications (remember, that ROSE projects have *ROS* extension), we will describe only project options.

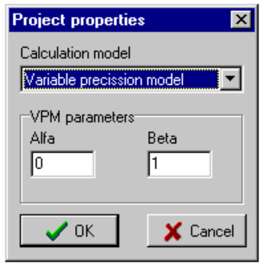
When you use **Project | Project options** menu command you will see following dialog:


Project properties

Calculation model: Classical rough set model

Here you can change computation model. There are two available:

- standard – classical rough set methodology is used,
- VPM – system uses Variable Precission Model, with parameters Alfa and Beta.


Project properties

Calculation model: Variable precission model

VPM parameters

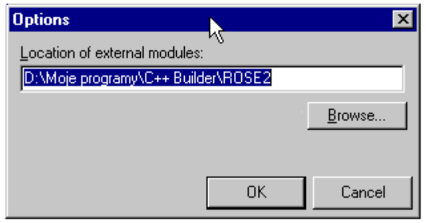
Alfa: 0

Beta: 1

Remember to save your project files when you finish your work or you will have to create them from the beginning.

System options

There are also some parameters that affect work of whole system. They can be acessed by **Tools | Options** menu command. When you do this following dialog is shown:


Options

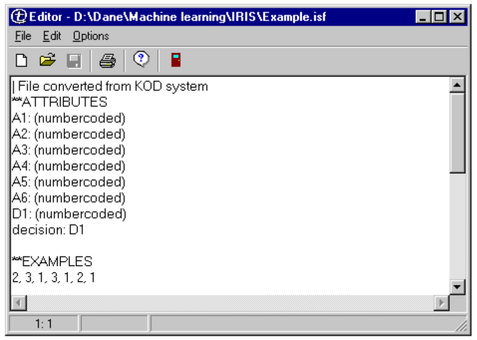
Location of external modules: D:\Moje programy\C++ Builder\ROSE2

There is an edit line that allows you to enter path, where all external modules can be found. It is set during installation, but in some cases (registry errors) it may be assigned to default *ROSE* folder. If you have problems running external methods please check this option first.

Viewers and editors

Text editor

Simple text editor is built into the *ROSE* environment. You may run it by using **Tools | Text** editor menu command or **Ctrl** and **E** shortcut key. It will also appear if you select **Text view** option.


Editor - D:\Dane\Machine learning\IRIS\Example.isf

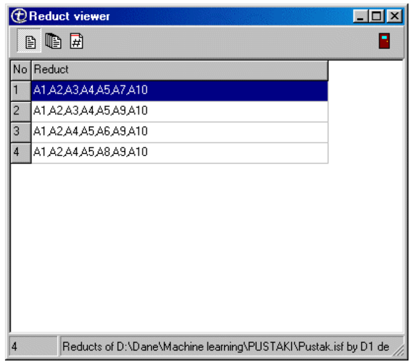
```
| File converted from KOD system
**ATTRIBUTES
A1: (numbercoded)
A2: (numbercoded)
A3: (numbercoded)
A4: (numbercoded)
A5: (numbercoded)
A6: (numbercoded)
D1: (numbercoded)
decision: D1

**EXAMPLES
2, 3, 1, 3, 1, 2, 1
```

You may enter and edit plain text files in this window. Beside typical file operations (open, save, new, save as), clipboard operations (cut, copy, paste), there is only an option to wrap lines (**Options | Wrap** lines menu command).

Reduct viewer

This viewer is meant to display the results of reduct generation in three possible ways:

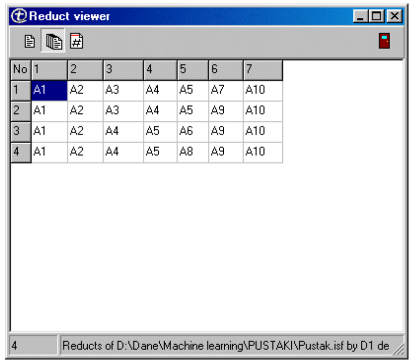
- simple view – all reducts are presented in one column,



No	Reduct
1	A1,A2,A3,A4,A5,A7,A10
2	A1,A2,A3,A4,A5,A9,A10
3	A1,A2,A4,A5,A6,A9,A10
4	A1,A2,A4,A5,A8,A9,A10

4 Reducts of D:\Dane\Machine learning\PUSTAKI\Pustak.isf by D1 de

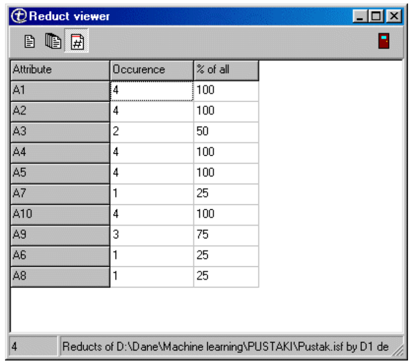
- attribute view – each column represents one attribute in reduct,



No	1	2	3	4	5	6	7
1	A1	A2	A3	A4	A5	A7	A10
2	A1	A2	A3	A4	A5	A9	A10
3	A1	A2	A4	A5	A6	A9	A10
4	A1	A2	A4	A5	A8	A9	A10

4 Reducts of D:\Dane\Machine learning\PUSTAKI\Pustak.isf by D1 de

- statistical view – occurrence statistics are shown for each attribute that belongs to any reduct.



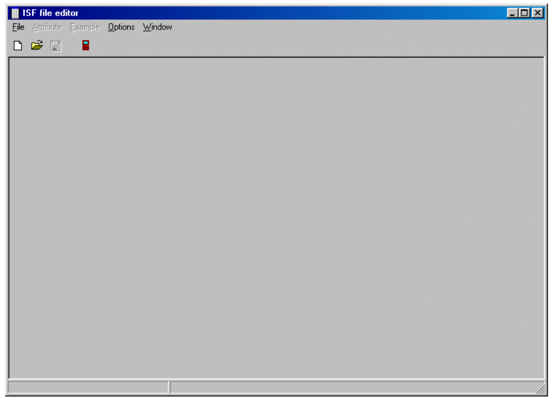
Attribute	Occurence	% of all
A1	4	100
A2	4	100
A3	2	50
A4	4	100
A5	4	100
A7	1	25
A10	4	100
A9	3	75
A6	1	25
A8	1	25

4 Reducts of D:\Dane\Machine learning\PUSTAKI\Pustak.isf by D1 de

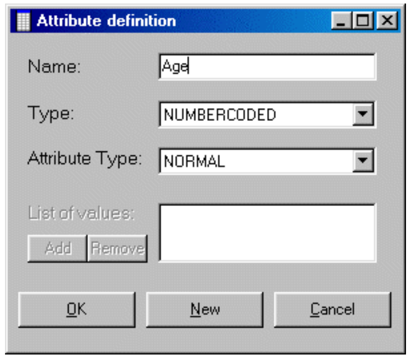
Three icons at the toolbar are used to select the type of view and the door icon is used to close the viewer.

Data editor

ROSE is equipped with visual editor for data files. It is spreadsheet like, so If you are familiar with any spreadsheet applications (like MS Excel™) you shouldn't have any problems using it. You may run it by using **Tools | Data editor** menu command or **Ctrl** and **I** shortcut key.



If you want to create new data file use **File | New** command. You will be shown a dialog for definition of attributes.



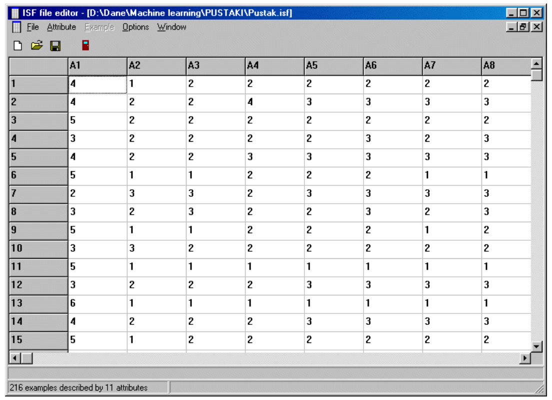
Name: Age

Type: NUMBERCODED

Attribute Type: NORMAL

Here you enter a name, select a type of attribute domain and role of attribute (conditional or decision). For the type '*enlisted*' you may also enter the list of allowed values using **Add** and **Remove** buttons. If you press **New** you will be able to add additional attributes.

After defining attributes you may enter data into the grid.

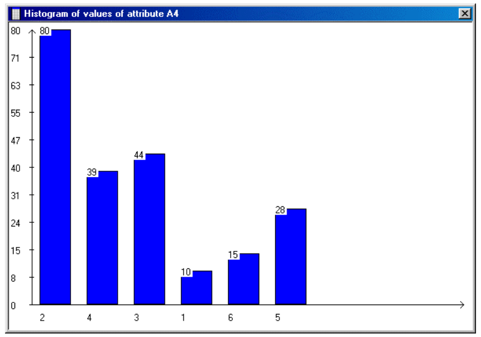


ISF file editor - [D:\Dane\Machine learning\PUSTAKI\Pustak.isf]

	A1	A2	A3	A4	A5	A6	A7	A8
1	4	1	2	2	2	2	2	2
2	4	2	2	4	3	3	3	3
3	5	2	2	2	2	2	2	2
4	3	2	2	2	2	3	2	3
5	4	2	2	3	3	3	3	3
6	5	1	1	2	2	2	1	1
7	2	3	3	2	3	3	3	3
8	3	2	3	2	2	3	2	3
9	5	1	1	2	2	2	1	2
10	3	3	2	2	2	2	2	2
11	5	1	1	1	1	1	1	1
12	3	2	2	2	3	3	3	3
13	6	1	1	1	1	1	1	1
14	4	2	2	2	3	3	3	3
15	5	1	2	2	2	2	2	2

216 examples described by 11 attributes

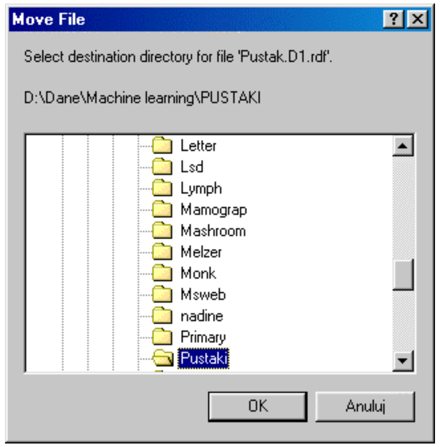
There are also some more advanced functions, like histogram calculation or manual discretization. For details check help file.



Managing project files

There are some functions available concerning management of the files contained in the current project. You may:

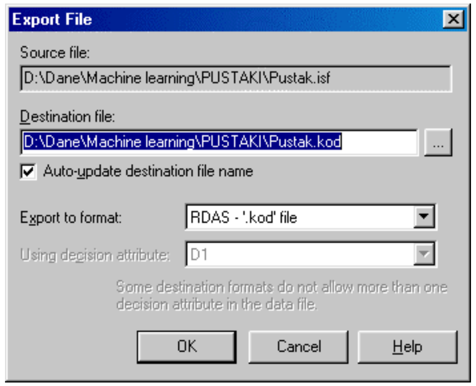
- move file to another folder – change its physical placement on the disk; **Project | Move file** or right click.



Select destination directory for file 'Pustak.D1.rdf'.

D:\Dane\Machine learning\PUSTAKI

- copy file – place a copy of a project file in another folder. The dialog is almost the same as for moving file; **Project | Copy file** or right click.
- export file – save data file in another format, e.g. for use in other applications; **Project |Export file** or right click.


Export File

Source file: D:\Dane\Machine learning\PUSTAKI\Pustak.isf

Destination file: D:\Dane\Machine learning\PUSTAKI\Pustak.kod

☑ Auto-update destination file name

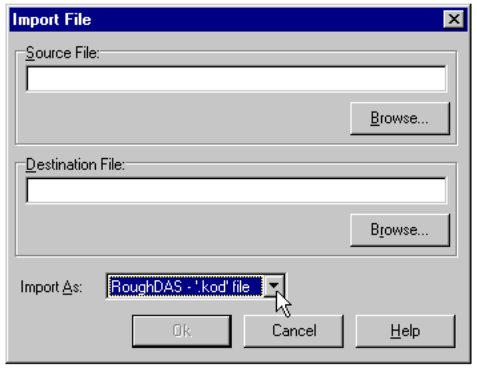
Export to format: RDAS - '.kod' file

Using decision attribute: D1

Some destination formats do not allow more than one decision attribute in the data file.

OK Cancel Help

- import file to project from another file formats; **Project | Import file to project** or right click.


Import File

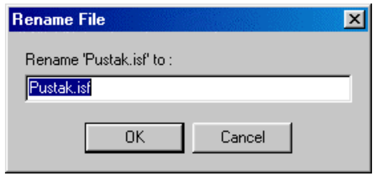
Source File:

Destination File:

Import As: RoughDAS - '.kod' file

Ok Cancel Help

- rename file; **Project | Rename file** or right click.


Rename File

Rename 'Pustak.isf' to : Pustak.isf

OK Cancel

Data analysis methods

All data analysis methods in ROSE system are divided into groups. There are five of them:

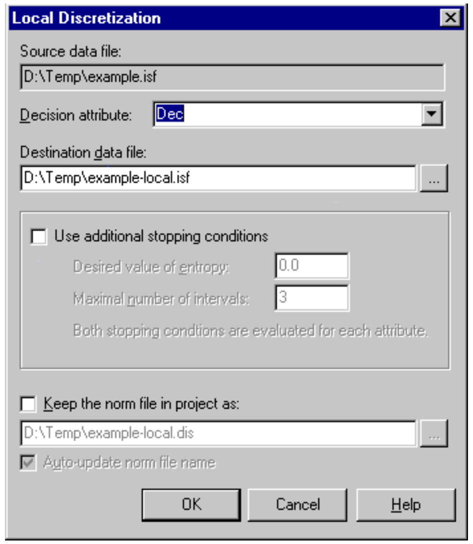
- Preprocessing – methods ment for preliminary data analysis and modifications, like discretization,
- Reducts – methods dealing with the reduction of attributes,
- Rules – methods used to generate decision rules,
- Classification – validation of decision rules,
- Similarity relation – methods using similarity relation approach.

One method is excluded from all groups – rough set approximations.

Following you will find description of all included methods with its associated dialog windows.

Local discretization

Preprocessing | Discretization | Local


Local Discretization

Source data file: D:\Temp\example.isf

Decision attribute: Dec

Destination data file: D:\Temp\example-local.isf

☐ Use additional stopping conditions

Desired value of entropy: 0.0

Maximal number of intervals: 3

Both stopping conditions are evaluated for each attribute.

☐ Keep the norm file in project as: D:\Temp\example-local.dis

☑ Auto-update norm file name

OK Cancel Help

It is used to discretize attributes with continuous domains into the ones with discrete domains. All such attributes in the source data file will be processed. It is an entropy-based method.

User must select one of the decision attributes and enter the name of the result file (there is a default filename). There are also some additional parameters:

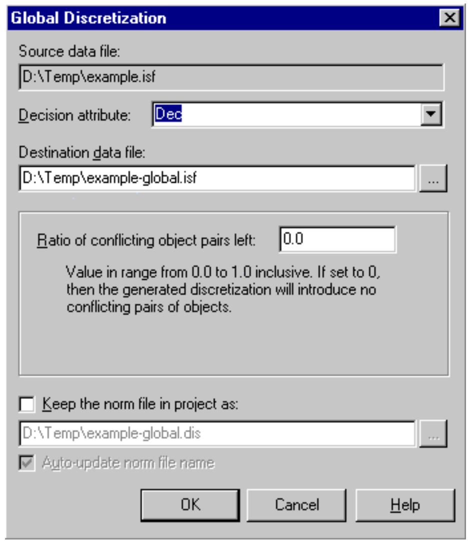
- Additional stopping conditions if checkbox next to them is active:
  - desired value of entropy <0,1> – if it is reached the method doesn't generate additional subintervals,
  - maximum number of intervals – method doesn't create more intervals than that.
- Keep the norm file in project as – if this is checked the file containing the discretization norms will be placed in project for future use or viewing. User should also give it a filename (default name is suggested).
- Auto update norm file name – if checked, the name of norm file will be automatically updated to reflect the changes in the destination file name.

Please note, that pressing '…' (browse) buttons will show file save dialogs for easier naming of the result files.

Running this method will result in new data file appearing in *project window*. There may also appear a discretization norms file according to selected options.

Global discretization

Preprocessing | Discretization | Global


Global Discretization

Source data file: D:\Temp\example.isf

Decision attribute: Dec

Destination data file: D:\Temp\example-global.isf

Ratio of conflicting object pairs left: 0.0

Value in range from 0.0 to 1.0 inclusive. If set to 0, then the generated discretization will introduce no conflicting pairs of objects.

☐ Keep the norm file in project as: D:\Temp\example-global.dis

☑ Auto-update norm file name

It is used to discretize attributes with continuous domains into the ones with discrete domains. All such attributes in the source data file will be processed. It is an entropy-based method.

User must select one of the decision attributes and enter the name of the result file (there is a default filename). There are also some additional parameters:

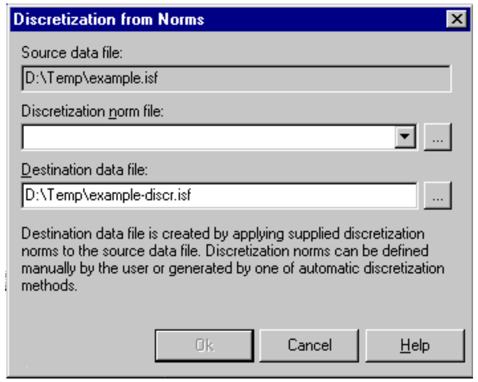
- Ratio of conflicting object pairs left <0,1> - changes stopping condition by allowing some conflicts in the method.
- Keep the norm file in project as – if this is checked the file containing the discretization norms will be placed in project for future use or viewing. User should also give it a filename (default name is suggested).
- Auto update norm file name – if checked, the name of norm file will be automatically updated to reflect the changes in the destination file name.

Please note, that pressing '…' (browse) buttons will show file save dialogs for easier naming of the result files.

Running this method will result in new data file appearing in *project window*. There may also appear a discretization norms file according to selected options.

Discretization from norms

Preprocessing | Discretization | From table


Discretization from Norms

Source data file: D:\Temp\example.isf

Discretization norm file:

Destination data file: D:\Temp\example-discr.isf

Destination data file is created by applying supplied discretization norms to the source data file. Discretization norms can be defined manually by the user or generated by one of automatic discretization methods.

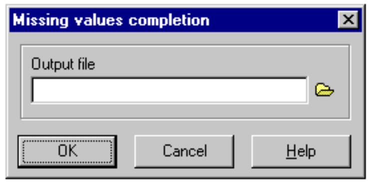
This method is used to discretize attributes with continuous domains into the ones with discrete domains. It is based on information provided by user or already obtained from automatic methods (contained in the discretization norm file).

User has to select a file containing discretization norms (Browse '…' button may be used) and give the name to the resulting file (there is default one).

Running this method will result in new data file appearing in the *project window*.

Missing values

Preprocessing | Missing values



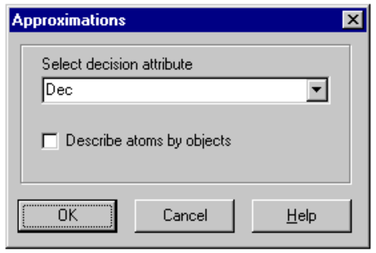
This method implements missing values completion using the most frequent value for the given attribute.

Its only parameter is output filename (Browse button may be used).

It will result in new date file in the *project window*.

Approximations

Approximations


Approximations

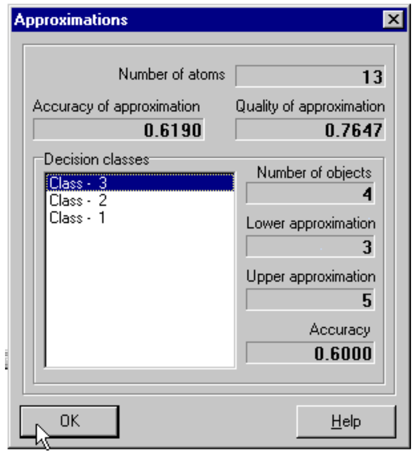
Select decision attribute: Dec

Describe atoms by objects

This method is used to generate rough set approximations. The method will depend on project options (see Project properties) – it will be either classical rough sets or variable precision model approach.

User must select one of the decision attributes and can decide, if atoms should be described by objects in the result file (see Appendix B).

Running it will result in new file appearing in *project window* and result dialog will be shown.


Approximations

Number of atoms: 13

Accuracy of approximation: 0.6190

Quality of approximation: 0.7647

Decision classes:
Class - 3
Class - 2
Class - 1

Number of objects: 4

Lower approximation: 3

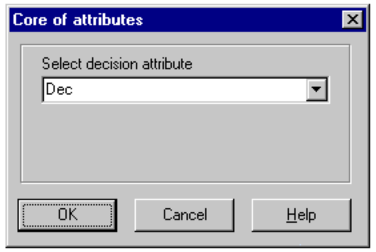
Upper approximation: 5

Accuracy: 0.6000

User may select any of the decision classes placed in the window on the left to see its approximation data on the right side.

Core

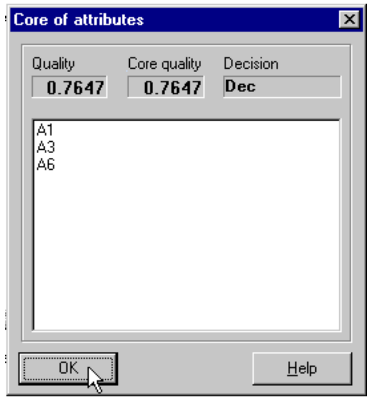
Reducts | Core



This method is used to obtain core of attributes, i.e. most meaningful attributes that are common part of all the reducts.

The only input parameter is a name of the decision attribute.

When completed a new file appears in project window and a result dialog opens.



Quality 0.7647
Core quality 0.7647
Decision Dec

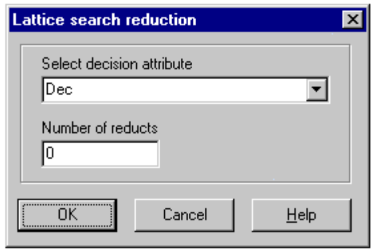
A1
A3
A6

This dialog contains following information:

- Quality – quality value for the whole set of attributes,
- Core quality – quality value for the attributes that belong to the core,
- Decision – used decision attribute,
- Listed attributes that construct the core.

Lattice search

Reducts | Lattice search

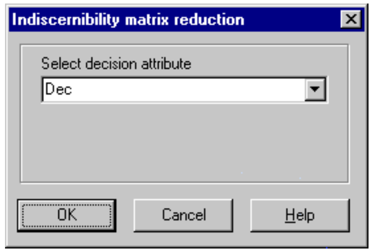


This method is used to obtain all (or limited number) reducts of the information table. It implements optimized lattice search method. Due to time consuming nature of this process user may limit the number of found reducts using *Number of reducts* input line. Decision attribute must be selected also.

When done a new icon appears in the project window and reduct viewer opens (for details see chapter describing it).

Indiscernibility matrix

Reducts | Indiscernibility matrix

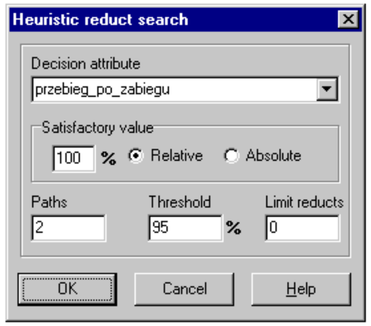


This method is used to obtain all reducts of the information table. Its implementation is based on a structure called indiscernibility matrix. It is very fast method so it should be preferred above lattice search. User must select one of decision attributes.

When done a new icon appears in the project window and reduct viewer opens (for details see chapter describing it).

Heuristic search

Reducts | Heuristic



Decision attribute: przebieg_po_zabiegu
Satisfactory value: 100 % [✓ Relative] Absolute
Paths: 2
Threshold: 95 %
Limit reducts: 0

This method is used to obbtain approximate reducts when it is impossible to generate all the exact ones. This heuristic works by adding attributes to the existing core by selecting most promising of them all and trying different paths. There are several input parameters:

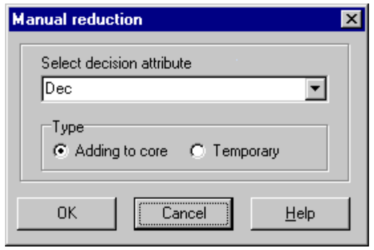
- decision attribute name,
- Limit reducts - boundary on number of pseudoreducts (in general this method may find even more pseudoreducts than real reducts and may be more time consuming, so it is plausible to reduce that number),
- Satisfactory value - value of quality of classification that pseudoreduct has to obtain,
- Relative - when selected it means that satisfactory value is given in percent, relatively to the quality of the whole set of attributes; if you enter 95 and the whole set has quality q, it means that pseudoreducts will have to have quality of classification at least 95% of q,
- Absolute - when selected it means that satisfactory value is given strictly, i.e. if you enter 0.98, it means that pseudoredects will have to have quality at least 0.98,
- Paths - heurisitc works by adding attributes to the core. At each node several attributes are taken into account. Paths describes how much is it. It has great influence on computational complexity of the method. If this

parameter is 1 there will be only one pseudoreduct. It is rather irrelevant to select more than 3 paths for larger sets,

- Threshold - this parameter is designed to cut some less meaning attributes when they are selected at the computational node. First Paths attributes, that are the best in the set, are taken into account, but it is probable, that some of these are quite meanigless. This parameter decides that if some of them give worse increase in quality of classification than the Threshold% of the best increase, they are not concerned as candidates for adding to the core.

Manual reducts

Reducts | Manual

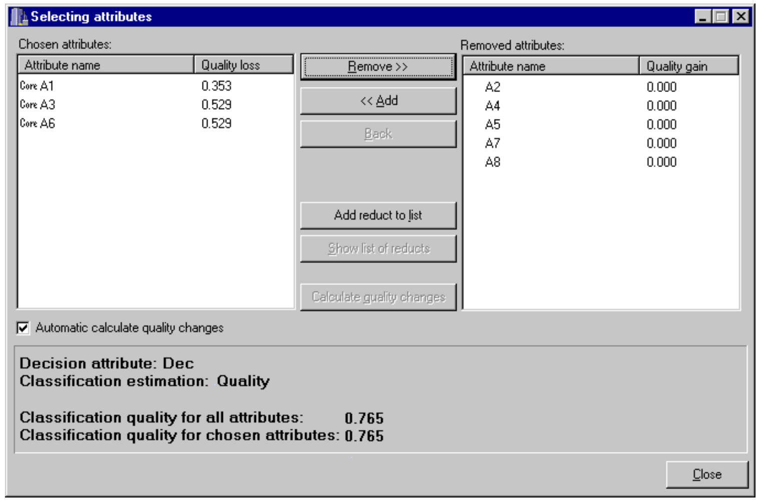


Select decision attribute: Dec
Type: [✓ Adding to core] Temporary

This method allows user to manually create reducts by adding attributes to the set and checking their quality of classification. Beside the selection of decision attribute there are two input options:

- Adding to core – core is used as the base for creating reducts,
- Temporary- user start with an empty set.

The manual reduct creation window is shown below.



Chosen attributes:

Attribute name	Quality loss
Core A1	0.353
Core A3	0.529
Core A6	0.529

Removed attributes:

Attribute name	Quality gain
A2	0.000
A4	0.000
A5	0.000
A7	0.000
A8	0.000

☑ Automatic calculate quality changes

Decision attribute: Dec
Classification estimation: Quality

Classification quality for all attributes: 0.765
Classification quality for chosen attributes: 0.765

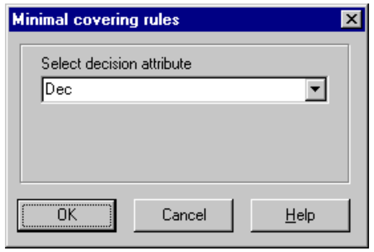
It is divided into two parts:

- left one contains attributes that belong to the current set; values next to them shows a predicted loss on quality of classification if such attribute should be removed,
- right one contains all attributes that can be added to the current set; values next to them show predicted increase in quality of classification if such attribute should be added.

Adding or removing is done either by use of the buttons in the middle section of the window or by double-clicking on one of the attributes. At the bottom short summary of the operations is presented.

Minimal covering

Rules | Minimal covering



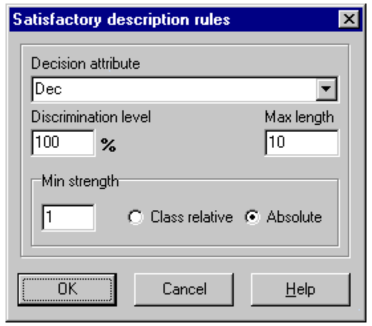
Select decision attribute: Dec

This method is used to generate a set of decision rules based on a minimal covering algorithm (minimal number of possibly shortest rules covering all the examples). It has only one parameter – name of the decision attribute.

When it is done a new file appears in the *project window*.

Satisfactory description

Rules | Satisfactory description



Decision attribute: Dec
Discrimination level: 100 %
Max length: 10
Min strength: 1 ○ Class relative ◉ Absolute

This method generates all possible rules satisfying the demands of the user. It has several input parameters:

- Decision attribute – name of the decision attribute,
- Discrimination level (in percent) –
- Max length – maximum length of generated rules,
- Min strength – minimal strength of generated rules (i.e. number of supporting objects),

- Class relative – min strength will be given in percent of all of class objects,
- Absolute – min strength is given in number of objects.

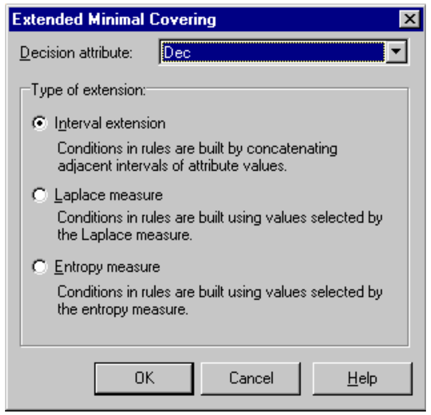
When done a new icon appears in the *project window*.

Extended minimal covering

Rules | Extended minimal covering

This option is used for generating decision rules using extended version of LEM2 algorithm. The modified LEM2 (so called ModLEM) can deal with continuous values of attributes without the need of prior discretization and generates rules with extended syntax of conditions (instead of simple tests in form *attribute = value*, ModLEM is able to generate tests in form: *attribute £ value*, *attribute ³ value* or *lower bound £ attribute £ upper bound*).

After having selected this method, the user is presented a dialog box shown below.



Extended Minimal Covering

Decision attribute: Dec

Type of extension:

[✓ Interval extension]
Conditions in rules are built by concatenating adjacent intervals of attribute values.

Laplace measure
Conditions in rules are built using values selected by the Laplace measure.

Entropy measure
Conditions in rules are built using values selected by the entropy measure.

OK | Cancel | Help

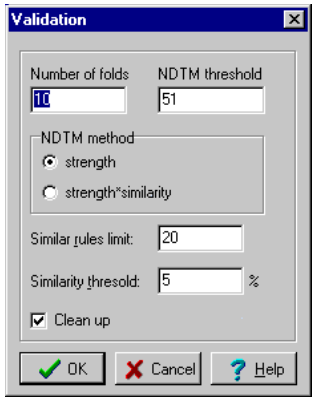
In the box 'Decision attribute' the user has to select an decision attribute, which will be considered during induction of decision rules. The user also has to select the type of ModLEM extension, as ModLEM contains three different extensions:

- Interval extensions – the conditions in rules are created by concatenating adjacent intervals of attribute values. The generated conditions have the syntax *lower bound £ attribute £ upper bound*.

- Laplace measure – the conditions in rules are built on cutting points, that minimize the value of Laplace measure. The generated conditions have the syntax *value £ attribute* or *attribute £ value.*
- Entropy measure – the conditions in rules are built on cutting points, that minimize the value of Entropy measure. The generated conditions have the syntax *value £ attribute* or *attribute £ value.*

Minimal covering and Satisfactory description validation

Classification | Minimal covering validation
Classification | Satisfactory description validation



Validation

Number of folds: 10
NDTM threshold: 51

NDTM method
[✓ strength]
strength*similarity

Similar rules limit: 20

Similarity thresold: 5 %

[✓ Clean up]

OK | Cancel | Help

There are some apparent parameters that user has to enter: input file containing examples to be classified, decision attribute and result file. The rest is explained here:

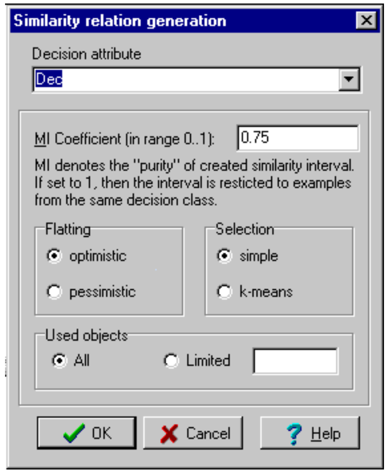
- NDTM threshold – threshold used if object matches one or several rules indicating different decision classes to determine majority class.
- NDTM method – method used in non-deterministic matching to determine which method should be used to comparing matched rules (one of two methods – strenght of rules or strength multiply by similarity factor - can be choosen).
- Similarity rules limit – maximal number of similar rules used to classifiaction of one object.

- Similarity threshold – minimal value of similarity factor of rule, which can be used to classifiaction of one object.

Similarity relation generation

Similarity relation | Relation generation

This option is used for generating similarity relation. The similarity relation replaces the classical indiscernibility relation and is especially useful for analyzing data containing continuous values, as it deals directly with continuous values making the discretization step redundant.


Similarity relation generation

Decision attribute: Dec

MI Coefficient (in range 0..1): 0.75

MI denotes the "purity" of created similarity interval. If set to 1, then the interval is resticted to examples from the same decision class.

Flatting: [✓ optimistic] pessimistic

Selection: [✓ simple] k-means

Used objects: [✓ All] Limited

The *'Decision attribute'* contains a decision attribute, which will be used as an active one during calculation of similarity relation.

The '*MI*' coefficient determines "purities" of created similarity intervals. If it is set to 1.0, then created intervals can only contain objects from the same class. If it smaller than 1.0, then the similarity intervals can contain objects from different classes. The default value is 0.75.

The '*Flatting*' parameter determines how the similarity intervals are processed to ensure their monotonicity. If the '*optimistic*' option is selected, then intervals will be extended if necessary to satisfy the monotonicity requirement, otherwise – if you select the '*pessimistic*' option, then they will be narrowed.

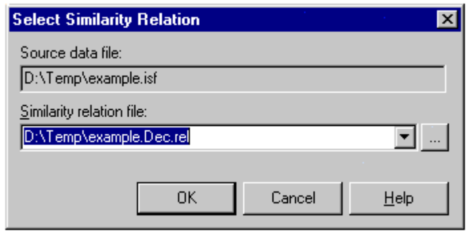
The '*Selection*' and '*Used objects*' parameters determine the way of selecting objects, which are in turn used as starting points of the similarity intervals. If '*Used objects*' is set to '*All*', then all objects from the data set are used as starting points of similarity intervals, and the setting of '*Selection*' has no effect. If the '*Used objects*' is set to '*Limited*' and the number of objectsis entered by the user, then the selection is based on the value of '*Selection*' parameter. Is '*Selection*' is set to 'simple', then every *k*-th (where *k* in the number of objects entered by the user) object is used as a starting point of the interval. Otherwise the *k*-means algorithm is used and the centroids of generated clusters are used as starting points of similarity intervals.

After successful generation of similarity relation, a new file is added to the project and ts icon appears in the project window.

Similarity approximations

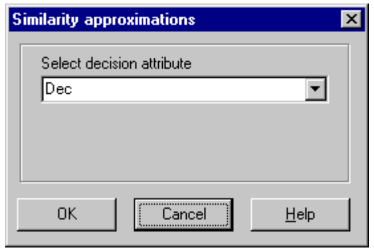
Similarity relation | Similarity approximations

This option is used for generating lower and upper approximations of decision classes. Generated approximations are based on similarity relation.


Select Similarity Relation

Source data file: D:\Temp\example.isf

Similarity relation file: D:\Temp\example.Dec.rel


Similarity approximations

Select decision attribute: Dec

After having selected this option, the user is presented the first dialog box of those presented above. The user has to select an appropriate relation file. The requested file can be chosen among all relation files contained in current project or the user can browse himself for the necessary file using '...' (browse) button.

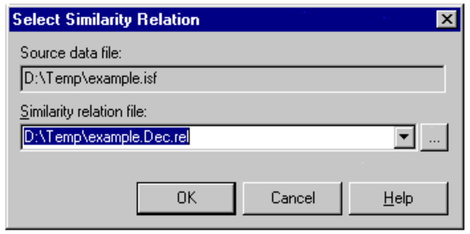
After having selected the relation file, the user is presented the second dialog box, which is used for selecting decision attribute, which will be considered during calculation of approximations. This must be the same attribute as that selected for generating similarity relation.

When approximations are successfully created, they are saved to file. The approximation file is added to the current project and its icon appears in the project window.

Similarity minimal covering

Similarity relation | Similarity minimal covering

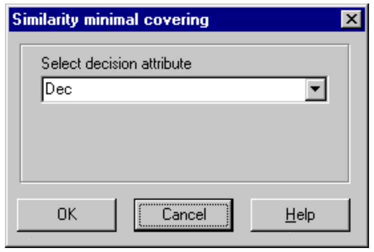
This option is used for inducing decision rules with the use of modified LEM2 algorithm. The modified LEM2 algorithm (so called ModLEM) used similarity relation instead of indiscernibility one and is able to generate rules with extended syntax of conditions (except simple *attribute = value* tests it is able to create tests in form of *lower bound* <= *attribute* <= *upper bound*).



Select Similarity Relation

Source data file: D:\Temp\example.isf

Similarity relation file: D:\Temp\example.Dec.rel



Similarity minimal covering

Select decision attribute: Dec

After having selected this option, the user is presented the first dialog box of those presented above. The user has to select an appropriate relation file. The requested file can be chosen among all relation files contained in current project or the user can browse himself for the necessary file using '...' (browse) button.

After having selected the relation file, the user is presented the second dialog box, which is used for selecting decision attribute, which will be considered during calculation of approximations. This must be the same attribute as that selected for generating similarity relation.

When rules are successfully induced, they are saved to RLF file. The RLF rule file is added to the current project and its icon appears in the project window.

5

References

1. Z.Pawlak: Rough sets. International_Journal of Information and Computer Science 11 (1982) no.5, 341-356.

2. Z.Pawlak, K.Slowinski, R.Slowinski: Rough classification of patients after highly selective vagotomy for duodenal ulcer. International Journal of Man-Machine Studies 24 (1986) 413-433.

3. J.Fibak, Z.Pawlak, K.Slowinski, R.Slowinski: Rough sets based decision algorithm for treatment of duodenal ulcer by HSV. Bulletin of the Polish Academy of Sciences, ser.Biological_Sci. 34 (1986) no.10-12, 227-246.

4. J.Fibak, K.Slowinski, R.Slowinski: The application of rough sets theory to the verification of treatment of duodenal ulcer by HSV. Proc. 6th International Workshop on Expert Systems and their_Applications, Agence de l'Informatique, Paris, 1986, 587-599.

5. K.Slowinski, R.Slowinski, J.Stefanowski: Rough sets approach to analysis of data from peritoneal lavage in acute pancreatitis. Medical_Informatics 13 (1988) no.3, 143-159.

6. K.Slowinski, R.Slowinski: Sensitivity analysis of rough classification. International Journal of Man-Machine Studies 30 (1989).

7. E.Krusinska, R.Slowinski, J.Stefanowski : Discriminant versus rough sets approach to vague data analiysis. Applied Stochastic Models and Data Analysis .

8.R.Slowinski, J.Stefanowski : Rough classification in incomplete information systems. Mathematical and Computer Modelling, 12 10/11 (1989) 1347-1357.

A

ISF File format

ISF (Information System File) is used in the ROSE system to store information system data, e.g. declaration of attributes and definition of objects.

General remarks

ISF is plain text file (it can only contain characters that have ASCII codes in range from 32 to 128 (decimally)).

It has been designed for use in different fields of data discovery and machine learning in mind, so it has section structure, i.e. it is divided into sections. Because there can be many such sections, their order in file is not meaningful, although if there exist two or more sections of the same type, only the first one will be accepted.

There are some general rules concerning the file:

- Each section has its own unique name, by which it is identified, preceded by two stars ('******') and followed by newline character ('**<CR>**'). Section is finished when there appears another section declaration or keyword '**`**END`**'.
- All empty lines are omitted.
- Line ends when there is '**<CR>**' or '**.<CR>**' sequence.
- All following characters are treated as separators: white spaces (one or more space or tabulator), '**,**'.
- When there is any of the following characters: '**|**', '**#**', '**!**' it is recognised as a start of a comment, which is terminated by '**<CR>**'.

- '\' (backslash) is used to split long lines of text. It means that the current line is continued after '**<CR>**' character.
- All of the above characters are reserved and cannot be used as literals in any of the sections.
- The text is caps sensitive.
- At the end of file there should be sequence '**`**END<CR>`**'.

Section description

Currently there are only two sections defined. Attributes section is used to declare data attributes and examples section defines all objects present in database or information system.

Attributes

Section remarks

This section is identified by keyword **ATTRIBUTES** (as mentioned earlier it should be preceded by '******' and followed by '**<CR>**'), all capital letters.

It has several reserved keywords and characters:

- '**:**' (colon) is used when declaring an attribute,
- '**[**', '**]**' (square brackets) used when declaring unordered enlisted attributes,
- '**<**', '**>**' (less and more signs) used when declaring ordered enlisted attributes,
- '**(**', '**)**' (round brackets) used when declaring attribute type,
- '**decision**' is used to declare that attribute is a decision.

All attribute names should be no longer that 30 characters. Remember to avoid white spaces in their names.

Order of attributes declaration is important with connection to Examples section.

Declaring an attribute

Attribute declaration has following syntax:

```
attribute_name:<separator>attribute_type<CR>
```

where *attribute_name* can be any text not containing reserved characters of length less then 30, *attribute_type* is described later.

There are 5 attribute types:

- continuous attributes identified by `(continuous)` statement in attribute declaration, their domain contains all real numbers,
- numbercoded attributes identified by `(numbercoded)` their domain consists of positive integer numbers plus 0,
- "dont care" attributes identified by `(omit)`, such attributes are omitted during the data analysis,
- unordered enlisted attributes identified by sequence `[value1, value2, …, valueN]`, domain of such attributes consists of all listed values, and their order is meaningless (e.g. attribute hair with values: blond, brunette, red, white),
- ordered enlisted attributes identified by sequence `<value1, value2, …, valueN>`, domain of attribute contains all listed values, but their order is meaningful (e.g. attribute temperature with values: low, moderate, high).

Values in enlisted types can be any valid strings no longer than 30 characters. Because ordered enlisted attributes are extensions to the first version of the file format, they may not be accepted by all software. In this case, please change them to unordered enlisted attributes as this will not influence data analysis in these modules.

Denoting attribute as decision

When you want to declare, that some attributes describe decisions (belonging of objects to decision classes), you just enter lines in the attributes section with the following syntax:

```
decision:<separator>attribute_name
```

where *attribute_name* has to be the name of one of the declared attributes, although it may be defined later in the section.

There can be many decision attributes, not only one.

EXAMPLES

Section remarks

This section is identified by keyword **EXAMPLES** (as mentioned earlier it should be preceded by '******' and followed by '**<CR>**'), all capital letters.

It has only one reserved character – '**?**', which denotes missing value of the attribute.

This section contains in each logical line (remember that lines can be split by use of '\' character) description of one object, according to the attributes declared in Attributes section. So objects values must match domains of their respective attributes. Whitespaces or separators separate attribute values. Each lines end with '**<CR>**' or '**.<CR>**'.

In specific situation instead of objects there can be file include declaration in the first line of the section. It has following syntax:

```
*INCLUDE:<separator>file_name
```

It means that all object definitions will be read from the file named *file_name*, which has to contain separate examples section

Exemplary file

```
#Exemplary ISF file

!This is a comment to

**ATTRIBUTES

decision: surgery

number: (omit)

age: (continuous)

temperature: <normal, high> |comment may appear here also

condition: [good,bad]

surgery: [YES,NO]

type: (numbercoded)

alive: [YES,NO]

decision: alive

**EXAMPLES

1, 21, normal, good, NO, 1,YES

2, 74, high, bad, YES, 3, NO.

3, 43, normal, bad, YES, 2, \

NO

4, 9, high, good, NO, 0, YES

**END
```

B

Atoms file format

General remarks

It is plain text file (it can only contain characters that have ASCII codes in range from 32 to 128 (decimally)). It contains three sections:

- header section containing general information about atoms, identified by keyword ‘**`**HEADER`**’,
- atom description section, identified by ‘**`**ATOMS`**’ keyword,
- description of atoms by objects section identified by ‘**`**OBJECTS`**’ keyword.

Objects section is optional.

There are some general rules concerning the file:

- All empty lines are omitted.
- Line ends when with ‘<**CR**>’ (newline character, carriage return),
- All following characters are treated as separators: white spaces (one or more spaces or tabulators), ‘**,**’.
- When there is any of the following characters: ‘|’, ‘#’, ‘**!**’ it is recognised as a start of a comment, which is terminated by ‘<**CR**>’.
- ‘\’ (backslash) is used to split long lines of text. It means that the current line is continued after ‘<**CR**>’.
- All of the above characters are reserved and cannot be used as literals in any of the sections.
- The text is caps sensitive.

- At the end of file there should be sequence ‘**`**END<CR>`**’.

Order of sections in the file is not important.

Header section

First line of this section contains name of the file describing the objects (ISF file), which was basis to atoms search.

Next, there is description of used decision attribute, using following syntax:

```
attribute_name
{
value1<separator>value1_code
…
valueN<separator>valueN_code
}
```

where *attribute_name* is the name of decision attribute, *value* describes all values of this attribute and *value_code* describes internal coding.

Following lines contain all the names of conditional attributes (each attribute in one line).

Next there is a statement that describes if there is Objets section in the file, using syntax:

```
objects:<separator>YES
```

or

```
objects:<separator>NO.
```

Section ends with number of the existing atoms.

Atoms section

Each atom is identified by two lines (remember that lines can be divided using '\' character) of text.

First line contains (using separators):

- ordinal number of atom,
- values of all conditional attributes.

Second line contains:

- cardinality of objects that belong to this atom,
- cardinality of objects belonging to decision classes, using syntax:

```
value_code<separator>number_of_objects
```

where *value_code* is a value of decision attribute (see header section) and *number_of_objects* is number of objects that belong to the atom and at the same time to the decision class. If cardinality of such set is 0, it is omitted.

Objects section (optional)

This section contains information, which objects constitute given atom, and to which decision class they belong. Each line describes one atom, using syntax:

- ordinal number of atom,
- description of objects belonging to the atom and one of the decision classes, with syntax:

```
value_code<separator>{objects}
```

Exemplary file

```
!Examplary atom file
**HEADER
urology.isf
alive
{
 yes 1
 no 2
}
age
sex
placement
objects: YES
3
**ATOMS
1 1 2 1
3 1 2 2 1
2 2 1 3
4 1 3 2 1
3 1 1 1
1 2 1
**OBJECTS
1 1 {1,5} 2 {3}
2 1 {2,4,8} 2 {6}
3 2 {7}
**END
```

C

Rules file format

General remarks

It is plain text file (it can only contain characters that have ASCII codes in range from 32 to 128 (decimally)). There are two versions of rule description file:

- simple (easily readable by human) with rls extension,
- full (more complicated, designed for reading by specified software) with rlf extension.

There are some general rules concerning the file:

- All empty lines are omitted.
- Line ends when with '**<CR>**' (newline character, carriage return),
- All following characters are treated as separators: white spaces (one or more spaces or tabulators), '**,**'.
- When there is any of the following characters: '**|**', '**#**', '**!**' it is recognised as a start of a comment, which is terminated by **'<CR>**'.
- '\' (backslash) is used to split long lines of text. It means that the current line is continued after '**<CR>**'.
- All of the above characters are reserved and cannot be used as literals in any of the sections.
- The text is caps sensitive.

- At the end of file there should be sequence '`**END<CR>`'.

All real numbers are written with precision of two decimal places.

RLS file

This file type contains header and body.

Header

Header contains information about which algorithm was used to generate rules, what were its parameters, date of creation, etc. All its lines are written as a comment. It has following syntax:

- first line contains signature of the rule generator,
- second line contains name of the data file (ISF),
- third line contains number of objects, written as: '`Objects=number_of_objects`',
- fourth line contains number of attributes, written as: '`Attributes=number_of_attributes`',
- fifth line contains the name of the decision attribute, written as: '`Decision=decision_name`',
- sixth line contains all decision classes, written as: '`Classes=class1,class2,…,classN`'.
- next, there is date of creation of the file,
- next line contains name of the algorithm used to generate rules,
- next line contains algorithm parameters,
- last line contains time of computation.

Rules

Each line contains description of exactly one rule. The line begins with sequence 'rule No.', where No is the number of the rule. After separator there is conditional part of the rule, using syntax:

```
    (attribute<separator>relation<separator>value)<separator>&     …
&<separator>(atribute<separator>relation<separator>value)
```

where attribute is the name of conditional attribute, value is its value, and relation is one of the following:

* **'='**,
* '>'**,**
* '<'**,**
* '>='**,**
* '<=',
* "**in**" – for unordinal attributes and intervals**.**

When conditional part of the rule is finished there comes assigment to the decision class(es), with syntax:

```
<separator>=><separator>(decision=value)<separator>OR     …
OR<separator>(decision=value);
```

Each line contains also additional information about the rule, as the following:

```
    [support,strength,relative
strength,level_of_discrimination]
```

Line ends with the '**<CR>**'.

Exemplary rules

```
(A13 = 2) & (wzrost >= œredni) => (d=1); [ 20, 18, 30.53%, 90.00% ]
(A1 < 12) & (wzrost = wysoki) => (d=A) OR (d=B); [ 10, 9, 15.25%, 90.00% ]
```

RLF

RLF file is much similar to the RLS. The difference is that it contains some additional information about each rule. Each rule description takes 2 lines. At the end of the first line, there is information about support in each of the decision classes, using syntax:

```
[support_class1,support_class2,…,support_classN]
```

Second line contains objects that support the rule divided by decision classes, using syntax:

```
    [{class1_objects},{class2_objects},…,{classN_objects}
]
```

where objects are listed by they ordinal numbers using separators.

Exemplary rule

```
    (a13 = 2) & (wzrost >= oredni) => (d=1); [20, 18, 30.53%, 90.00%] [18, 1, 1, 0]
    [{1, 2, 3, 4, 5, 6, 7, 8, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43 }, {10}, {99},
{}]
```

D

Troubleshooting

Here you can find some remarks and hints concerning problems you may encounter when using *ROSE* system.

ROSE system

P: You get error message: ‘Attempt to launch external module failed’.

H: One of the modules is missing or not present in current modules path. Check that the entry in Tools | Options dialog is correct or reinstall system.

P: You get error messages when running external modules.

H1: After running them, open corresponding log files, e.g. when encountering problems looking for reducts see reducts.log or consult the contents of the *console window*.

H2: There may be a problem, when project file and working directory is located on a Novell Netware file server. Please use your local harddrive.