Introduction

RoSDA is a tool kit for performing data analysis tasks in context of Rough Set Theory and Dominance based Rough Set Approach. The tool kit provides both GUI and API functions for easy to use and integrate purpose, developed by Knowledge & Data Science Research Centre (KDRC).

Rough Set based Data ANALYSIS (RoSDA)

VERSION 1.1

https://kdrc.live/rosda/

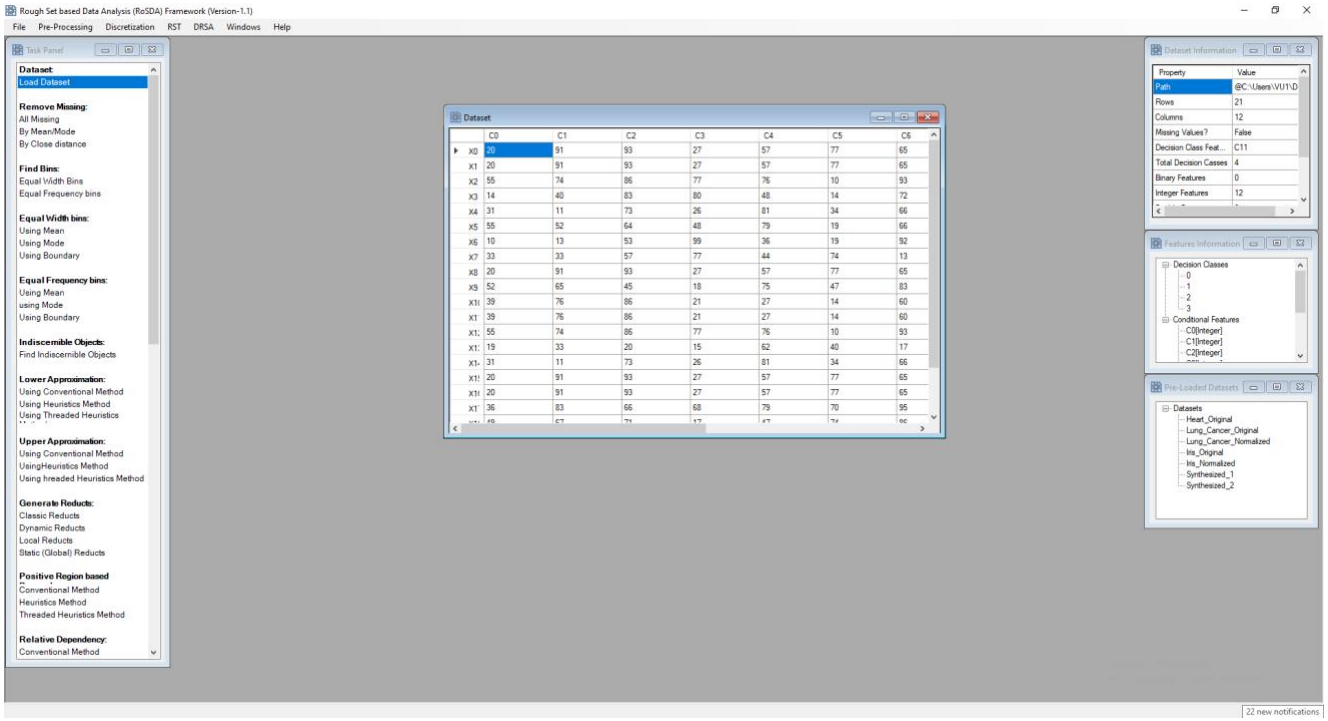
Manual

Manual provides details of the following:

- RoSDA Interface
- Available Operations
- How to operate each form
- APIs
- Disclaimer

RoSDA Interface

RoSDA interface comprises of a main window form with one menu bar and five panels as shown below:



1: Menu bar: Contains menus for performing different tasks

2: Task Panel (left): Contains different tasks (available in menus as well)

3: Dataset Grid (Middle): Comprises of Data Grid that shows dataset.

4: Dataset Information Panel (right top): Contains complete information about dataset.

5: Feature Information Panel (right middle): contains information about features in dataset

6: Preloaded datasets (right bottom): contains pre-loaded datasets. User can double click any dataset to display in dataset grid.

Available Operations

1. Rough Set Theory:

1.1 Task: Data:

The tool will be based on classical concept of data representation in the form:

$\alpha = (U, C \cup D)$

where *"C"* represents conditional attribute(s) or features and *"D"* represents decision attribute(s).

Tool Support:

Tool supports datasets comprising of conditional and decision attributes. There may be any number of conditional and one decision attribute. The last column will always be treated as decision attribute.

Task: 1.1.1 Data Loading:

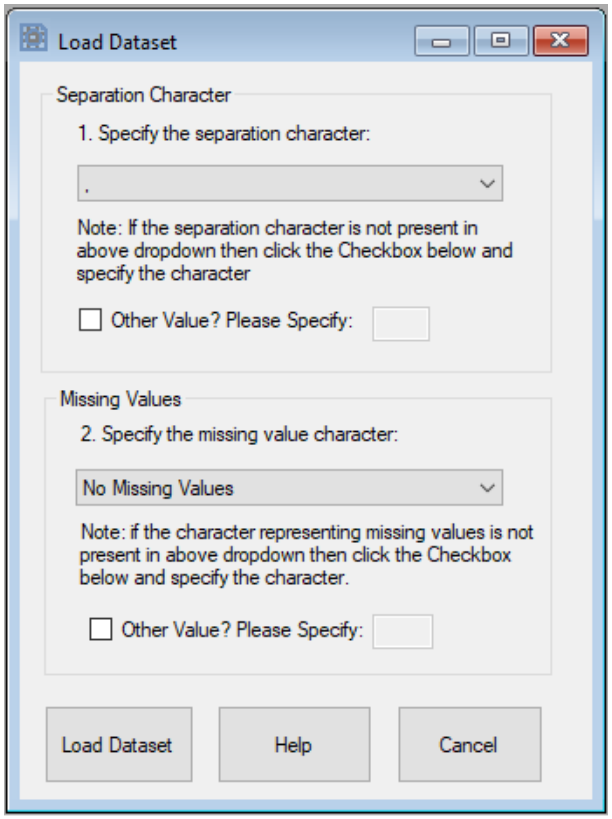
In first version, tool will read data from CSV file. Later versions will support other formats as well.

Tool Support:

Tool supports datasets in any file format. A file can be loaded in two ways:

1) From Menu (File > Load Dataset)
2) From Task Panel

By clicking File>Load Dataset menu or double clicking “Load Dataset” option from Task Panel will display “Load Dataset” form as shown below:

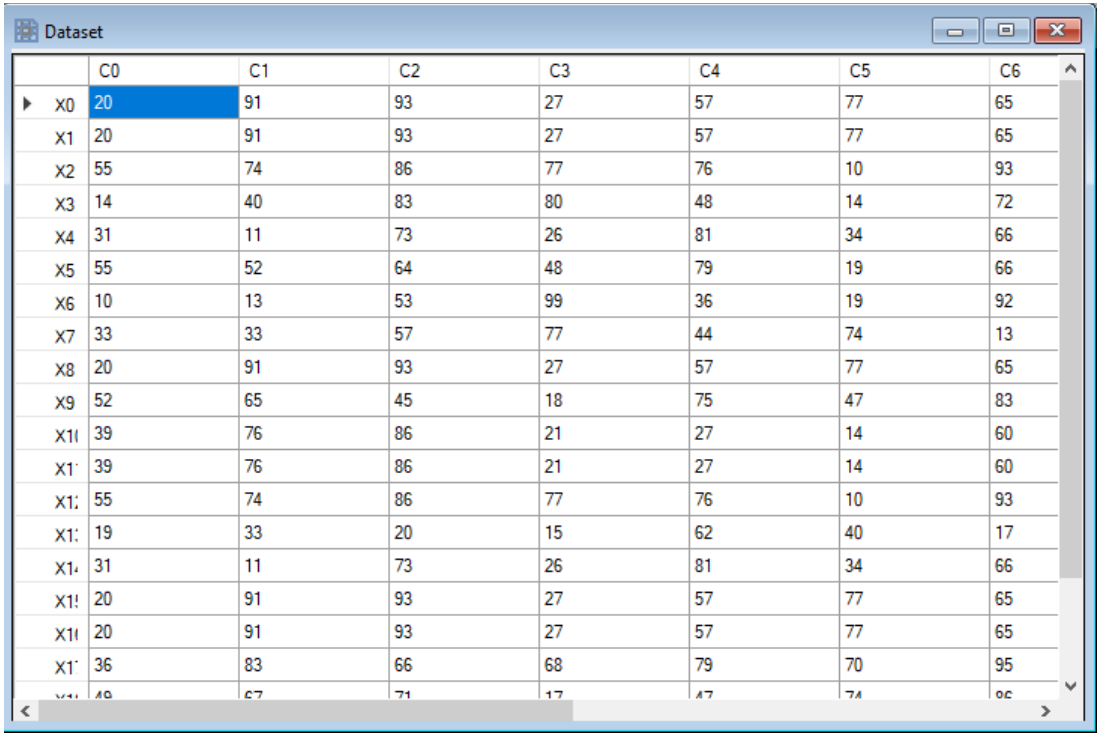


In “Load Dataset” form, you will specify the “Separation character” and “Missing Value” information.

The separation character is the one that is used to separate the data values. For example in CSV, it will be a comma (,). If the separation character is not available in separation character dropdown, user can specify the character by checking the “Other value” checkbox and entering the character in textbox next to it.

Similarly in “Missing Values” box you will specify the character that is used to represent missing values in dataset (if dataset has missing values).

After this user will click the “Load Dataset” button and system will display the “Open File” dialogue. From this dialogue the file containing the dataset will be selected. After clicking the “Open” button on “Open File” dialogue, the system will display the dataset in the DataGrid as shown below:



	C0	C1	C2	C3	C4	C5	C6
X0	20	91	93	27	57	77	65
X1	20	91	93	27	57	77	65
X2	55	74	86	77	76	10	93
X3	14	40	83	80	48	14	72
X4	31	11	73	26	81	34	66
X5	55	52	64	48	79	19	66
X6	10	13	53	99	36	19	92
X7	33	33	57	77	44	74	13
X8	20	91	93	27	57	77	65
X9	52	65	45	18	75	47	83
X1(	39	76	86	21	27	14	60
X1'	39	76	86	21	27	14	60
X1;	55	74	86	77	76	10	93
X1:	19	33	20	15	62	40	17
X1.	31	11	73	26	81	34	66
X1!	20	91	93	27	57	77	65
X1(	20	91	93	27	57	77	65
X1'	36	83	66	68	79	70	95

Note: Objects are represented as X1, X2, X3, …. Xn. The attributes are represented using “C1, C2, C3,…,Cn. The last column i.e. Cn will be used as decision attribute.

Task: 1.1.2 Handling Missing Values:

Dataset in real life may contain missing values. Tool will provide various methods for handling the missing values, for example:

- **Remove Incompletes:** Remove all the objects having incomplete values.
- **MeanMode:** Fill missing values for numerical attributes with the mean value of all observed entries for that attribute. For Strings tool will provide Mode operation.
- **Using Distance Measure:** The object having missing value will be compared with other objects and the missing value will be assigned with the value of corresponding attribute for the object having minimum distance with current object under consideration.

Tool support:

Tool supports all the three mentioned methods to handle the missing values. Functionality is available in:

1: Menus:

Pre-Processing > Remove All Missing

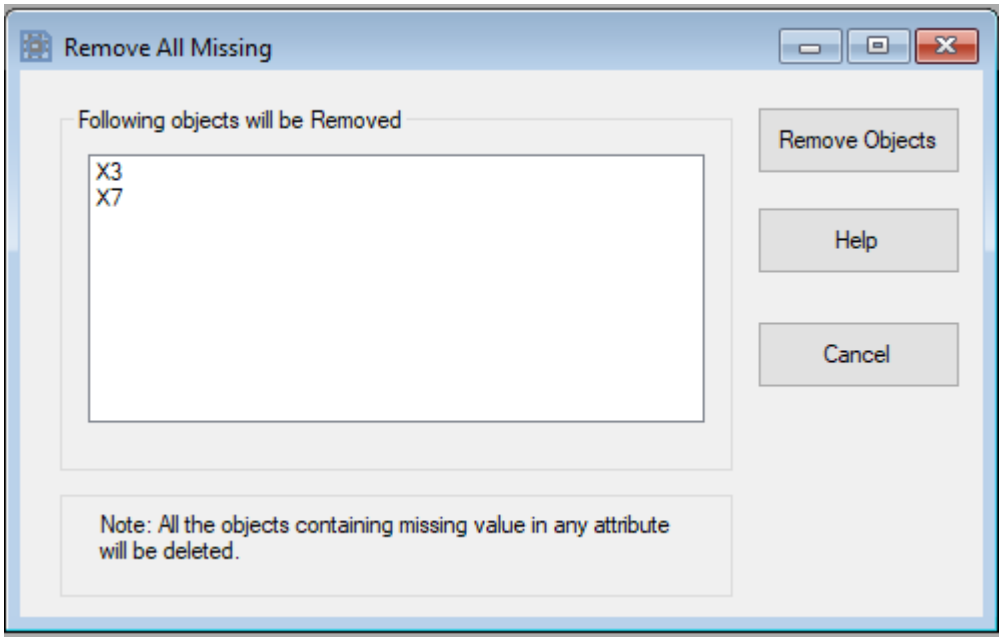
Pre-Processing > Remove Missing by Mean/Mode

Pre-Processing > Remove Using Close Distance

2: Task Panel

Menu: Pre-Processing > Remove All Missing

“Remove All Missing” removes all objects containing missing values. By clicking “Pre-Processing > Remove All Missing”, tool will display “Remove All Missing” form as shown below:

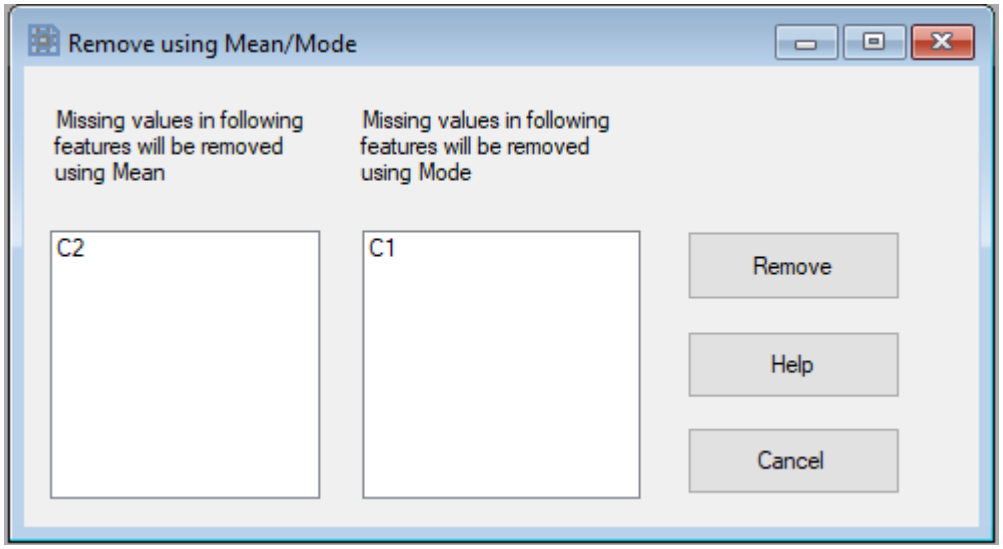


This form will show all objects containing the missing values and by clicking the “Remove Objects” button, all the objects containing the missing values will be removed.

Menu: Pre-Processing > Remove Missing by Mean/Mode

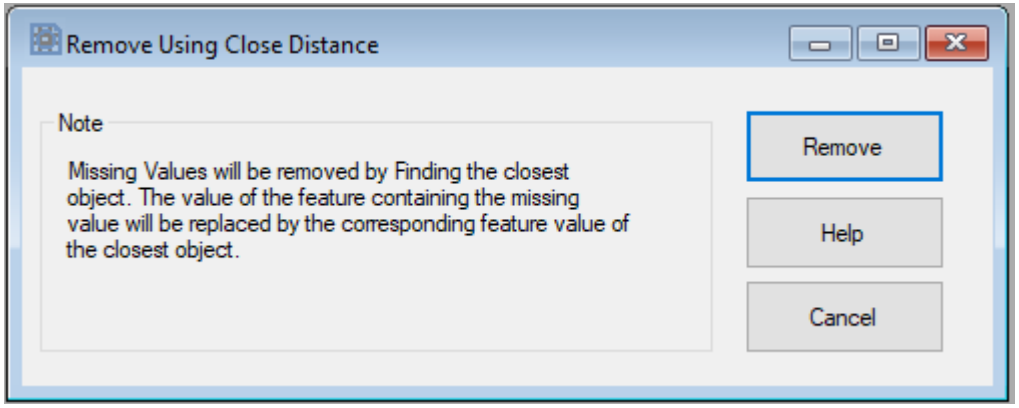
“Remove Missing by Mean/Mode” lets user remove the missing values through Mean and Mode operations. Missing values in integer and float type features are removed using Mean operation (i.e. the missing value is replaced by the mean of all values in that feature). On the other hand, in string type features, the missing values are removed using Mode operation. Listboxes in the “Remove using Mean/Mode” form show the features in which missing values will be removed using Mean operation and the features in which the values will be removed using Mod Operation.

Note: if in a feature, there are two Mode values, then the first value will be used.



Menu: Pre-Processing > Remove Using Close Distance

"Remove Using Close Distance" removes the missing values in an object using the corresponding value in that object. If the corresponding value in closest object is also a missing value, then the corresponding value in next closest object will be used. Figure below shows the "Remove Using Closest" object. By clicking the "Remove" button all the missing values will be replaced.



Task: 1.1.3 pre-loaded datasets:

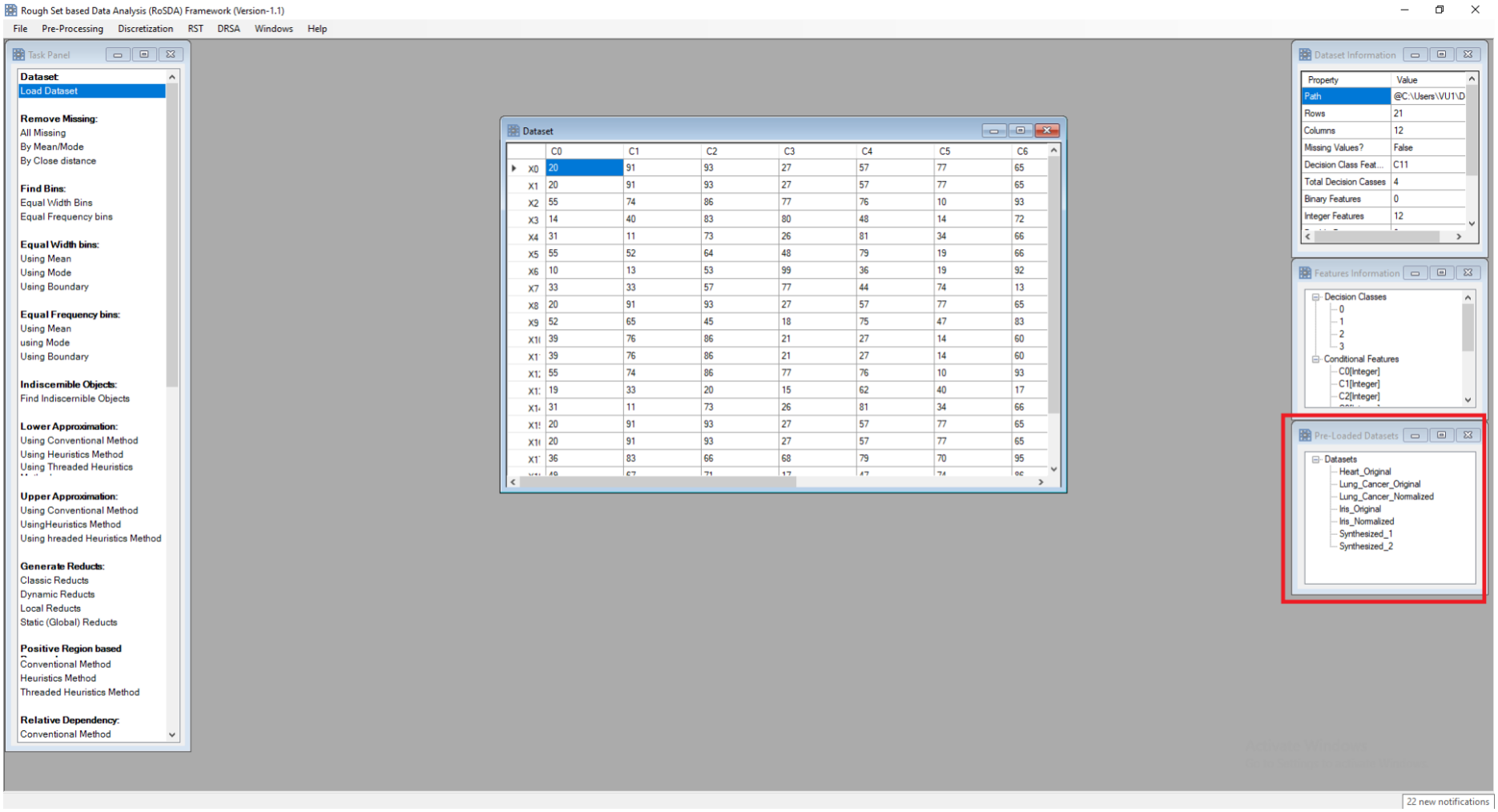
Various datasets e.g. IRIS, Vehicle, Breast Cancer etc. will already be provided (in original and normalized form) along with the tool, so that user don't need to download these datasets from other sources.

Tool Support:

Tool comes with three already provided seven datasets. Following datasets have been provided:

- Heart (Original)
- Lung Cancer (Original)
- Lung Cancer (Normalized)
- Iris (Original)
- Iris (Normalized)
- Synthesized_1 (Custom dataset, does not contain missing values)
- Synthesized_2 (Custom dataset, contains missing values)

Any of these datasets can be loaded by double clicking the name of the dataset from "Pre-loaded Datasets" Panel on main windows form as shown below:


Rough Set based Data Analysis (RoSDA) Framework (Version-1.1)
File
Pre-Processing
Discretization
RST
DRSA
Windows
Help
Task Panel
Dataset:
Load Dataset
Remove Missing:
All Missing
By Mean/Mode
By Close distance
Find Bins:
Equal Width Bins
Equal Frequency bins
Equal Width bins:
Using Mean
Using Mode
Using Boundary
Equal Frequency bins:
Using Mean
using Mode
Using Boundary
Indiscernible Objects:
Find Indiscernible Objects
Lower Approximation:
Using Conventional Method
Using Heuristics Method
Using Threaded Heuristics
Upper Approximation:
Using Conventional Method
UsingHeuristics Method
Using hreaded Heuristics Method
Generate Reducts:
Classic Reducts
Dynamic Reducts
Local Reducts
Static (Global) Reducts
Positive Region based
Conventional Method
Heuristics Method
Threaded Heuristics Method
Relative Dependency:
Conventional Method
Dataset
C0 C1 C2 C3 C4 C5 C6
X0 20 91 93 27 57 77 65
X1 20 91 93 27 57 77 65
X2 55 74 86 77 76 10 93
X3 14 40 83 80 48 14 72
X4 31 11 73 26 81 34 66
X5 55 52 64 48 79 19 66
X6 10 13 53 99 36 19 92
X7 33 33 57 77 44 74 13
X8 20 91 93 27 57 77 65
X9 52 65 45 18 75 47 83
X10 39 76 86 21 27 14 60
X11 39 76 86 21 27 14 60
X12 55 74 86 77 76 10 93
X13 19 33 20 15 62 40 17
X14 31 11 73 26 81 34 66
X15 20 91 93 27 57 77 65
X16 20 91 93 27 57 77 65
X17 36 83 66 68 79 70 95
Dataset Information
Property Value
Path @C:\Users\VU1\D
Rows 21
Columns 12
Missing Values? False
Decision Class Feat... C11
Total Decision Casses 4
Binary Features 0
Integer Features 12
Features Information
Decision Classes
0
1
2
3
Conditional Features
C0[Integer]
C1[Integer]
C2[Integer]
Pre-Loaded Datasets
Datasets
Heart_Original
Lung_Cancer_Original
Lung_Cancer_Normalized
Iris_Original
Iris_Normalized
Synthesized_1
Synthesized_2
22 new notifications

Task: 1.2. Discretization

Some algorithms require data in form of nominal attributes. The first version of the tool will provide the following methods for data discretization:

- **Equal Width Binning:** Separating all possible values into 'N' number of bins, each having the same width.
- **Equal Depth (Frequency) Binning:** Separating all possible values into 'N' number of bins, each having the same number of observations.

The following approaches will be used for smoothing purpose:

- **Smoothing by bin means:** In smoothing by bin means, each value in a bin is replaced by the mean value of the bin.
- **Smoothing by bin median:** In this method each bin value is replaced by its bin median value.
- **Smoothing by bin boundary:** In smoothing by bin boundaries, the minimum and maximum values in a given bin are identified as the bin boundaries. Each bin value is then replaced by the closest boundary value.

Tool Support:

Tool supports both discretization methods i.e. discretization by using Equal Width Binning and discretization by using Equal Frequency Binning. Once the bins are formed, the values can be smoothed either by using mean, median or boundary operations.

Task can be performed from:

1: Menus:

Discretization > Find Bins > Find Equal Width Bins

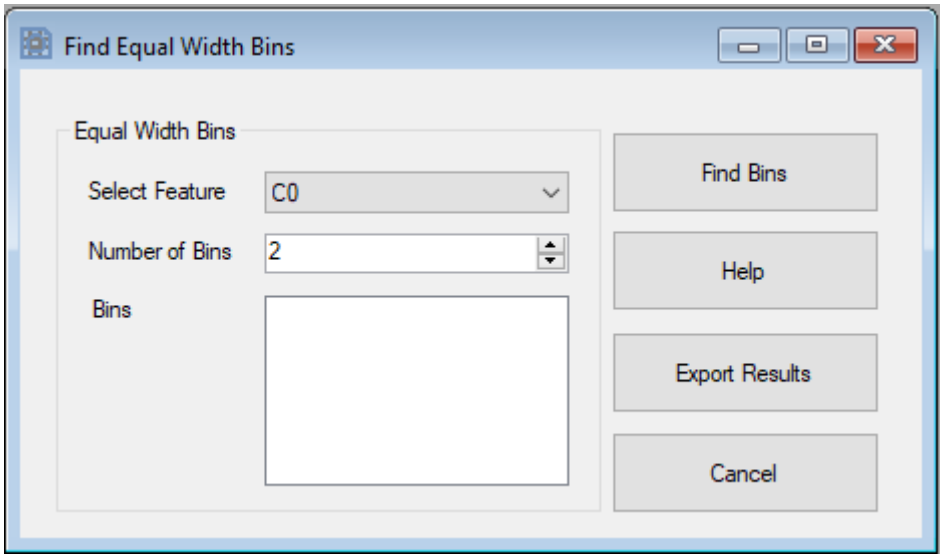
Discretization > Find Bins > Find Equal Frequency Bins

2: Task Panel

Menus:

Menu: Discretization > Find Bins > Equal Width Bins

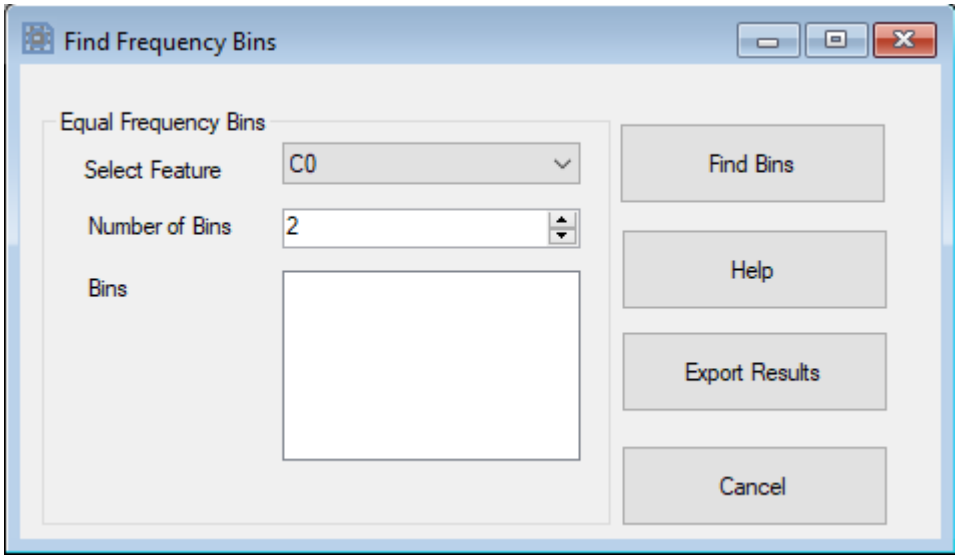
“Find Bins > Equal Width Bins” submenu from the “Discretization” menu lets user find the bins that will be formed using Equal Width Binning method. The form below can be used to show the bins using Equal Width Binning Method:



User can specify any feature in which Bins will be formed. In “Equal Width Bins” form, bins will be formed using “Equal Width Binning”. Any number of bins can be formed, however, so far, minimum two and maximum four bins are formed.

Menu: Discretization > Find Bins > Equal Frequency Bins

“Find Bins > Equal Frequency Bins” submenu from the “Discretization” menu lets user find the bins that will be formed using Equal Frequency Binning method. The form below can be used to show the bins using Equal Frequency Binning Method:

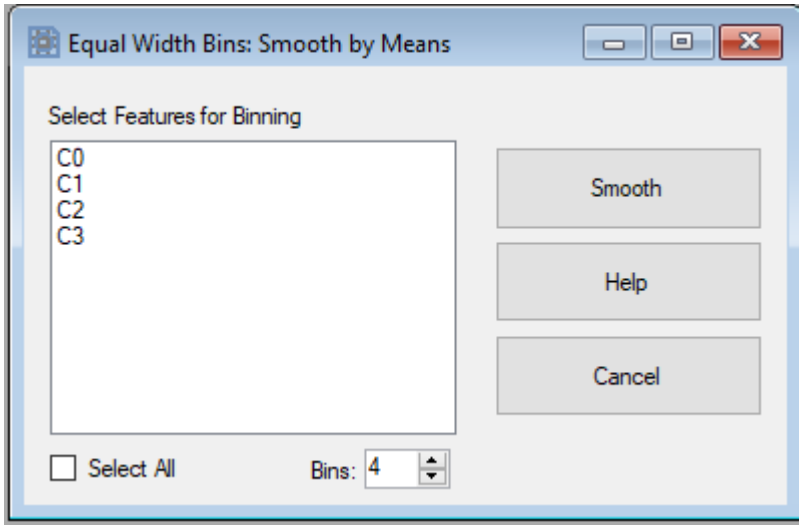


User can specify any feature in which Bins will be formed. In "Equal Frequency Bins" form, bins will be formed using "Equal Frequency Binning" method. Any number of bins can be formed, however, so far, minimum two and maximum four bins are formed.

Menu: Discretization > Equal Width Bins > Smooth by Means

The submenu "Discretization > Equal Width Bins > Smooth by Means" lets user smooth the feature values by using Mean values of the Equal Width Bins.

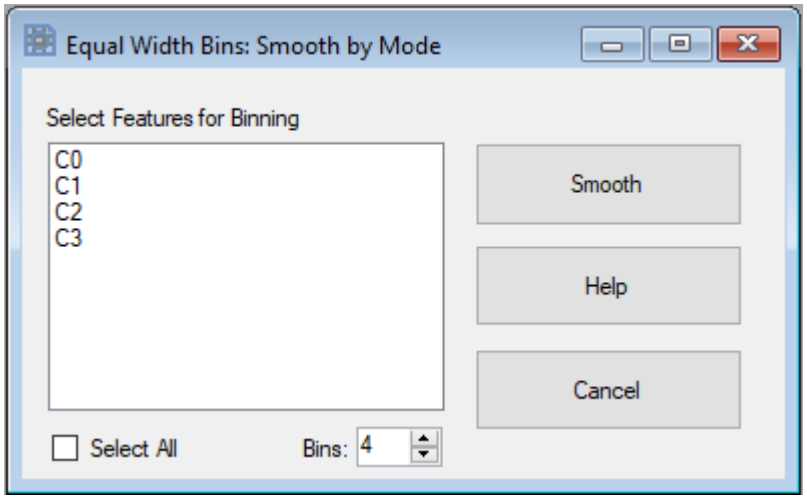
Figure below shows the "Smooth by Means" form:



Menu: Discretization > Equal Width Bins > Smooth by Mode

The submenu “Discretization > Equal Width Bins > Smooth by Mode” lets user smooth the feature values by using Mode values of the Equal Width Bins.

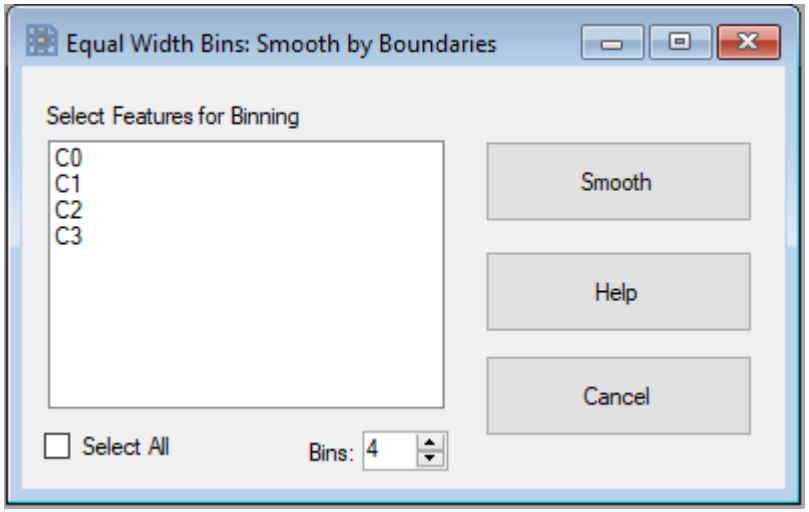
Figure below shows the “Smooth by Mode” form:



Menu: Discretization > Equal Width Bins > Smooth by Boundaries

The submenu “Discretization > Equal Width Bins > Smooth by Boundaries” lets user smooth the feature values by using Boundary values of the Equal Width Bins.

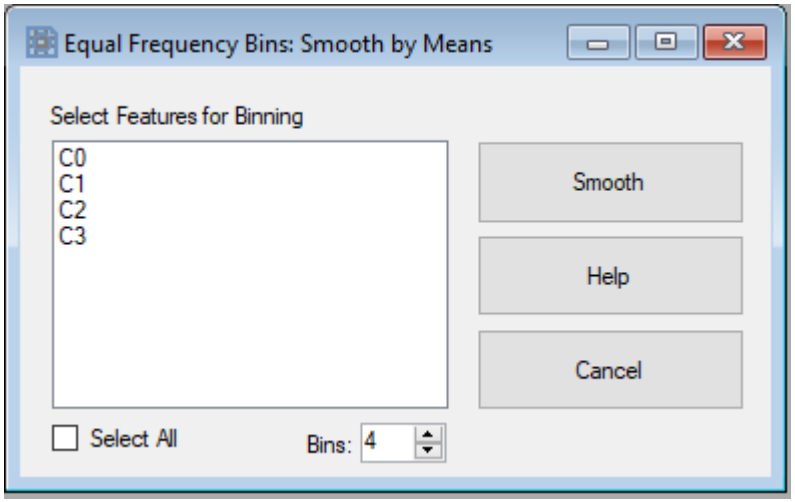
Figure below shows the “Smooth by Boundaries” form:



Menu: Discretization > Equal Frequency Bins > Smooth by Mean

The submenu “Discretization > Equal Frequency Bins > Smooth by Boundaries” lets user smooth the feature values by using mean values of the Equal Frequency Bins.

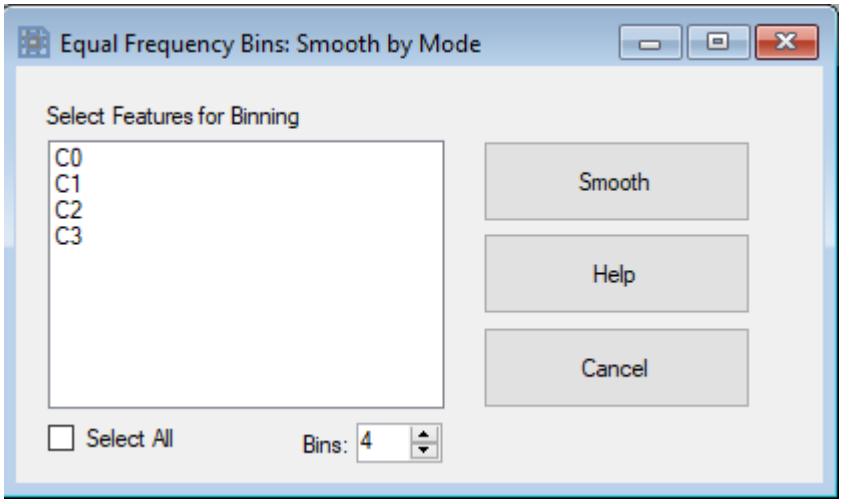
Figure below shows the “Smooth by Mean” form:



Menu: Discretization > Equal Frequency Bins > Smooth by Mode

The submenu “Discretization > Equal Frequency Bins > Smooth by Mode” lets user smooth the feature values by using mode values of the Equal Frequency Bins.

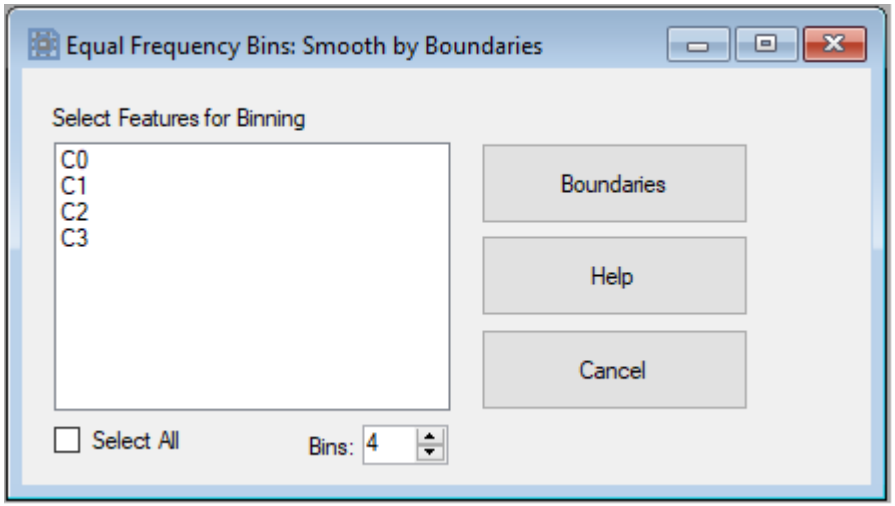
Figure below shows the “Smooth by Mode” form:



Menu: Discretization > Equal Frequency Bins > Smooth by Boundaries

The submenu “Discretization > Equal Frequency Bins > Smooth by Boundaries” lets user smooth the feature values by using boundaries values of the Equal Frequency Bins.

Figure below shows the “Smooth by Boundaries” form:



Note: For each binning method to work, the dataset should not have any missing value.

1.3 Rough Set Preliminaries:

Task: 1.3.1 Indiscernibility:

Indiscernibility identifies the objects that with regard to certain attributes cannot be distinguished from each. The tool will let users find the indiscernible objects with respect to any attribute set.

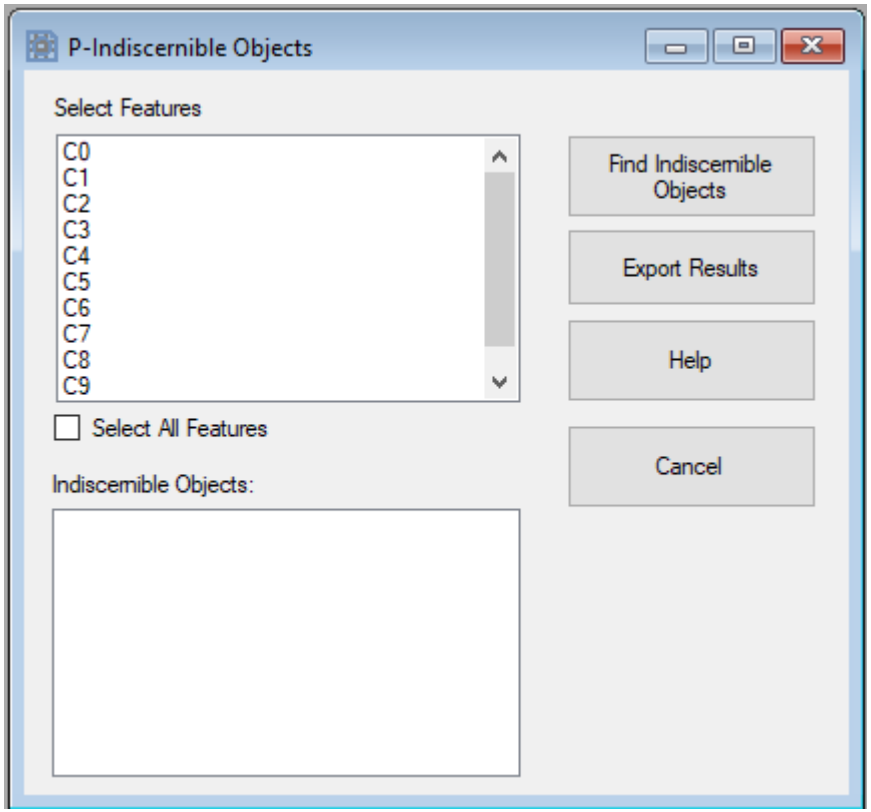
Tool Support:

Tool lets select P-Indiscernible objects for number of selected features. The task can be performed from:

1. **Menu:** RST > Find Indiscernible Objects
2. **Task Panel**

Menu:

By clicking the "RST > Find Indiscernible Objects", tool will display "P-Indiscernible Objects" form as shown below:



When the form is loaded, the tool displays all the features in dataset. By selecting any number of features and then clicking “Find Indiscernible Objects” button, the tool will display all the indiscernible objects from each other (in a single row) with respect to the selected features.

The same task can be performed from the task panel as well.

Note: to find P-Indiscernible objects, dataset should not have any missing value.

Task: 1.3.2 Approximations (Lower and Upper):

Tool will provide calculation of approximations for any concept present in the dataset. Both lower and upper approximations will be calculated, using conventional and advanced methods.

Tool Support:

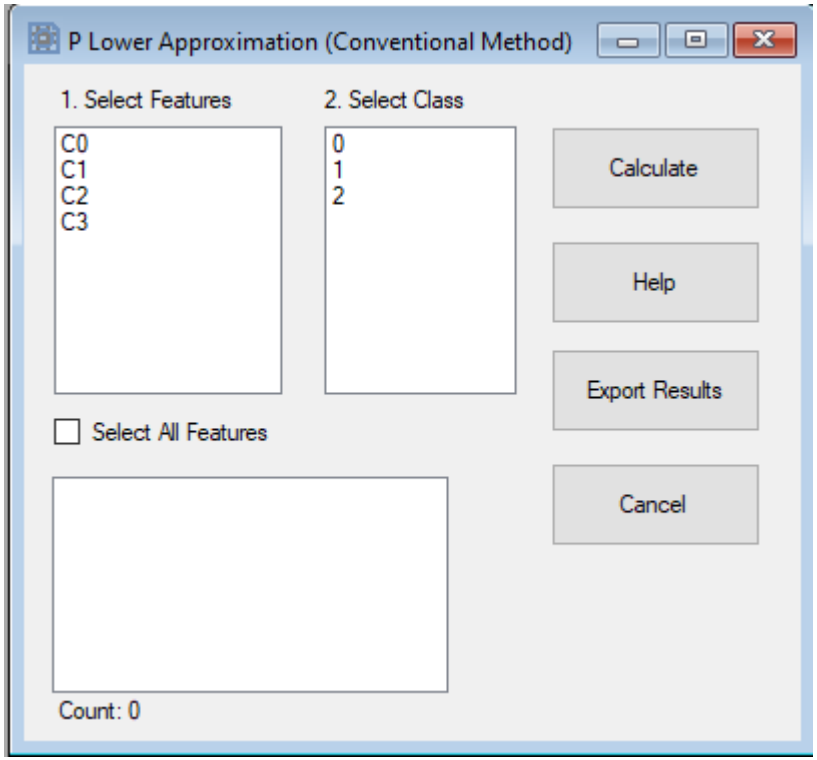
Tool supports calculation of both P-Lower and P-Upper approximations using any concept.

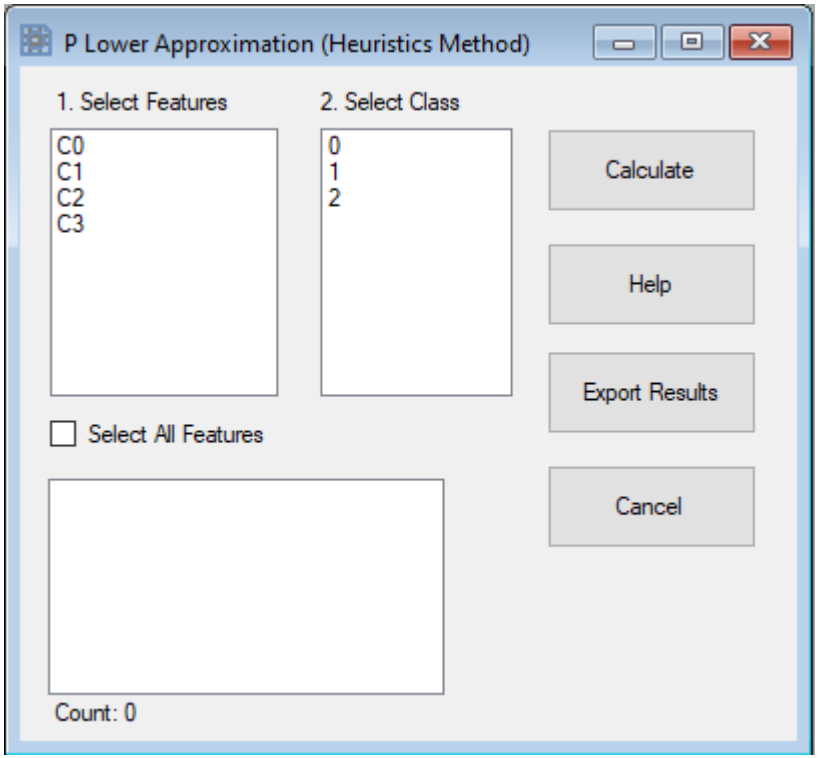
P-Lower approximation can be calculated using conventional method, heuristics method and threaded heuristics based method from the following menus.

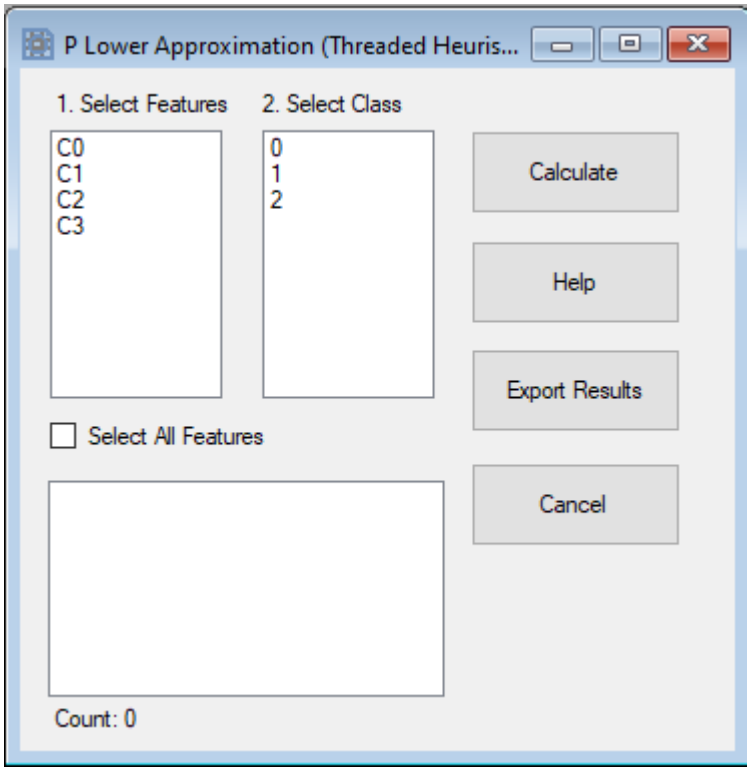
1. **Menu:**
    a. **Menu: RST > Find P-Lower Approximation > Conventional Method**
    b. **Menu: RST > Find P-Lower Approximation > Heuristics Method**
    c. **Menu: RST > Find P-Lower Approximation > Threaded Heuristics Method**

2. **Task Panel**

Figures below show all three forms to calculate P-Lower approximation using the above mentioned methods:


P Lower Approximation (Conventional Method)
1. Select Features
2. Select Class
C0
C1
C2
C3
0
1
2
Calculate
Help
Export Results
Cancel
Select All Features
Count: 0


P Lower Approximation (Heuristics Method)
1. Select Features
2. Select Class
C0
C1
C2
C3
0
1
2
Calculate
Help
Export Results
Cancel
Select All Features
Count: 0



By selecting the feature(s) and the decision class and clicking the “Calculate” button, the objects belonging to lower approximation will be displayed.

Following is detail of each method:

Conventional Method Button: Calculates approximations using conventional Rough Set Theory method.

Heuristics Method Button: Calculates the heuristic approach to calculate P-Lower approximations.

Threaded Heuristics Method Button: Calculates the parallel version of heuristic approach to calculate P-Lower approximations. Threads have been used to implement the parallel version.

The same task can be performed using Task Panel.

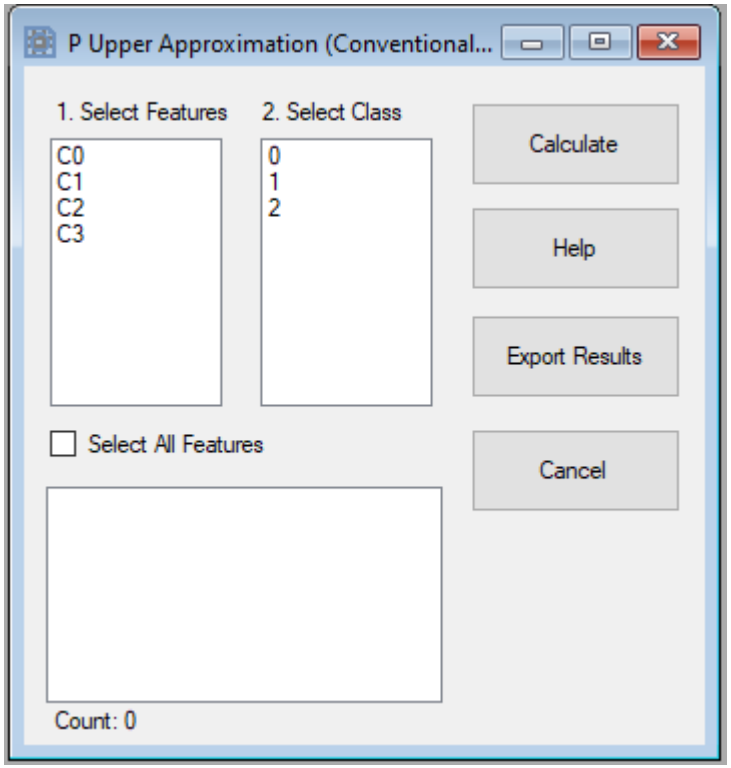
Note: To calculate approximations, dataset should not contain any missing value.

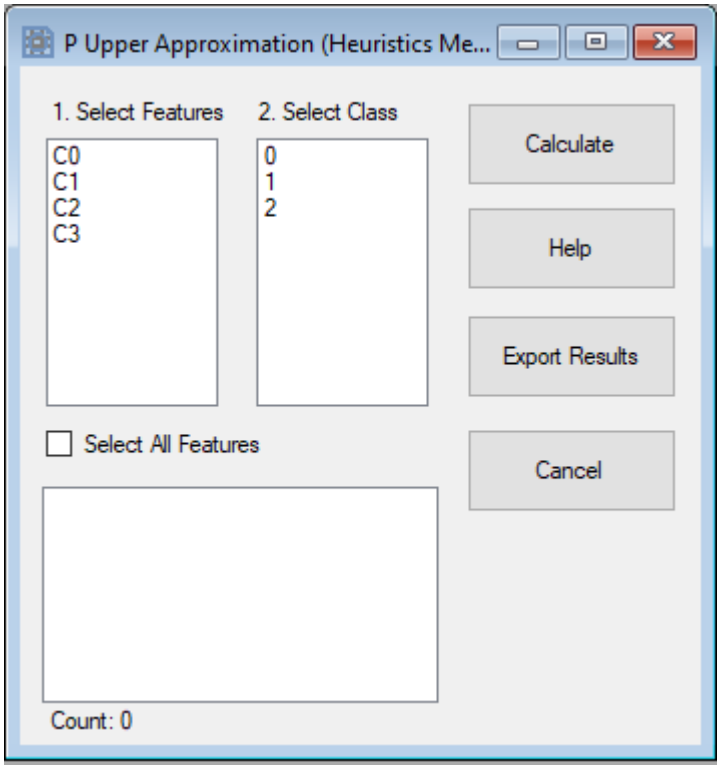
Similarly, P-Upper approximation can be calculated using conventional method, heuristics method and threaded heuristics based method from the following menus.

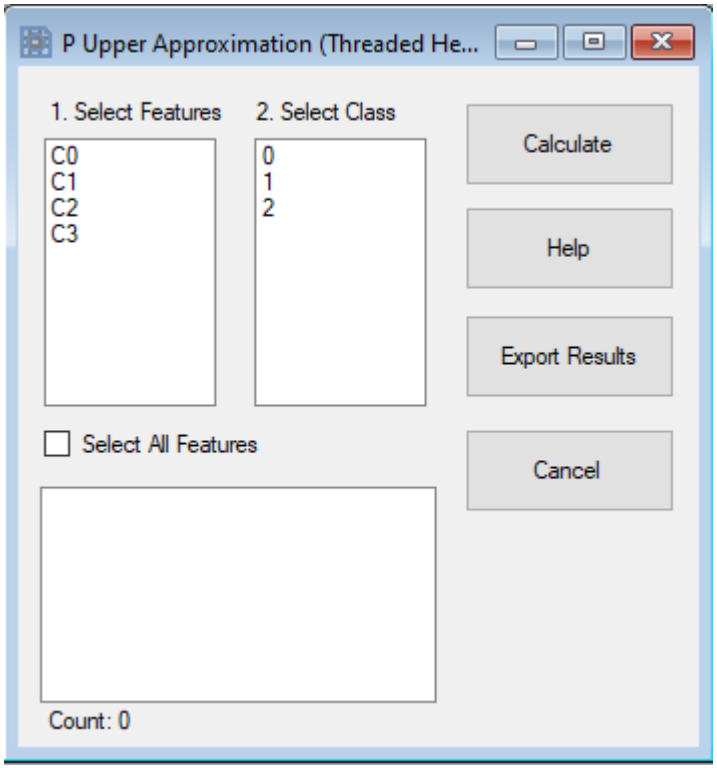
1. **Menu:**
    a. **Menu: RST > Find P-Upper Approximation > Conventional Method**
    b. **Menu: RST > Find P-Upper Approximation > Heuristics Method**
    c. **Menu: RST > Find P-Upper Approximation > Threaded Heuristics Method**

2. **Task Panel**

Figures below show all three forms to calculate P-Upper approximation using the above mentioned methods:


P Upper Approximation (Conventional...
1. Select Features
2. Select Class
C0
C1
C2
C3
0
1
2
Calculate
Help
Export Results
Cancel
Select All Features
Count: 0


P Upper Approximation (Heuristics Me...
1. Select Features
2. Select Class
C0
C1
C2
C3
0
1
2
Calculate
Help
Export Results
Cancel
Select All Features
Count: 0



By selecting the feature(s) and the decision class and clicking “Calculate” button will display the P-Upper approximation of the selected concept (class) using the select feature(s).

Following is detail of each method:

Conventional Method Button: Calculates approximations using conventional Rough Set Theory method.

Heuristics Method Button: Calculates the heuristic approach to calculate P-Lower approximations.

Threaded Heuristics Method Button: Calculates the parallel version of heuristic approach to calculate P-Lower approximations. Threads have been used to implement the parallel version.

The same task can be performed using Task Panel.

Note: To calculate approximations, dataset should not contain any missing value.

Task: 1.3.3 Dependency:

Dependency is the most commonly used measure to perform data analysis. Tool will let the users calculate dependency of decision class on any set of attributes using both the conventional and advance methods.

Both positive region-based dependency and relative dependency will be calculated.

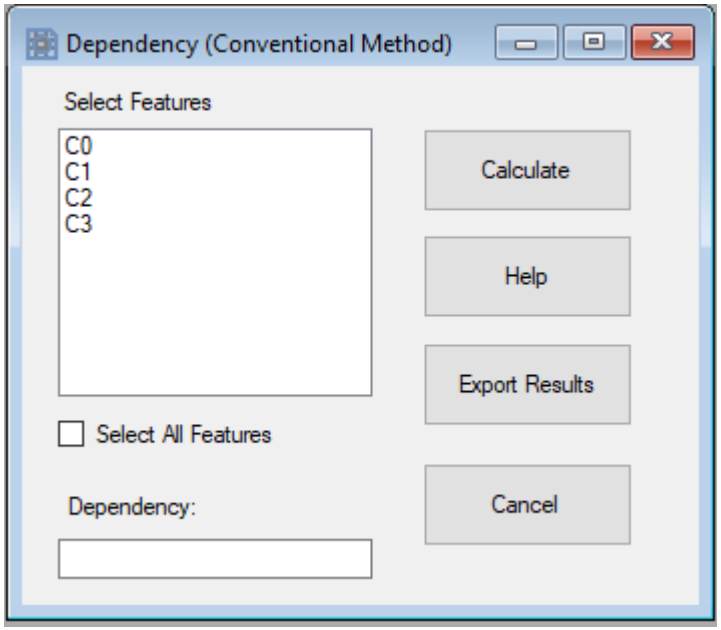
Tool Support:

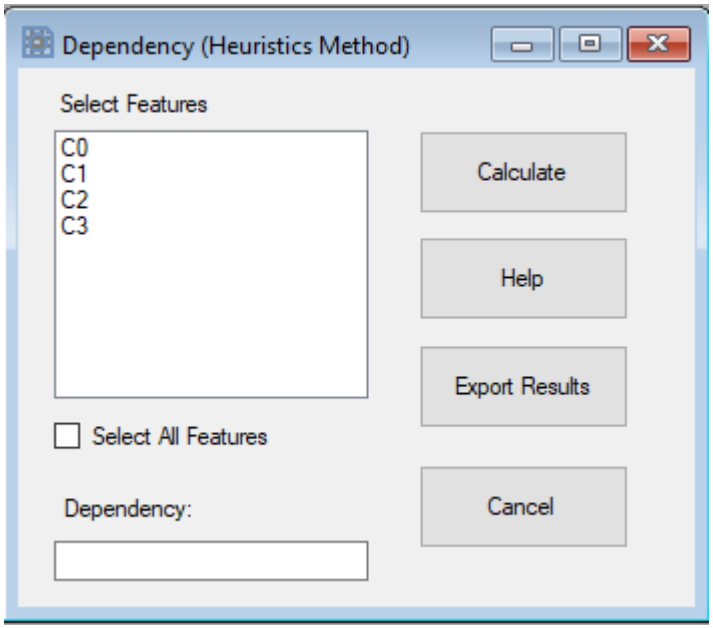
Tool supports both Positive Region based dependency and Relative Dependency measures.

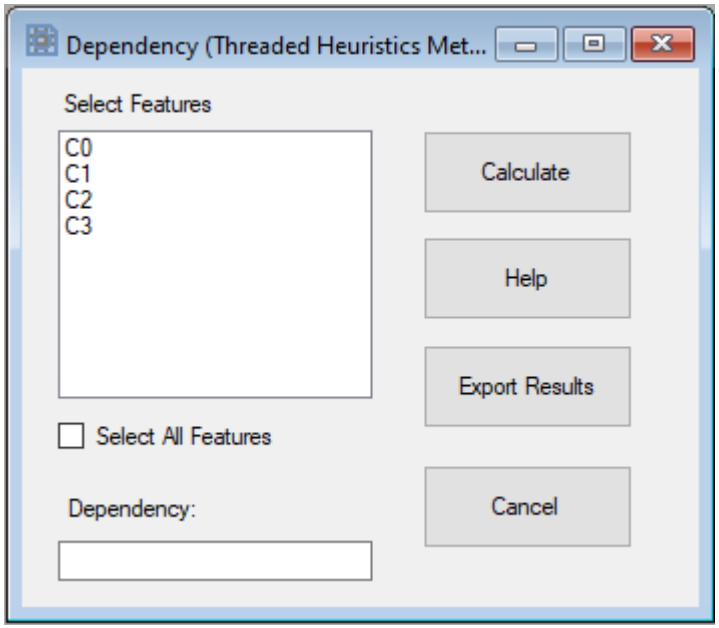
Positive Region based Dependency can be calculated by using conventional method, heuristics based method and threaded heuristics based method by using the following menus:

1. Menus
    a. **Menu: RST > Positive Region based Dependency > Conventional Method**
    b. **Menu: RST > Positive Region based Dependency > Heuristics based Approach**
    c. **Menu: RST > Positive Region based Dependency > Threaded Heuristics Approach**
2. Task Panel

Figures below show the forms used to calculate each type of dependency:


Dependency (Conventional Method)
Select Features
C0
C1
C2
C3
Calculate
Help
Export Results
Select All Features
Dependency:
Cancel


Dependency (Heuristics Method)
Select Features
C0
C1
C2
C3
Calculate
Help
Export Results
Select All Features
Dependency:
Cancel



By selecting the feature(s) and then clicking “Calculate” button will display the “Positive Region Based” dependency in the textbox.

Following is details of each of the above mentioned methods.

Conventional Method Button: Calculates Positive Region based Dependency using conventional Rough Set Theory method.

Heuristics Method Button: Calculates the heuristic approach to calculate Positive Region based Dependency.

Threaded Heuristics Method Button: Calculates the parallel version of heuristic approach to calculate Positive Region based Dependency. Threads have been used to implement the parallel version.

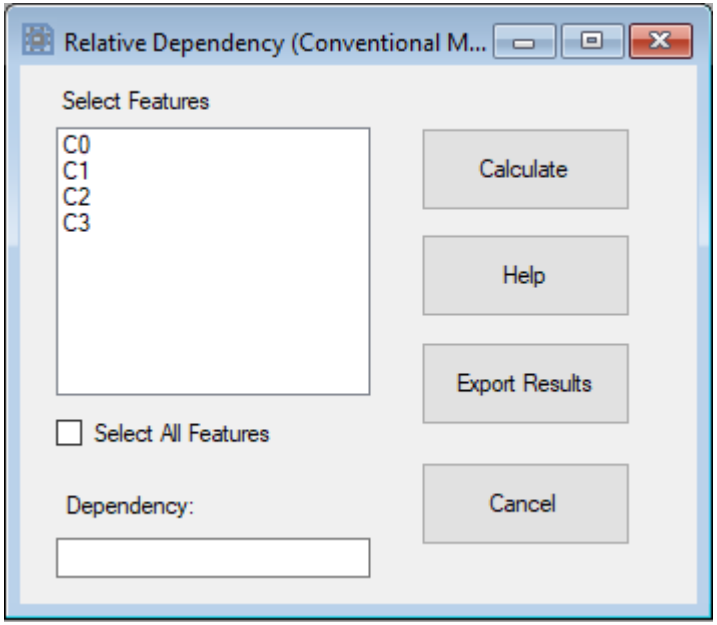
The same task can be performed using Task Panel.

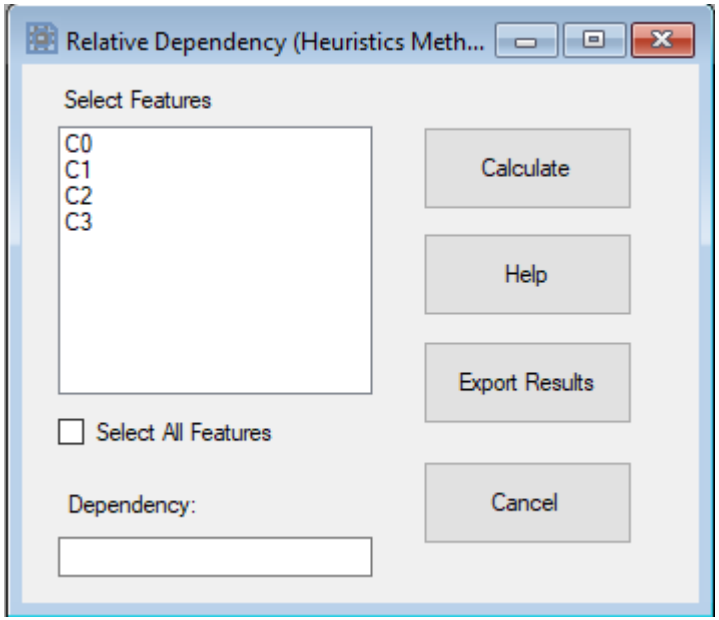
Note: To calculate Positive Region based Dependency, dataset should not contain any missing value.

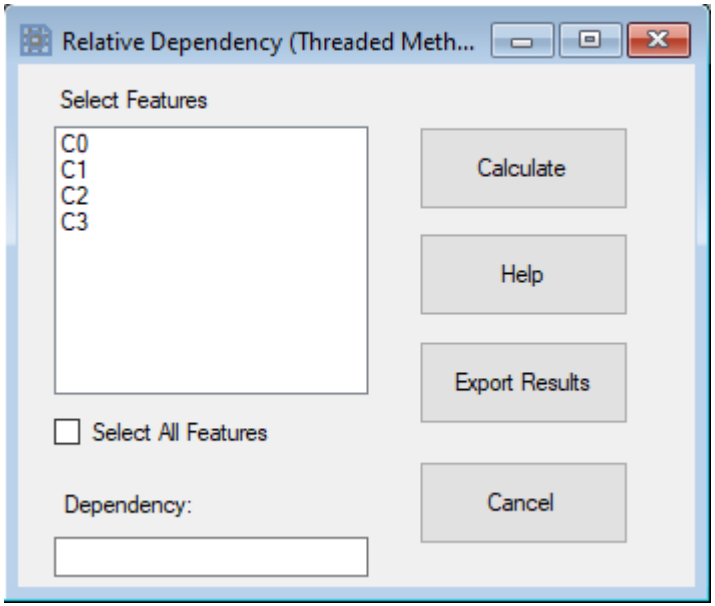
Similarly, Relative Dependency can be calculated by using conventional method, heuristics based method and threaded method by using the following menus:

1. **Menus**
    a. **Menu: RST > Relative Dependency > Conventional Method**
    b. **Menu: RST > Relative Dependency > Heuristics based Method**
    c. **Menu: RST > Positive Region based Dependency > Threaded Heuristics Method**
2. **Task Panel**

Figures below show the forms used to calculate relative dependency using each method:




Relative Dependency (Heuristics Meth...
Select Features
C0
C1
C2
C3
Calculate
Help
Export Results
Select All Features
Dependency:
Cancel



By selecting the feature(s) and then clicking “Calculate” button will display the “Relative” dependency in the textbox.

Conventional Method Button: Calculates Relative Dependency using conventional approach.

Heuristics Method Button: Calculates the heuristic approach to calculate Relative Dependency.

Threaded Heuristics Method Button: Calculates the parallel version of heuristic approach to calculate Relative Dependency. Threads have been used to implement the parallel version.

The same task can be performed using Task Panel.

Note: To calculate Relative Dependency, dataset should not contain any missing value.

Task: 1.3.4 Reducts:

Reduct is a key concept in Rough Set Theory. It can be used to remove some data without loss of information. Tool will let the users calculate both the static/dynamic and Local/Global Reducts using various state of the art algorithms. Furthermore, user will be able to generate Reducts using both the dependency measure and the discernibility matrix.

Tool Support:

Tool supports local (w.r.t. an object), Global (static), Classic Reducts and Dynamic Reduts. Classic reducts have been generated using discernibility matrix whereas static, dynamic reducts are calculated using positive region based dependency.

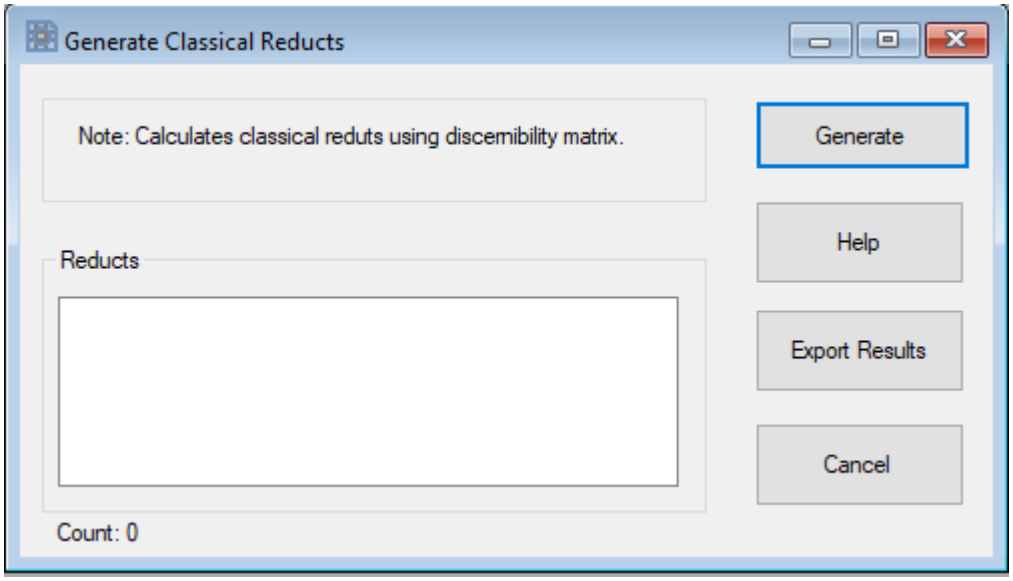
The following menus can be used to assess the forms for calculating Reducts:

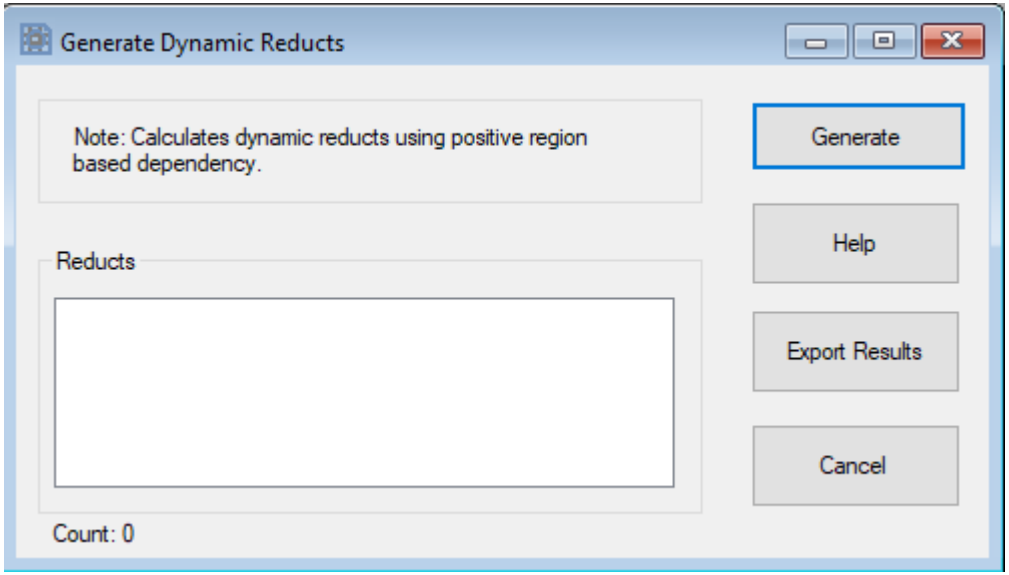
1. **Menus**
    a. **Menu: RST > Generate Reducts > Classic Reducts**
    b. **Menu: RST > Generate Reducts > Dynamic Reducts**
    c. **Menu: RST > Generate Reducts > Local Reducts**
    d. **Menu: RST > Generate Reducts > Static (Global) Reducts**

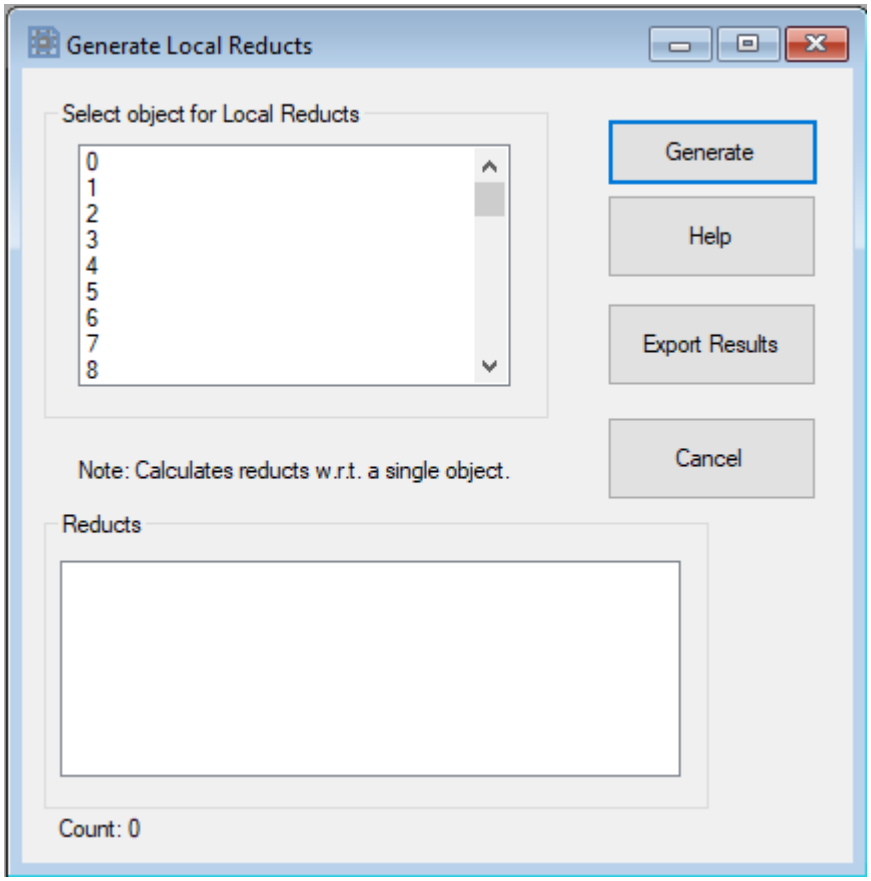
2. **Task Panel**

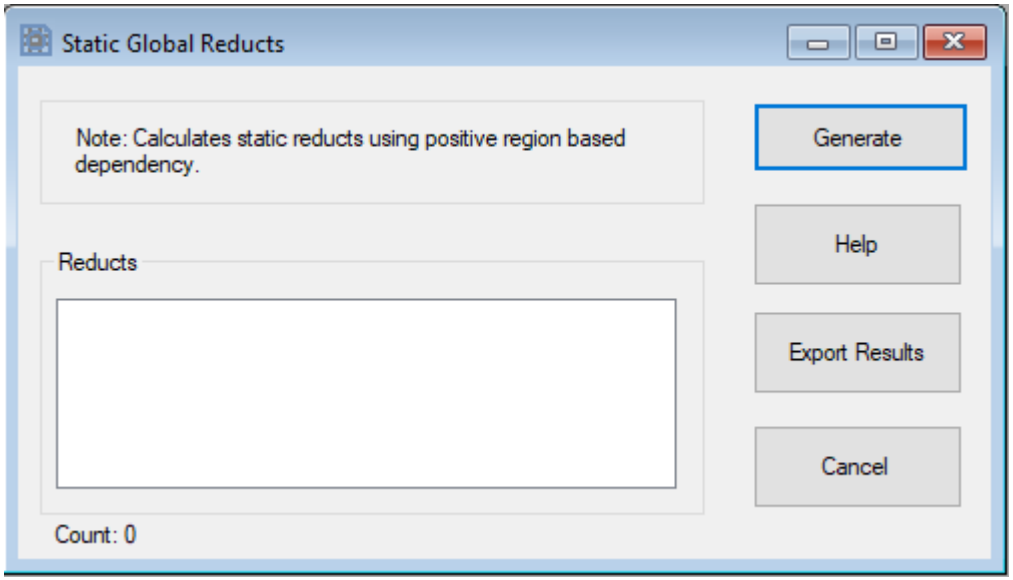
Figures below show each of the forms used to calculate the above mentioned types of reducts.

Reducts can be calculated by using:


Generate Classical Reducts
Note: Calculates classical reduts using discernibility matrix.
Generate
Help
Reducts
Export Results
Cancel
Count: 0


Generate Dynamic Reducts
Note: Calculates dynamic reducts using positive region based dependency.
Generate
Help
Reducts
Export Results
Cancel
Count: 0


Generate Local Reducts
Select object for Local Reducts
0
1
2
3
4
5
6
7
8
Generate
Help
Export Results
Cancel
Note: Calculates reducts w.r.t. a single object.
Reducts
Count: 0



By clicking the “Generate” button the reducts will be generated in the Reducts Listbox.

Task: 1.3.5 Inconsistency Identification:

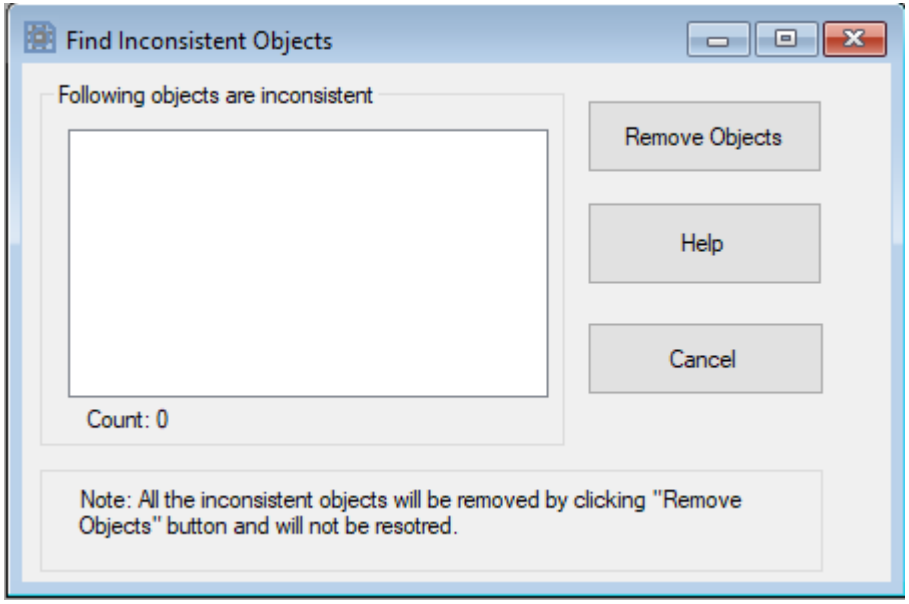
Sometimes it is necessary to identify the inconsistent objects i.e., the objects with same conditional attributes but having different decision class. Tool will provide functionality to identify all such objects. Furthermore, inconsistent datasets will be transformed to consistent ones by removing the inconsistent objects. The consistent dataset can then be used to perform further tasks e.g. calculation of relative dependency.

Tool Support:

Tool provides support to identify and remove inconsistent objects. The task can be perform from:

1. Menu: RST > Find Inconsistent Objects
2. Task Panel

By clicking the submenu “RST > Find Inconsistent Objects, tool will display “Find Inconsistent Objects” form, as shown below:



Form will display the inconsistent objects in the datasets. By clicking the “Remove Objects” button, all the inconsistent objects will be removed and Datagrid will updated. If there are no inconsistent objects, then inconsistent objects listbox will be empty.

The same task can be performed from Task Panel

Note: To find inconsistent objects, dataset should not contain any missing value.

Task: 1.4 Feature Selection:

The first version of the tool will provide various state of the art approaches for feature selection using Rough Set Theory. It will include both the semi exhaustive and heuristic based approaches.

In semi exhaustive methods, both forward feature generation and backward feature generation approaches will be provided.

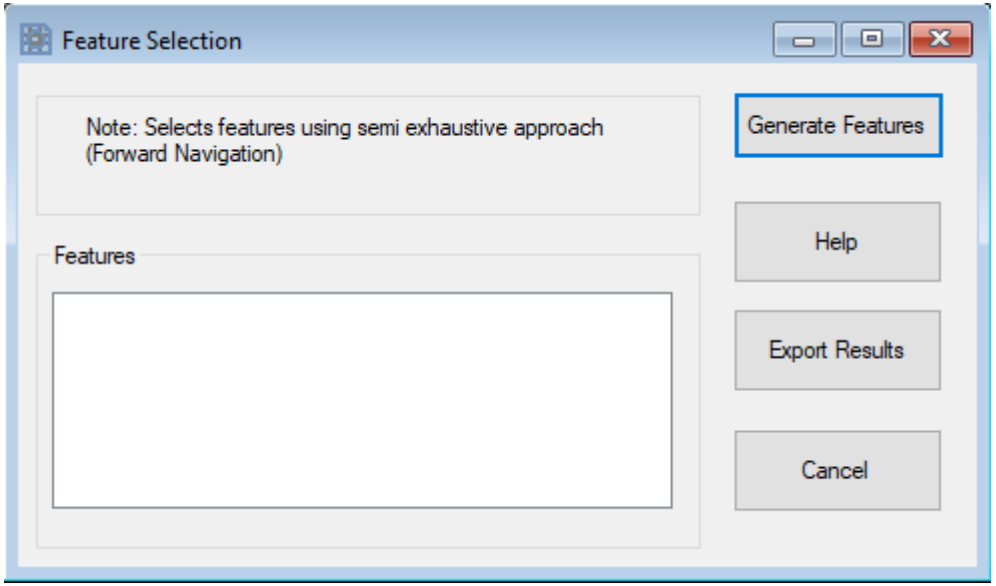
Tool Support:

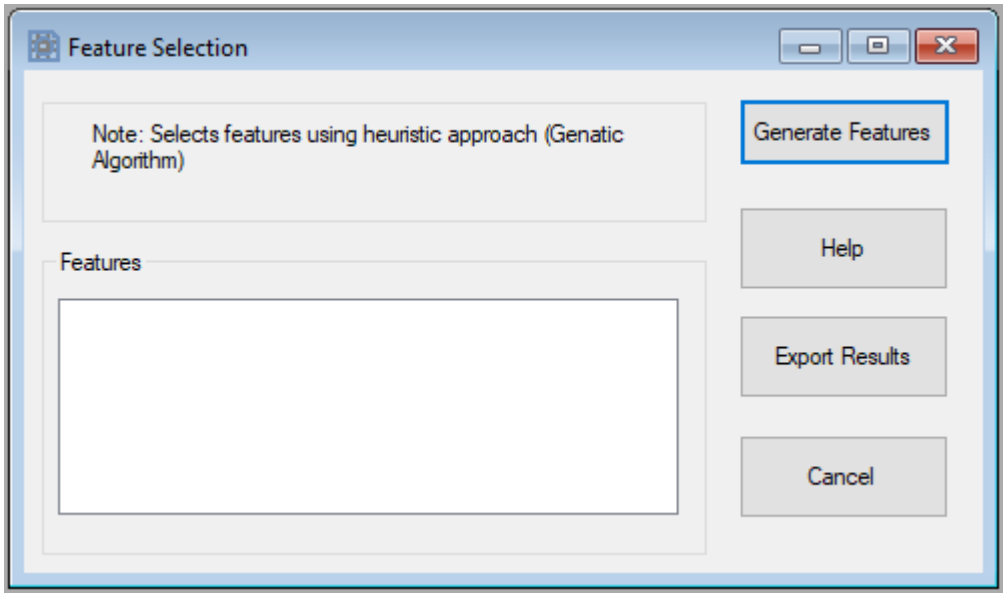
Tool provides semi exhaustive (both Forward Navigation and Backward Elimination) approaches and heuristics based approach (Genetic Algorithm and Random Feature Vectors)

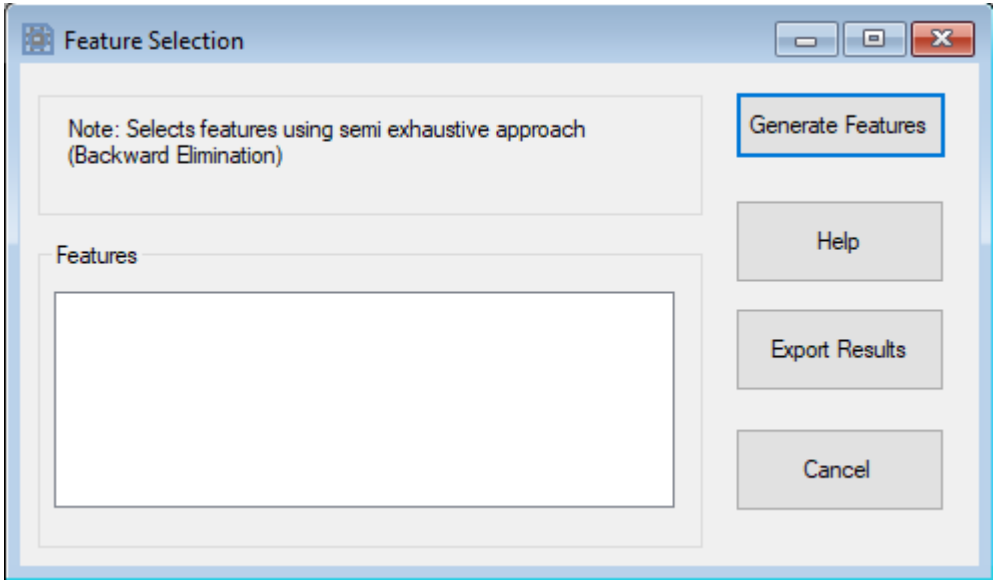
The task can be performed from:

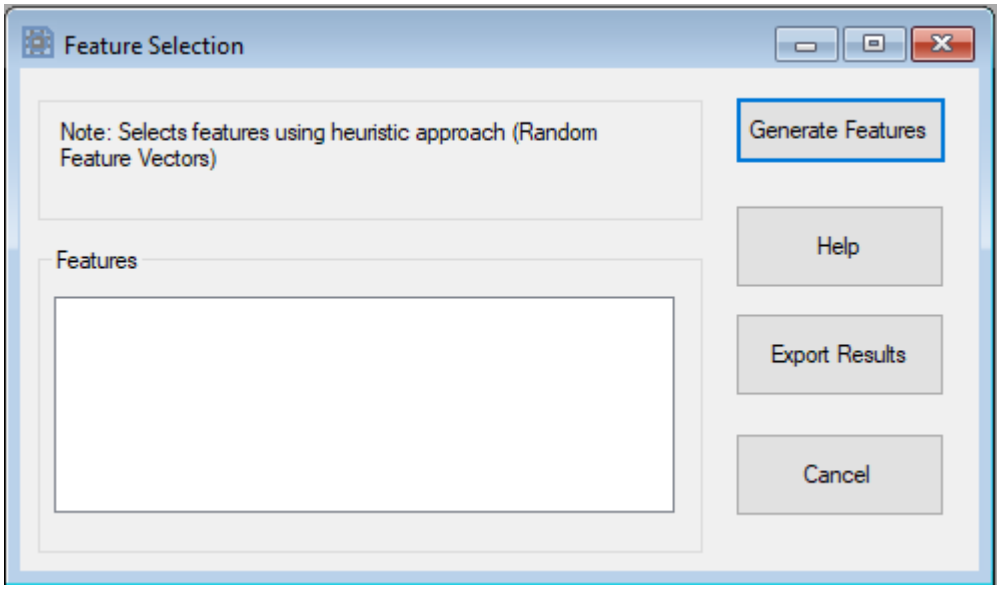
1. **Menu:**
    a. **RST > Feature Selection > Forward Navigation**
    b. **RST > Feature Selection > Backward Elimination**
    c. **RST > Feature Selection > Genetic Algorithm**
    d. **RST > Feature Selection > Random Feature Vectors**
2. **Task Panel**

Figures below show each of the forms used to perform feature selection using the above mentioned methods:




Feature Selection
Note: Selects features using heuristic approach (Genatic Algorithm)
Generate Features
Help
Features
Export Results
Cancel


Feature Selection
Note: Selects features using semi exhaustive approach (Backward Elimination)
Generate Features
Help
Features
Export Results
Cancel



By clicking the "Generate Features" button on each form, user can generate features by corresponding feature selection algorithm. The features will be shown in the features listbox at the bottom of the form.

Same task can be performed from the Task Panel.

Note: To perform feature selection, dataset should not contain any missing value.

1.5 Rule Extraction:

Once the Reducts are generated, we can use the Reduct attributes for generation of decision rules. Various algorithms in the first version will be provided to generate the rules from Reducts. Both the local and global Reducts will be used to generate the rules.

Tool Support:

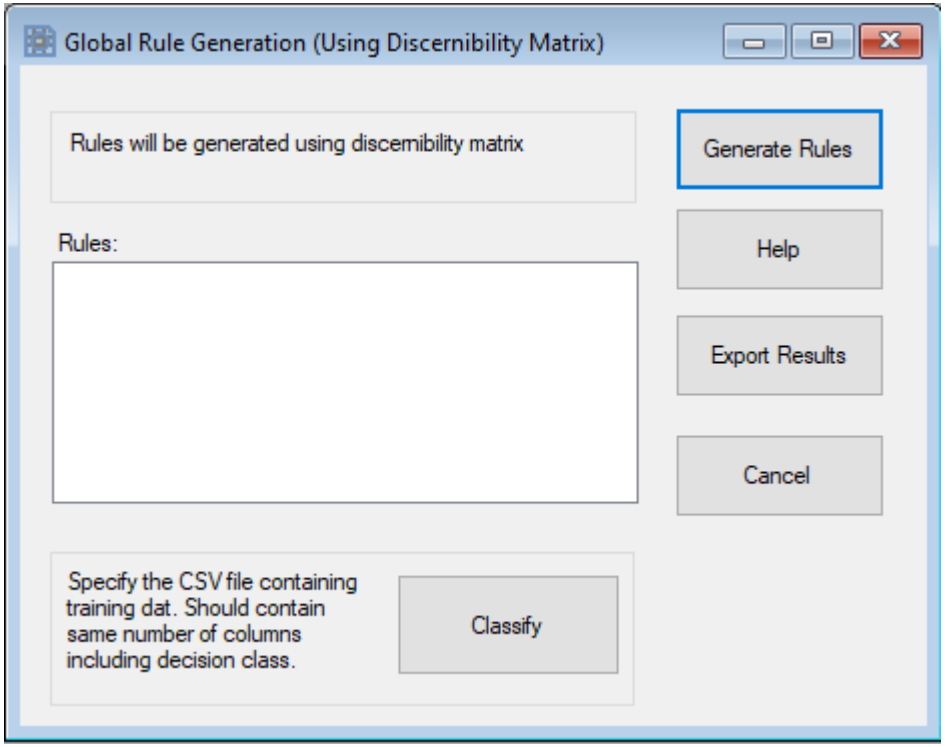
Tool supports generation of rules both from global and local reducts. Rules from global reucts can be generated from:

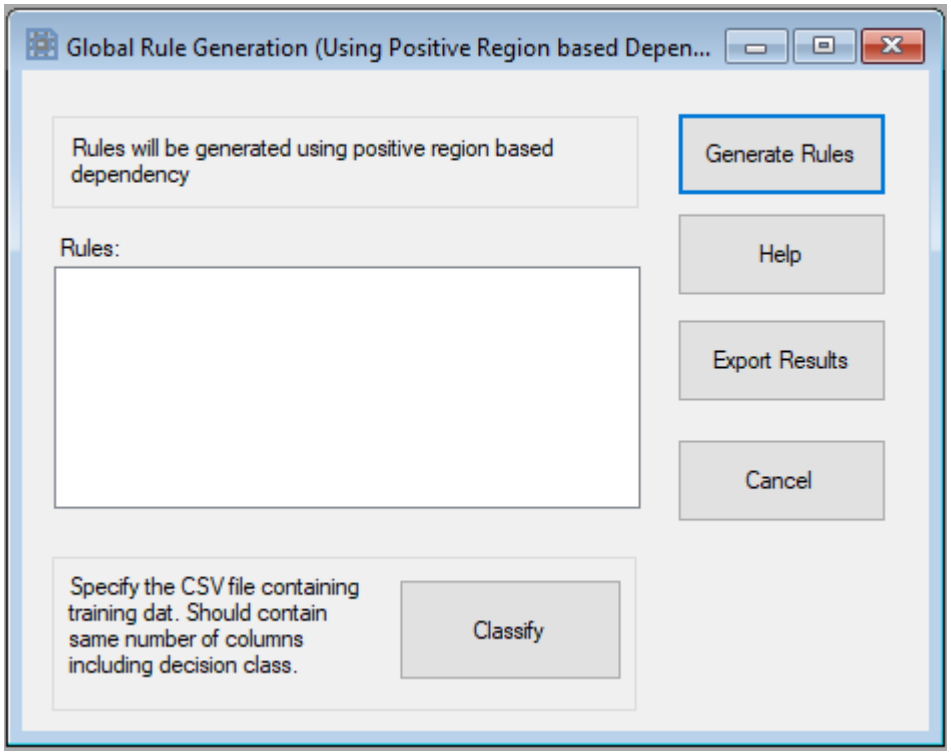
1. **Menu:**

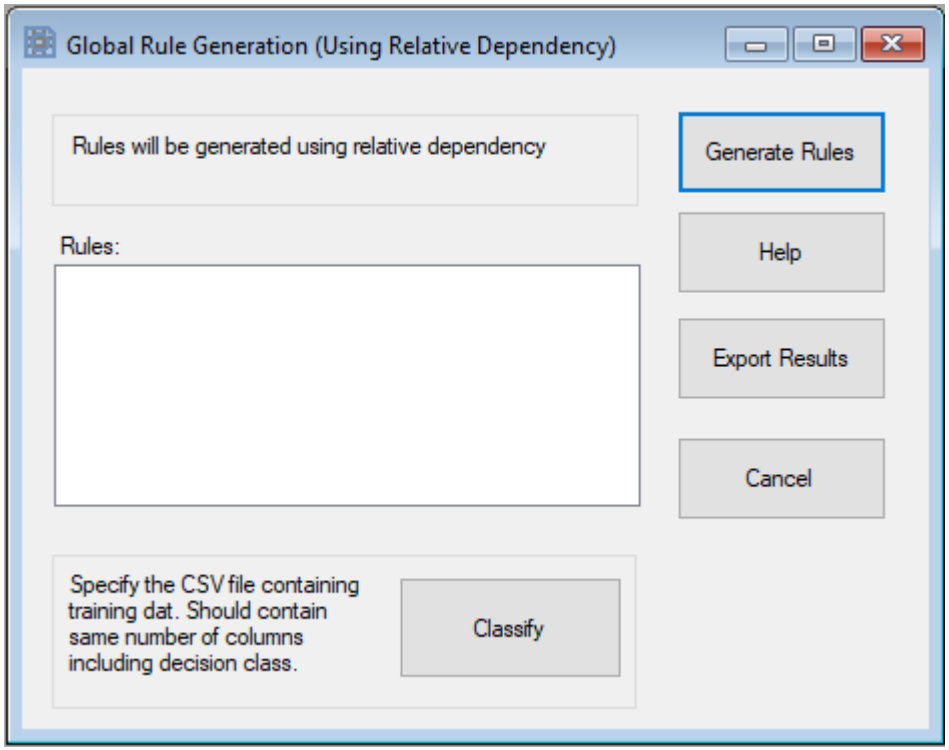
a. **Menu: RST > Generate Rules (Using Global Reducts) > Using Discernibility Matrix**
b. **Menu: RST > Generate Rules (Using Global Reducts) > Using Positive Region based Dependency**
c. **Menu: RST > Generate Rules (Using Global Reducts) > Relative Dependency**

2. **Task Panel**

Figures below show each of the form used to generate rules using the above mentioned methods:




Global Rule Generation (Using Positive Region based Depen...
Rules will be generated using positive region based dependency
Generate Rules
Help
Rules:
Export Results
Cancel
Specify the CSV file containing training dat. Should contain same number of columns including decision class.
Classify



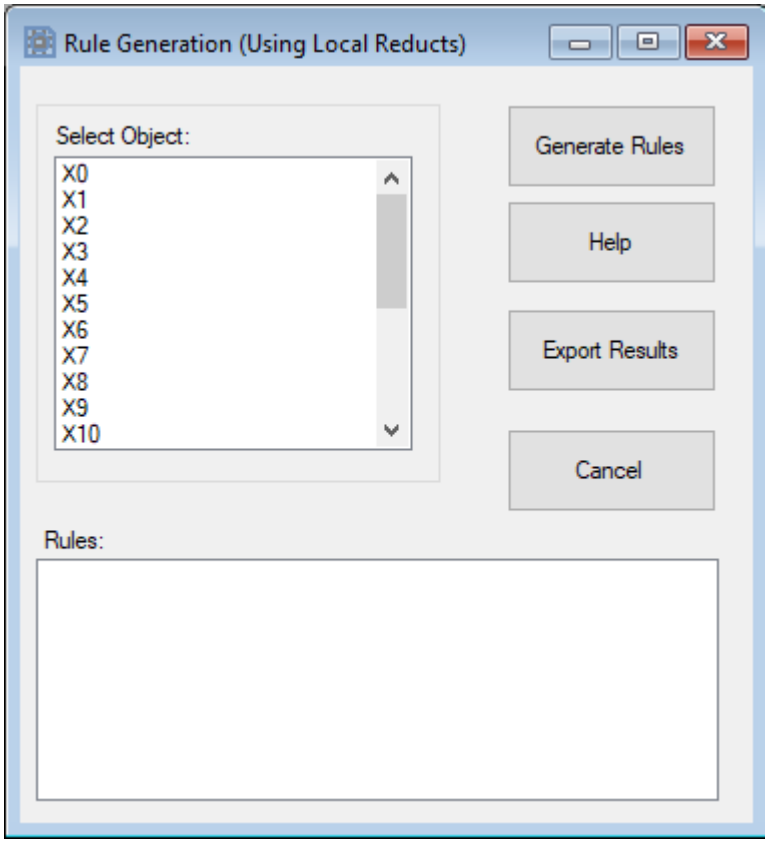
By clicking the “Generate Rules” button, the rules will be generated by using Reducts. Rules are generated by first identifying the reducts and then the values of features are taken from each set of indiscernible objects. All indiscernible objects (w.r.t.) a set of features have same value of those features.

Rules will be generated only if dataset does not contain any missing values.

Rules from Local Reucts can be generated from:

1. Menu: RST > Generate Rules (Using Local Reducts)
2. Task Panel

As shown in picture of the form below:



By clicking the “Generate Rules” button, the rules will be generated by using Local Reducts w.r.t. the selected object.

Rules are generated in the form of Key-Value pair e.g.

Rule: (C2:1.4),(C3:0.2) -> (C4: Iris-setosa)

Means that:

If (C2=1.4) and (C3=0.2) then C4=Iris-Setosa.

Note: Rules will be generated only if dataset does not contain any missing values.

Task: 1.6 Classification:

Classification is an important data mining and analysis task. Once the rules are generated as mentioned in section 1.4, the rules can be used to label the unknown data. So, the user will be able to perform classification using the tool.

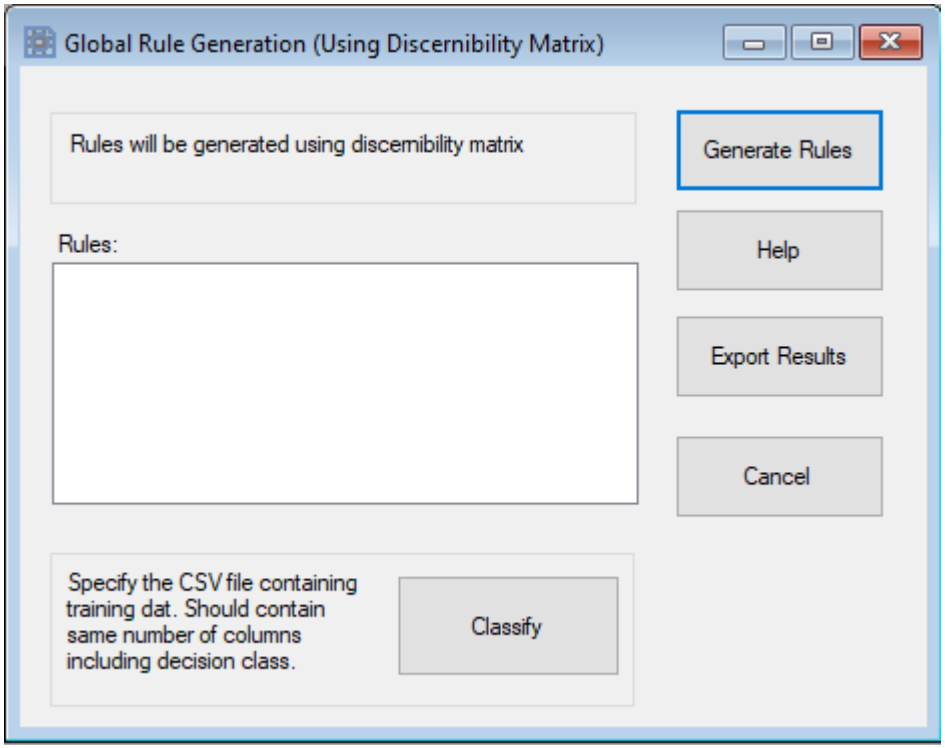
Tool Support:

Tool lets the user to perform classification tasks once the Rules are generated using Global Reduct.

By clicking the menu:

1. Menu: RST > Generate Rules (Using Global Reducts) > Using Discernibility Matrix
2. Menu: RST > Generate Rules (Using Global Reducts) > Using Positive Region based Dependency
3. Menu: RST > Generate Rules (Using Global Reducts) > Relative Dependency

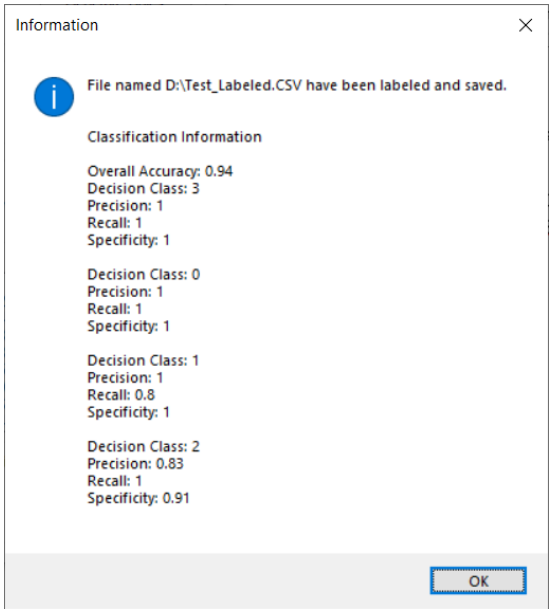
Rule Generation forms using global reducts will be shown. Rule generation form using discernibility matrix is shown below:



By clicking the classify button, “Open File” dialog box will appear from where the path of the file containing the test data will be given. The file containing the test data should be a CSV file and not having any missing value. The dataset in the file should have the same number of features as that of the currently loaded dataset.

Once the training data file is selected and “Open” button on “Open file” dialogue is clicked, the tool will label (classify) the test data and the file will be saved at “D” drive with the name “Test_Labeled.CSV”.

Tool will then display the classification metrics including Overall Accuracy and Precision, Recall and Specificity for each class in the form of MessageBox as follows:



Note: to perform classification, the test data should contain same number of features as that of in currently loaded dataset (through which rules were generated).

2. Dominance based Rough Set Approach (DRSA):

DRSA is an extension of classical RST. The tool will provide all the preliminary functionality of DRSA.

Tool Support:

Tool provides the preliminary functionality of DRSA. In next sessions we will explain the details.

2.1 Data:

All the tasks mentioned in section 1.1 will be provided here as well.

Tool Support:

Tools lets open the datasets from CSV file. details have been provided earlier about how to open a dataset.

2.2 Data Normalization:

User will be able to perform data normalization e.g. converting string attributes to corresponding numeric values in order to calculate dominance. The normalization will be done by two methods:

- **Manual:** User himself will be able to specify the order of the values belonging to value set of the attributes.
- **Automatic:** tool itself will be able to assign the order to the attribute values. A user can select any of these two modes.

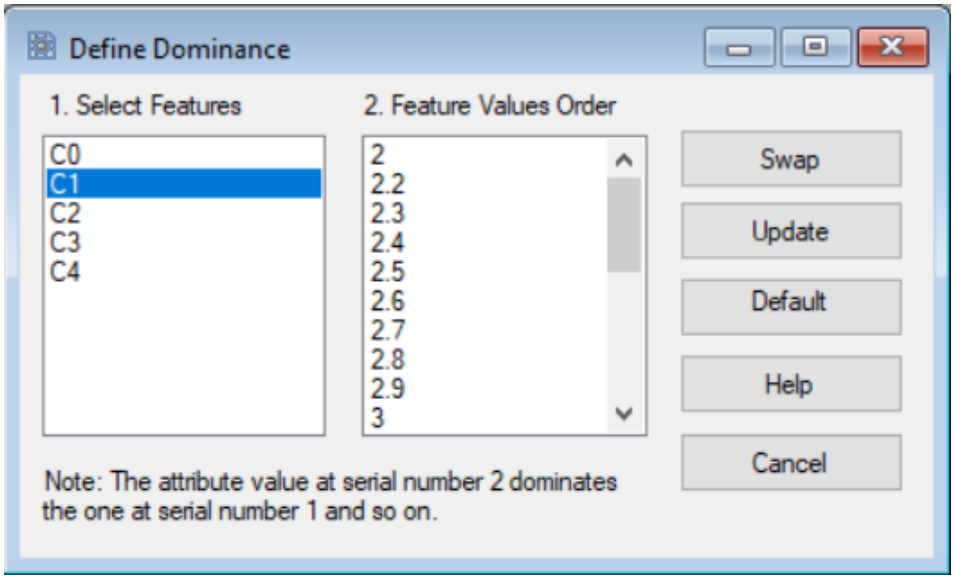
Tool Support:

Tool supports normalization of dataset (defining dominance) both manually and automatically. Tool uses dictionary concept to store the values and uses their indexes to perform operations. User, however, is always shown the original values. Whenever a dataset is loaded, the dominance of the values is defined by **default (automatically)**. For this purpose Dataset class (in source code) has a special method named "NormalizeData". For each attribute that is read from the file the values are stored in dictionary in sorted order (the features containing missing values are not normalized by default).

The same task can be performed from:

1. Menu: DRSA > Define Dominance
2. Task Panel

The form used to define dominance is shown below. Order of values can be checked as follows:



The “Define dominance” panel above shows that the feature “C1” has a number of values and the value “2.3” is better than “2.2” which is again better than “2”.

We can change the order **(manually)** by selecting a value and then clicking the “Swap” button. The order (dominance) of the selected value will be swapped with that of the value before it. After clicking the “Update” button, the mentioned order will be saved for that feature.

“Default” button can be used to set the default dominance order where the values in each feature are ordered according to datatype of that feature.

2.3 DRSA Preliminaries:

Task: 2.3.1 Class Unions:

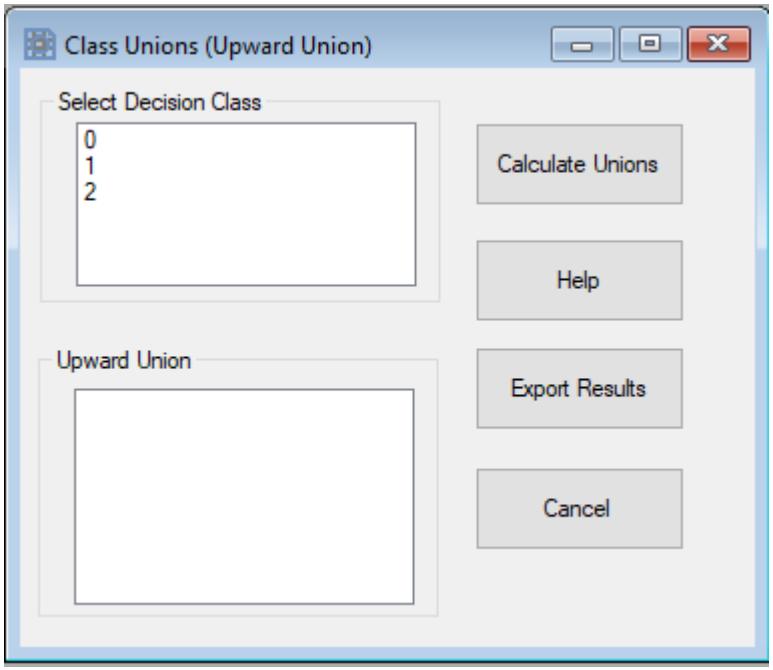
User will be able to find class union for any decision class. Both upward and downward unions will be provided.

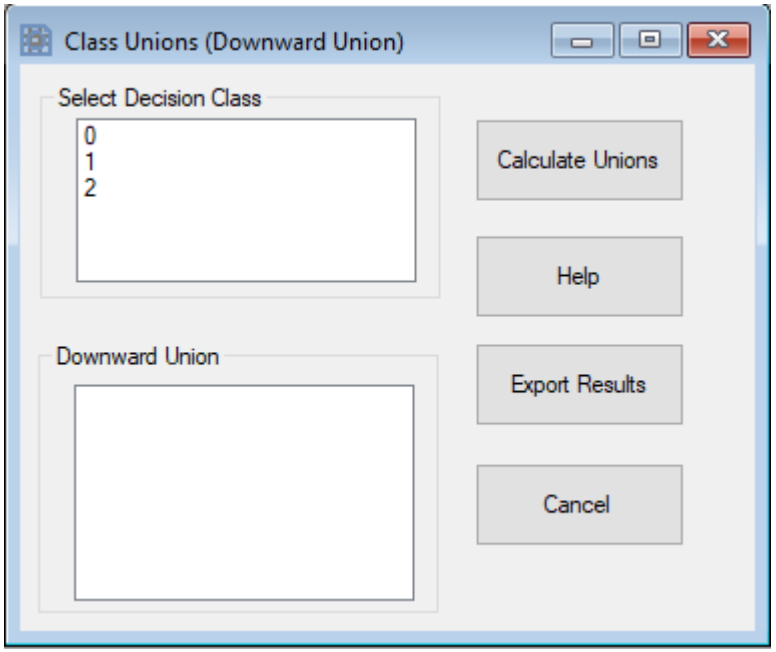
Tool Support:

Tool provides support to upward and downward union of classes. The task can be performed from:

1. **Menu**
    a. **Menu: DRSA > Calculate Union of Classes > Upward Union**
    b. **Menu: DRSA > Calculate Union of Classes > Downward Union**
2. **Task Panel**

Forms used to calculate union are shown below:





When user selects any class and clicks “Calculate Unions” button, the upward and downward union of classes will be displayed in corresponding Listboxes in each form.

The same task can be performed from Task Panel.

Note: To find inconsistent objects, dataset should not contain any missing value.

Task: 2.3.2 Calculation of Dominance:

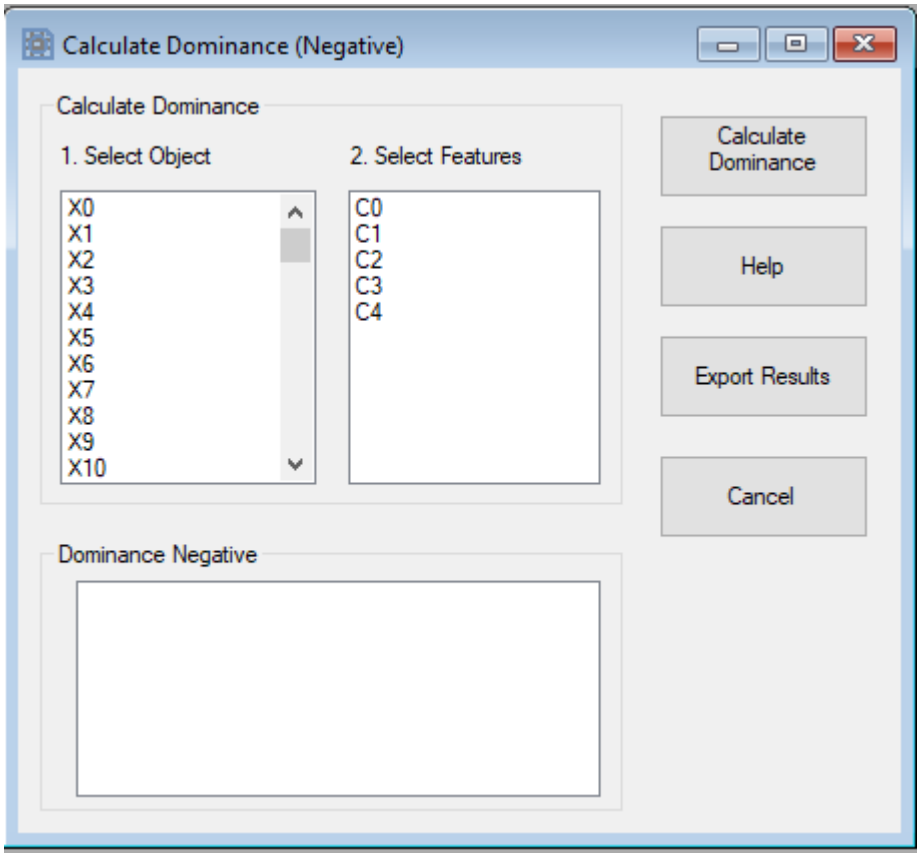
Tool will provide the functionality to calculate both positive and negative dominances with respect to any set of conditional attributes provided by the user. Conventional approach will be used.

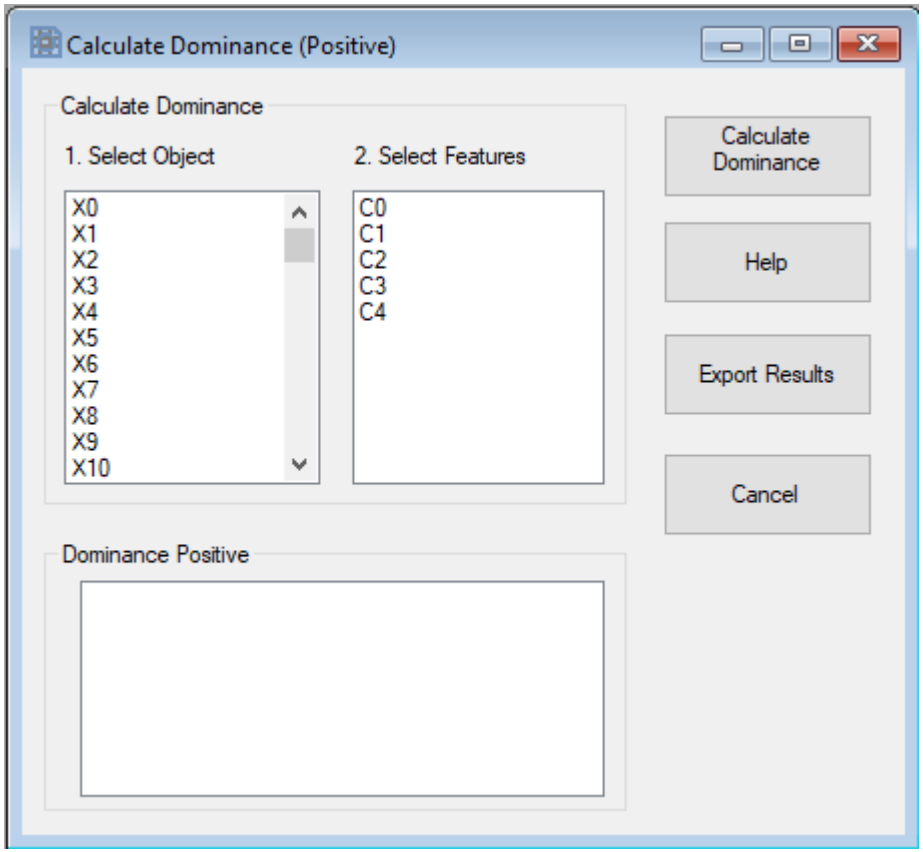
Tool Support:

Tool provides support to calculate both positive and negative dominances of an object with respect to any set of conditional attributes. The task can be performed from:

1. **Menu:**
    a. **DRSA > Calculate Dominance > Dominance Negative**
    b. **DRSA > Calculate Dominance > Dominance Positive**
2. **Task Panel**

Figures below show the forms used to calculate both positive and negative dominance using the above mentioned menu.


Calculate Dominance (Negative)
Calculate Dominance
1. Select Object
2. Select Features
X0
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
C0
C1
C2
C3
C4
Calculate Dominance
Help
Export Results
Cancel
Dominance Negative



When user selects any object and the feature(s), then by clicking the “Calculate Dominance” button, both the dominance (positive or negative) of the object will be shown in corresponding listboxes on the form.

The same task can be performed from Task Panel.

Note: To find inconsistent objects, dataset should not contain any missing value.

Task: 2.3.3 Calculation of Consistent and Inconsistent Objects:

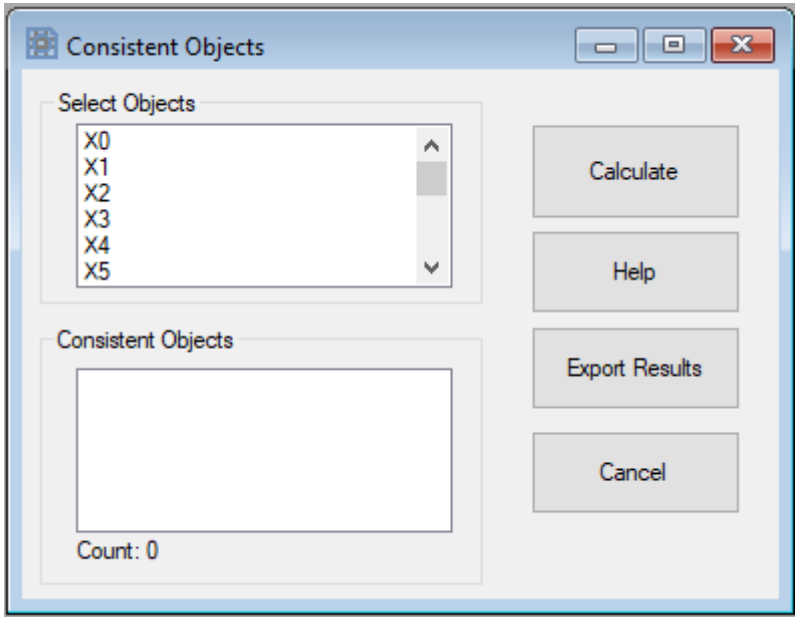
Dominance Principle states that if an object *x* dominates *y* with respect to all conditional attributes, its decision class should dominate that of object y as well. System will be able to identify all the consistent and inconsistent objects.

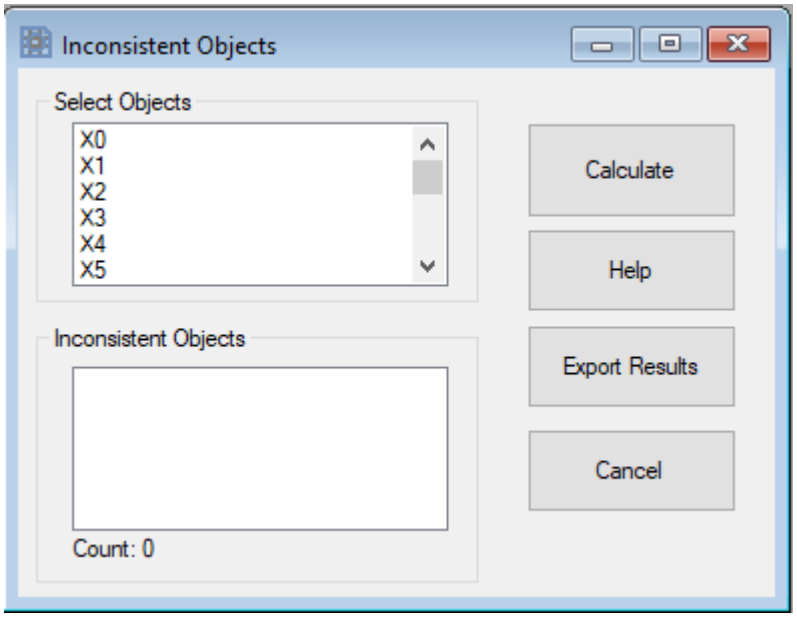
Tool Support:

Tool provides support to calculate both consistent and inconsistent objects w.r.t. any reference object. The task can be performed from:

1. Menu: DRSA > Find Consistent/Inconsistent objects > Consistent Objects
2. Menu: DRSA > Find Consistent/Inconsistent objects > Inconsistent Objects
3. Task Panel

Forms used to calculate both consistent and inconsistent objects are given below:





When user selects any object and clicks “Calculate” button, the consistent or inconsistent objects w.r.t. the selected object will be shown in corresponding listboxes on the form.

The same task can be performed from Task Panel.

Note: To find inconsistent objects, dataset should not contain any missing value.

Task 2.3.4 Lower and upper approximations

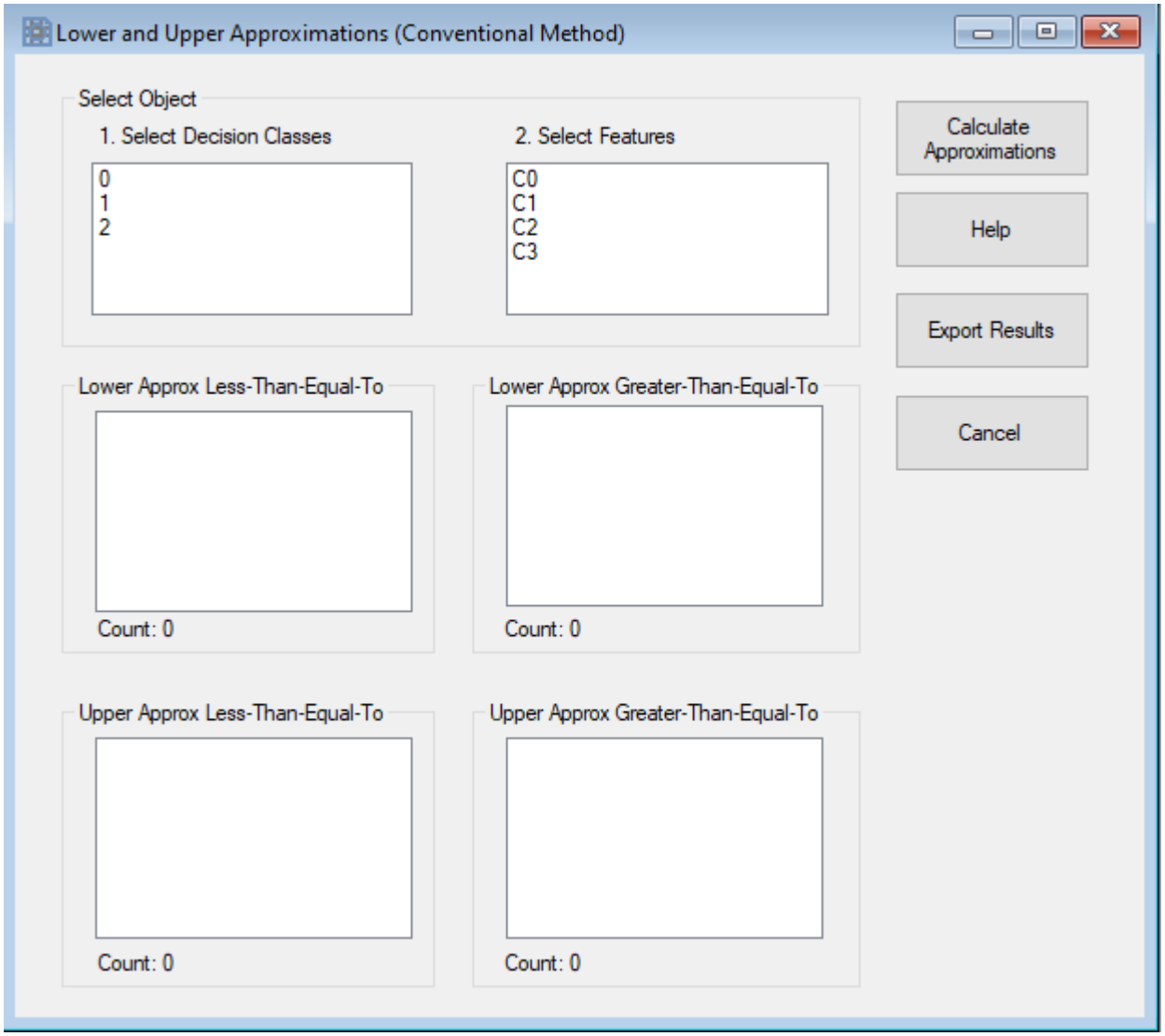
Approximations are important concept both in RST and DRSA. The tool will provide complete set of functionalities to calculate approximations for both upward and downward union of classes with respect to any set of attributes for any decision class. Implementation of both the conventional and advance methods will be provided.

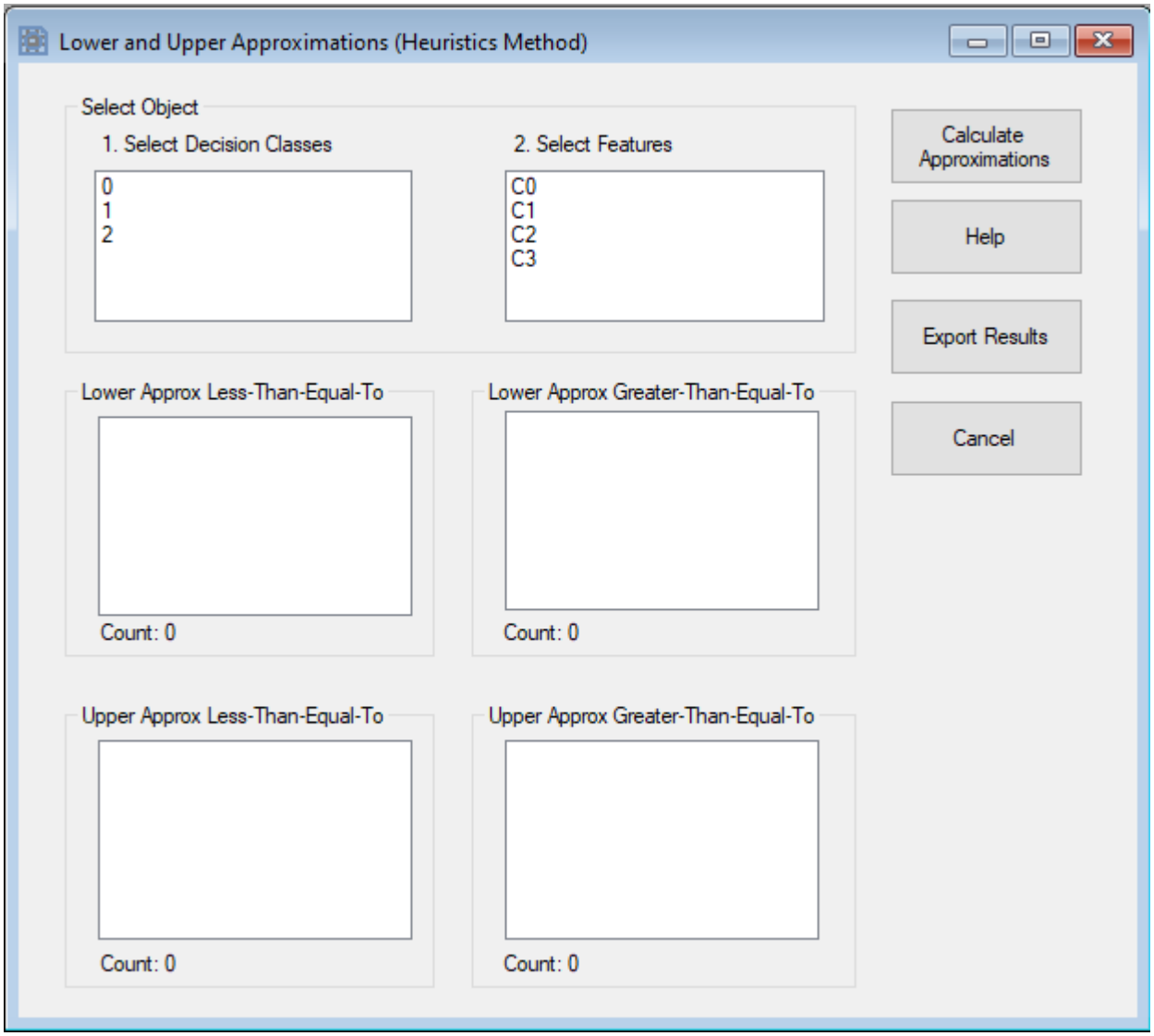
Tool Support:

Tool provides support to calculate approximations for both upward and downward union of classes with respect to any set of attributes for any decision class. Approximations can be calculated using both the conventional and heuristics-based approach. The task can be performed from:

1. Menu
    a. Menu: DRSA > Find Approximations > Conventional Method
    b. Menu: DRSA > Find Approximations > Heuristics based Method
2. Task Panel

Figures below show the forms used for calculating approximations


Lower and Upper Approximations (Conventional Method)
Select Object
1. Select Decision Classes
0
1
2
2. Select Features
C0
C1
C2
C3
Calculate Approximations
Help
Export Results
Cancel
Lower Approx Less-Than-Equal-To
Count: 0
Lower Approx Greater-Than-Equal-To
Count: 0
Upper Approx Less-Than-Equal-To
Count: 0
Upper Approx Greater-Than-Equal-To
Count: 0


Lower and Upper Approximations (Heuristics Method)
Select Object
1. Select Decision Classes
0
1
2
2. Select Features
C0
C1
C2
C3
Calculate Approximations
Help
Export Results
Cancel
Lower Approx Less-Than-Equal-To
Count: 0
Lower Approx Greater-Than-Equal-To
Count: 0
Upper Approx Less-Than-Equal-To
Count: 0
Upper Approx Greater-Than-Equal-To
Count: 0

When user selects any decision class and the features, then by clicking the “Calculate Approximations” button will display the Lower Approximations and Upward Approximations in the corresponding listboxes.

The same task can be performed from Task Panel.

Note: To find inconsistent objects, dataset should not contain any missing value.

Task: 2.3.5 Quality of approximation:

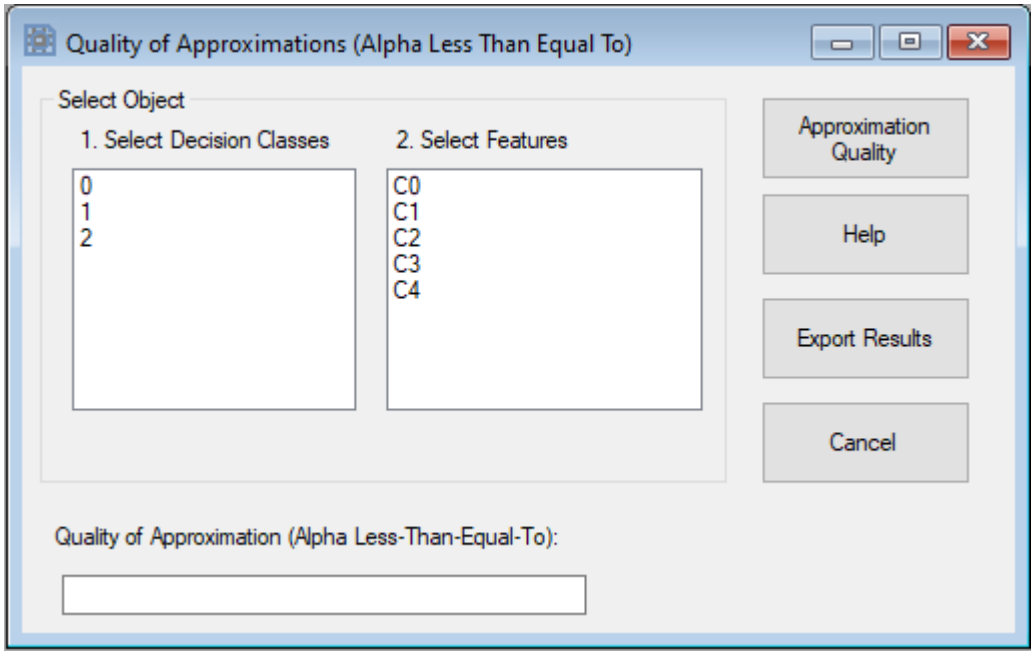
The quality of approximation of the ordinal classification *Cl* by a set of attributes *P* is defined as the ratio of the number of objects P-consistent with the dominance principle and the number of all the objects in *U*. Tool will provide the functionality to calculate quality of approximation for all approximations mentioned in section 2.3.4.

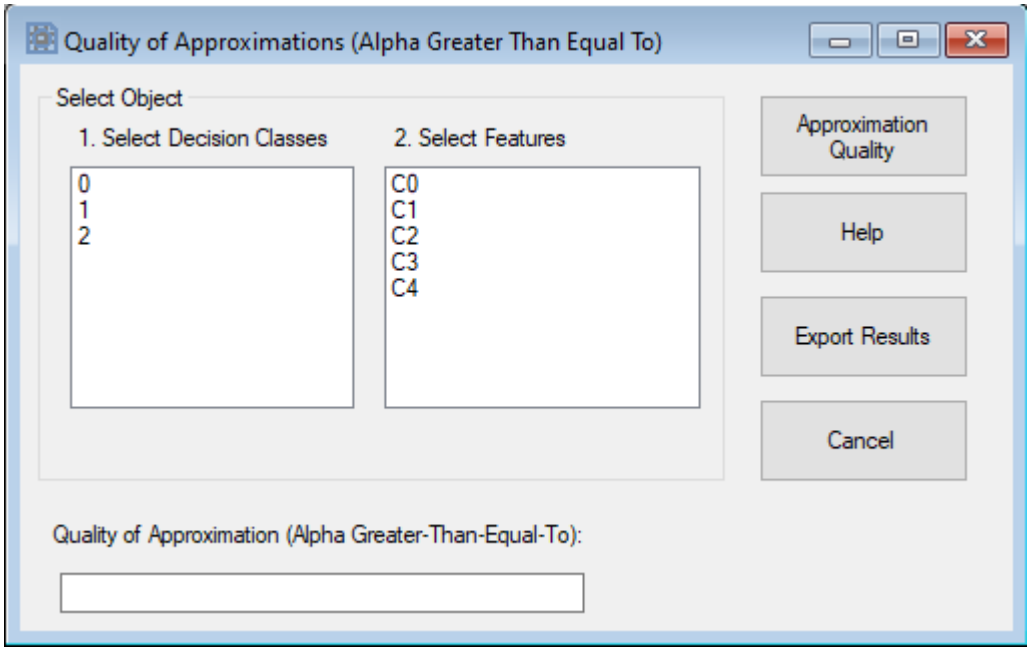
Tool Support:

Tool provides support to calculate quality of approximations using lower and upper approximations. The task can be performed from:

1. **Menu:**
    a. **Menu: DRSA > Find Approximation Quality > Alpha Less-Than-Equal-To**
    b. **Menu: DRSA > Find Approximation Quality > Alpha Greater-Than-Equal-To**
2. **Task Panel**

Figures below show the forms used to calculate approximation quality.


Quality of Approximations (Alpha Less Than Equal To)
Select Object
1. Select Decision Classes
2. Select Features
0
1
2
C0
C1
C2
C3
C4
Approximation Quality
Help
Export Results
Cancel
Quality of Approximation (Alpha Less-Than-Equal-To):



When user selects any decision class and the features, then by clicking the "Approximation Quality" button displays the quality of approximations calculated using upward union of classes (Alpha Greater-Than-Equal-To) and downward union of classes (Alpha Less-Than-Equal-To).

The same task can be performed from Task Panel.

Note: To find inconsistent objects, dataset should not contain any missing value.

Task: 2.3.6 Reducts:

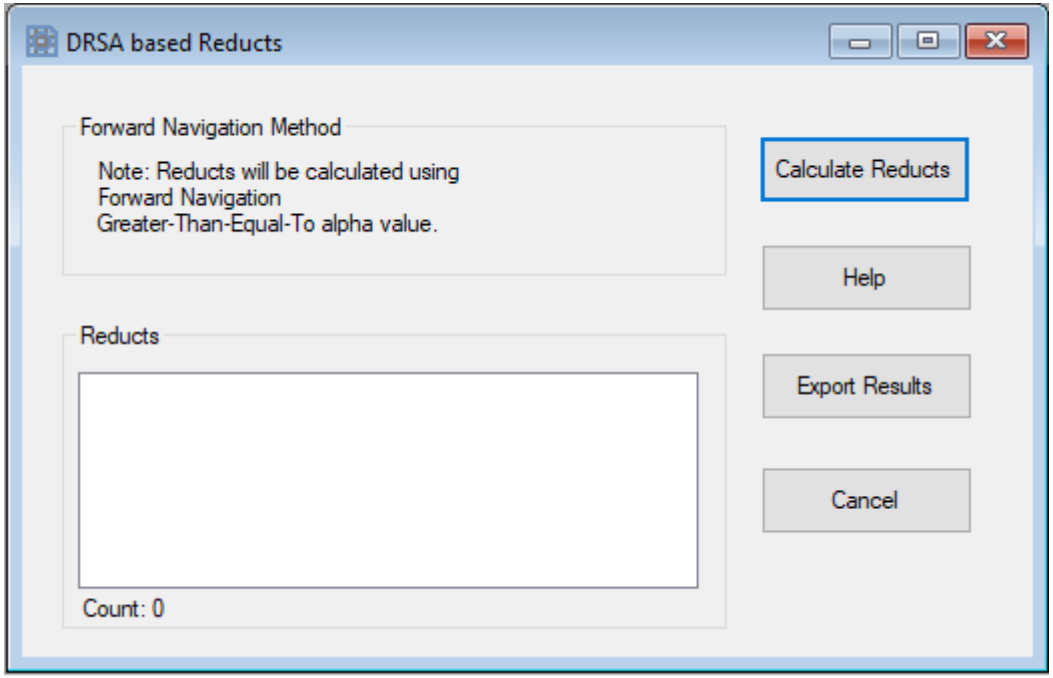
Tool will provide the complete set of functionalities to calculate Reducts. Different Reduction algorithms will be provided for the user to execute and compare with their own developed algorithms.

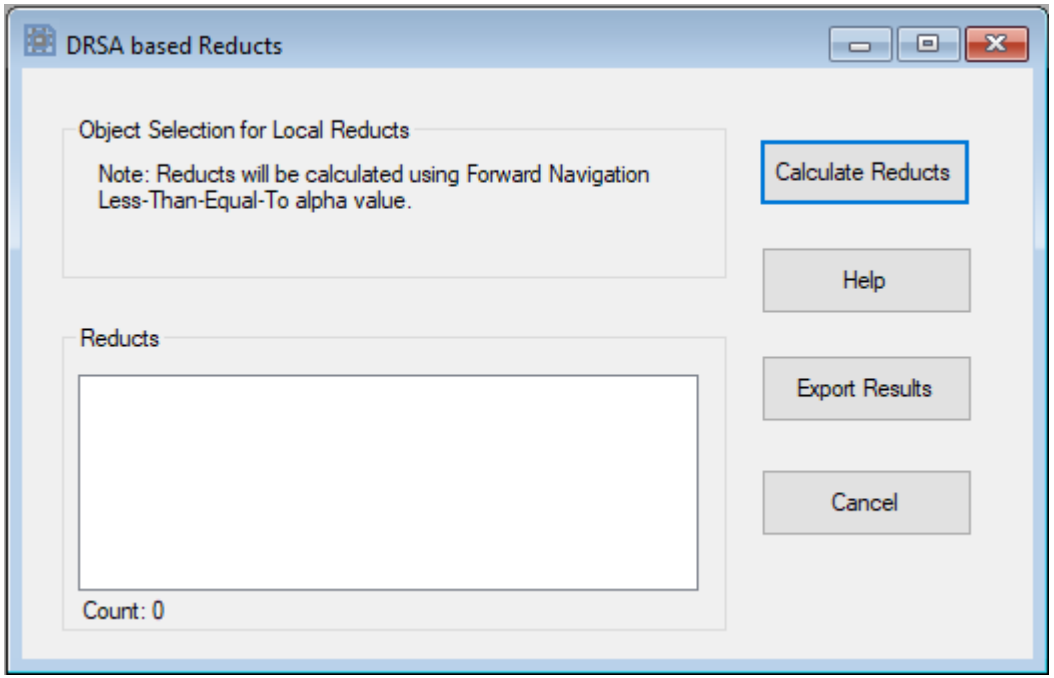
Tool Support:

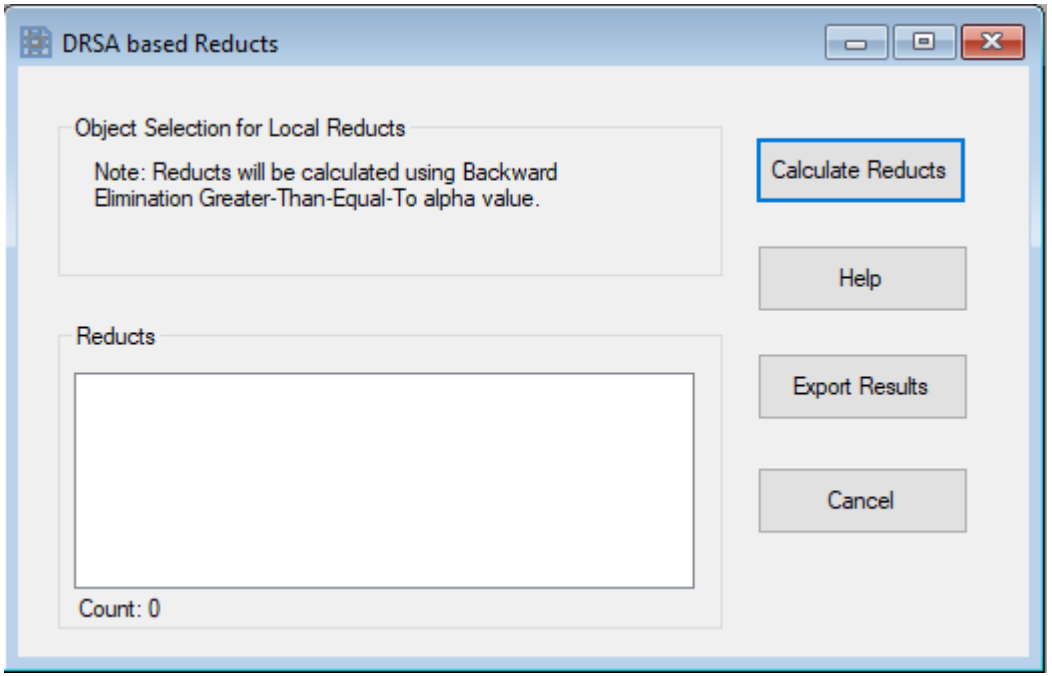
Tool provides support to calculate reducts using the accuracy of approximations measure (for upward and downward union of classes). The task can be performed from:

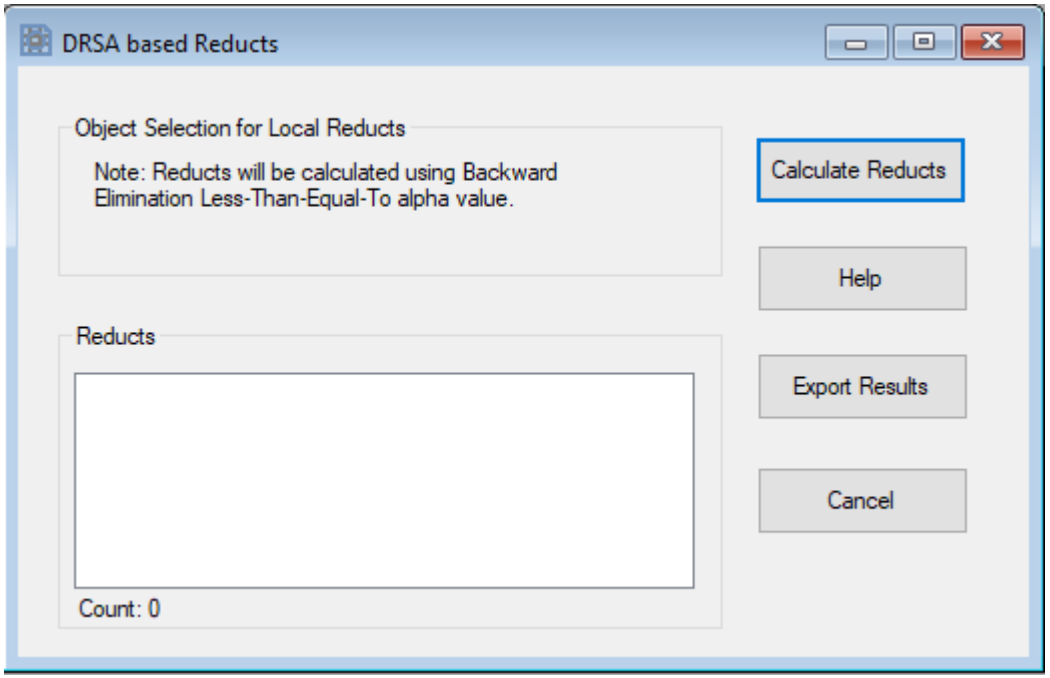
1. **Menu:**
    a. **Menu: DRSA > DRSA based Reducts > Forward Navigation Greater-Than-Equal-To**
    b. **Menu: DRSA > DRSA based Reducts > Forward Navigation Less-Than-Equal-To**
    c. **Menu: DRSA > DRSA based Reducts > Backward Elimination Greater-Than-Equal-To**
    d. **Menu: DRSA > DRSA based Reducts > Backward Elimination Less-Than-Equal-To**
2. **Task Panel**

Figures below show the forms used to calculate Reducts using above mentioned methods:




DRSA based Reducts
Object Selection for Local Reducts
Note: Reducts will be calculated using Forward Navigation Less-Than-Equal-To alpha value.
Calculate Reducts
Help
Reducts
Export Results
Cancel
Count: 0


DRSA based Reducts
Object Selection for Local Reducts
Note: Reducts will be calculated using Backward Elimination Greater-Than-Equal-To alpha value.
Calculate Reducts
Help
Reducts
Export Results
Cancel
Count: 0



By clicking "Calculate Reducts" button, the reducts will be generated using the corresponding reduct generation algorithm. The generated reducts will be displayed in the Reducts listbox at the bottom of the form.

The same task can be performed from Task Panel.

Note: To find inconsistent objects, dataset should not contain any missing value.

Task: 2.3.7 Decision Rules:

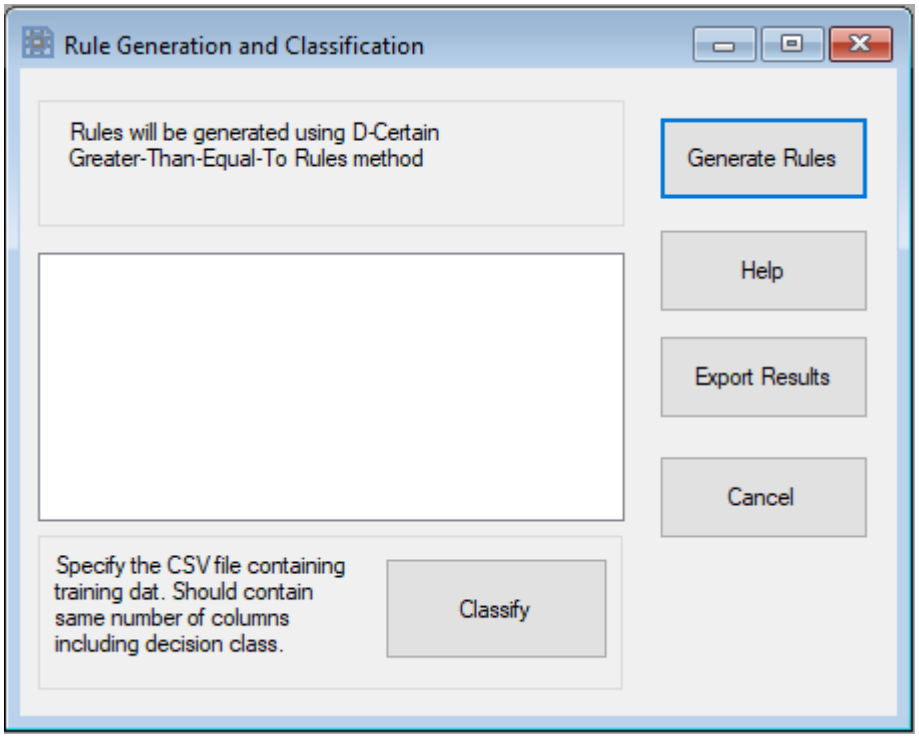
The dominance-based rough approximations of upward and downward unions of decision classes can serve to induce a generalized description of objects in terms of "if . . . , then . . . " decision rules. Tool will provide functionality to induce such decision rules. User will be able to extract both *Certain* and *Possible* types of rules.

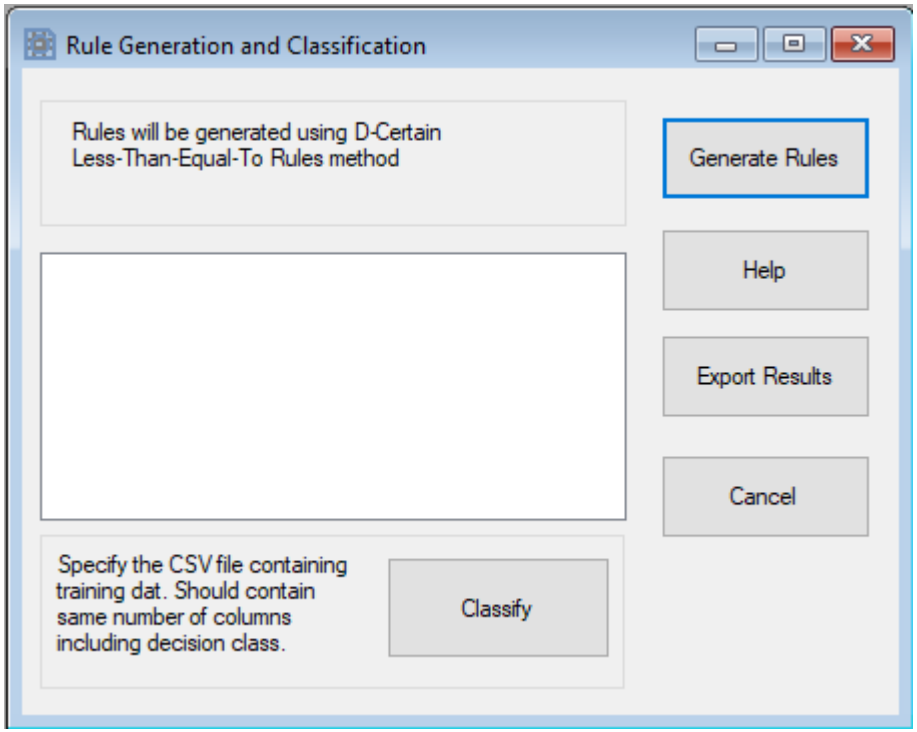
Tool Support:

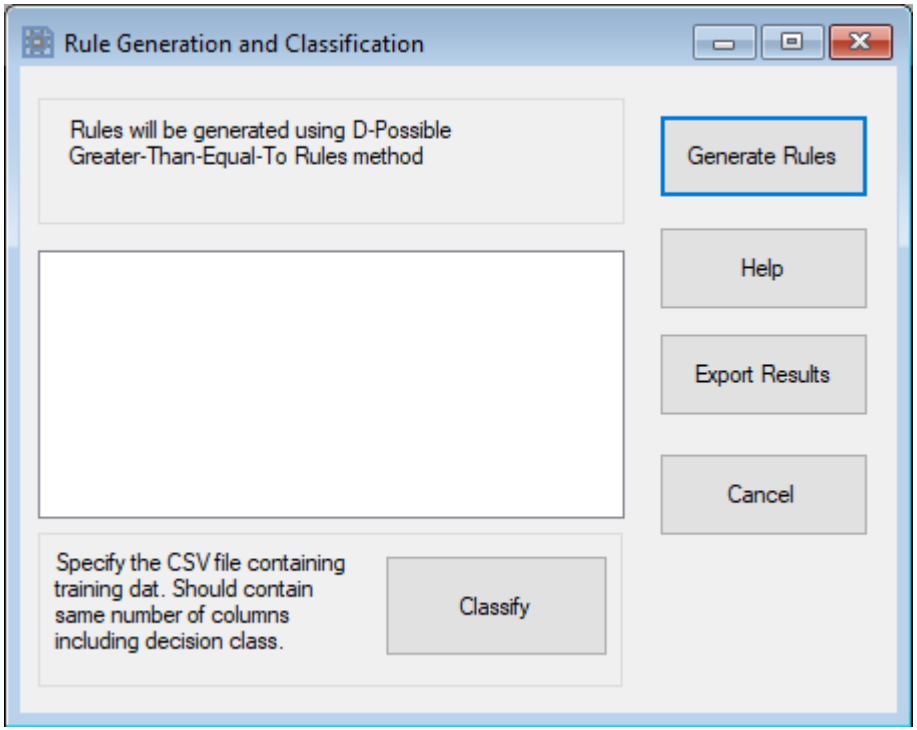
Tool provides support to extract both the Certain and Possible types of rules. The task can be performed from:

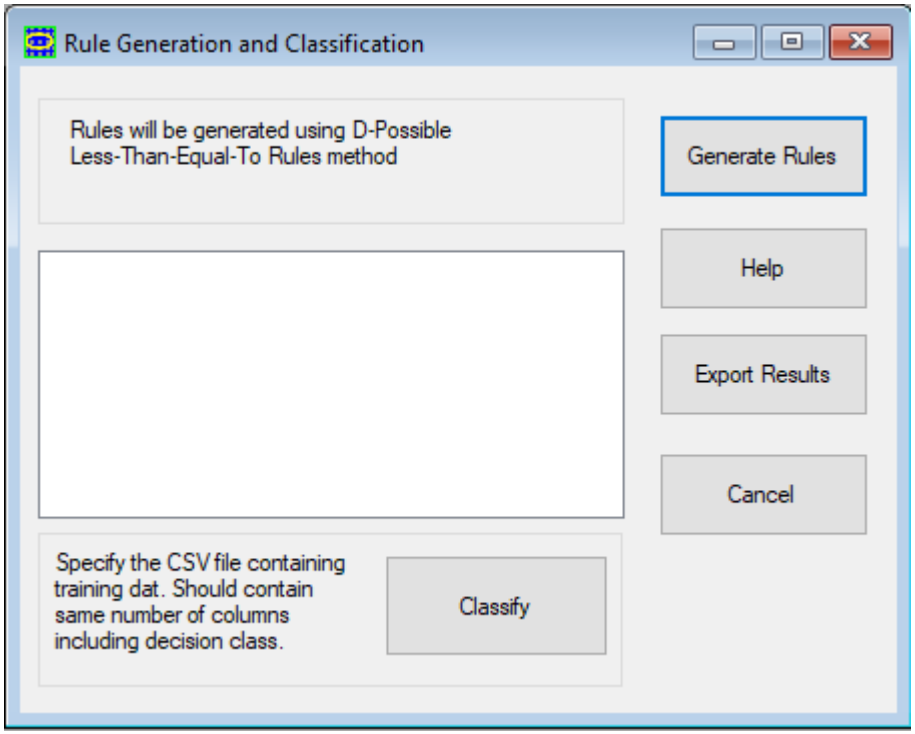
1. **Menu**
    a. **Menu: DRSA > DRSA based Rules and Classifications > D-Certain Greater-Than-Equal-To**
    b. **Menu: DRSA > DRSA based Rules and Classifications > D-Certain Less-Than-Equal-To**
    c. **Menu: DRSA > DRSA based Rules and Classifications > D-Possible Greater-Than-Equal-To**
    d. **Menu: DRSA > DRSA based Rules and Classifications > D-Possible Less-Than-Equal-To**
2. **Task Panel**

The forms used to calculate rules are shown below:




Rule Generation and Classification
Rules will be generated using D-Certain Less-Than-Equal-To Rules method
Generate Rules
Help
Export Results
Cancel
Specify the CSV file containing training dat. Should contain same number of columns including decision class.
Classify


Rule Generation and Classification
Rules will be generated using D-Possible Greater-Than-Equal-To Rules method
Generate Rules
Help
Export Results
Cancel
Specify the CSV file containing training dat. Should contain same number of columns including decision class.
Classify



By clicking any of the “Generate Rules” button, any of the D-Certain or D-Possible types of rules can be generated depending upon the button clicked. The generated rules are displayed at the bottom of the form.

The same task can be performed from Task Panel.

Note: To find inconsistent objects, dataset should not contain any missing value.

2.3.8 Classification:

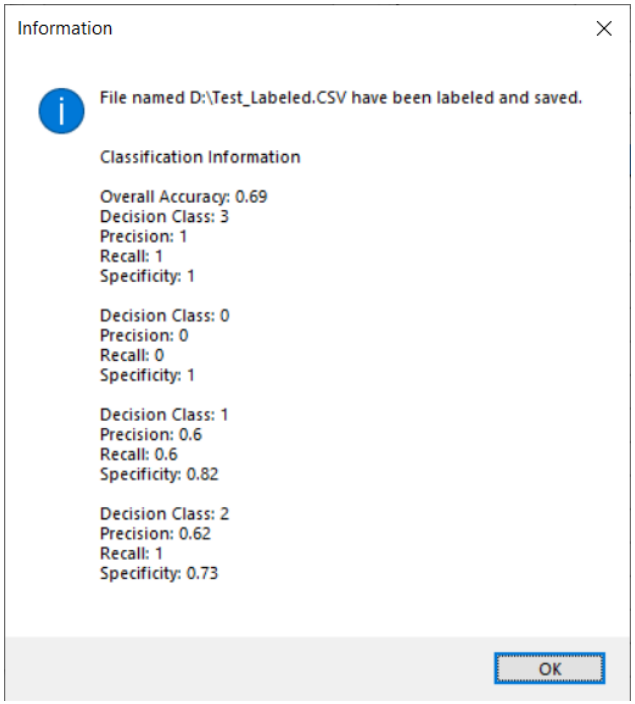
Once the rules are extracted, we can use them to perform classification task for unknown records. The tool will provide functionality to extract rules from training data and perform classification for test and other unknown data. Definitely the support to verify the accuracy of classification will also be available e.g. development of confusion matrix etc.

Tool Support:

After generating the rules mentioned in section 2.3.7, by clicking the classify button, “Open File” dialog box will appear from where the path of the file containing the test data will be given. The file containing the test data should be a CSV file and not having any missing value. The dataset in the file should have the same number of features as that of the currently loaded dataset.

Once the training data file is selected and “Open” button on “Open file” dialogue is clicked, the tool will label (classify) the test data and the file will be saved at “D” drive with the name “Test_Labeled.CSV”.

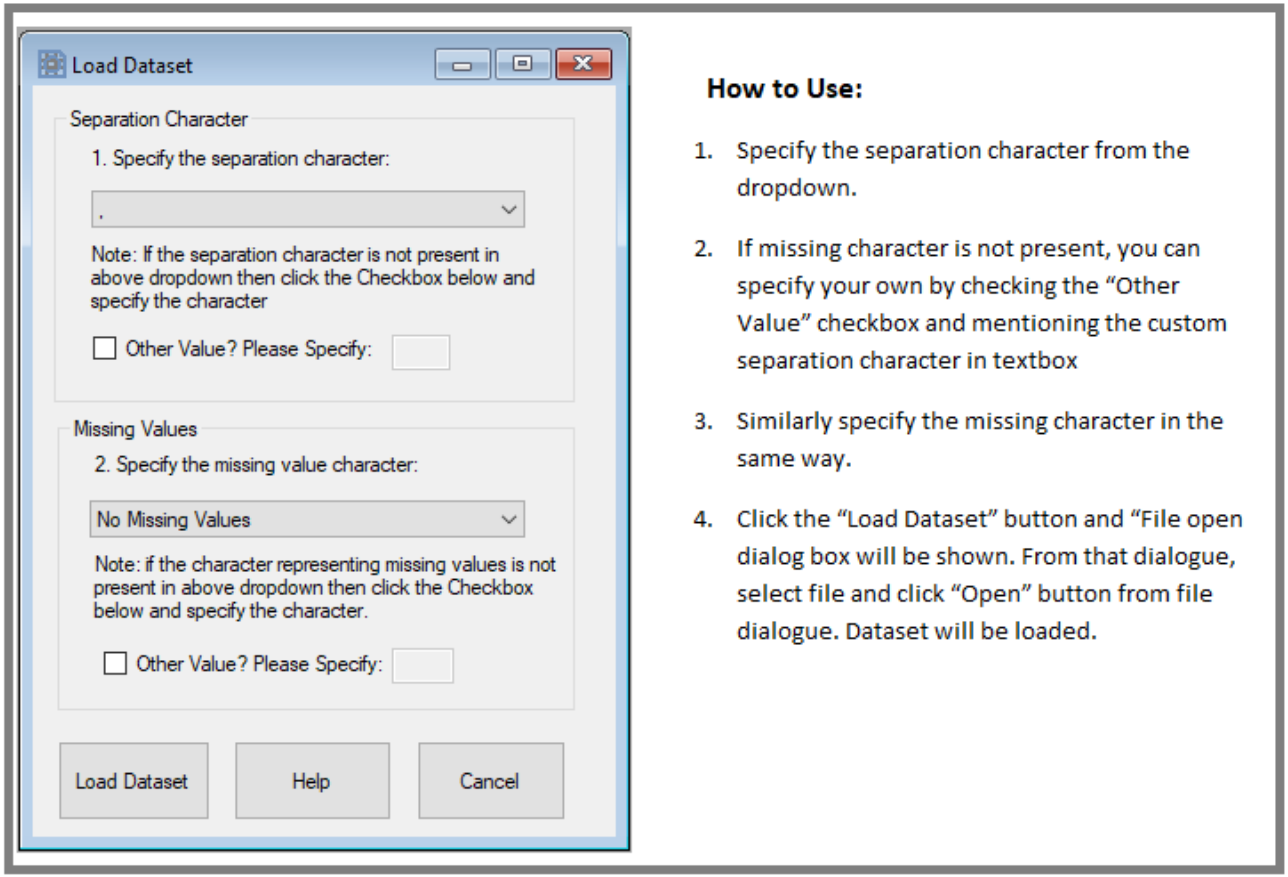
Tool will then display the classification metrics including Overall Accuracy and Precision, Recall and Specificity for each class in the form of MessageBox as follows:


Information
File named D:\Test_Labeled.CSV have been labeled and saved.
Classification Information
Overall Accuracy: 0.69
Decision Class: 3
Precision: 1
Recall: 1
Specificity: 1
Decision Class: 0
Precision: 0
Recall: 0
Specificity: 1
Decision Class: 1
Precision: 0.6
Recall: 0.6
Specificity: 0.82
Decision Class: 2
Precision: 0.62
Recall: 1
Specificity: 0.73
OK

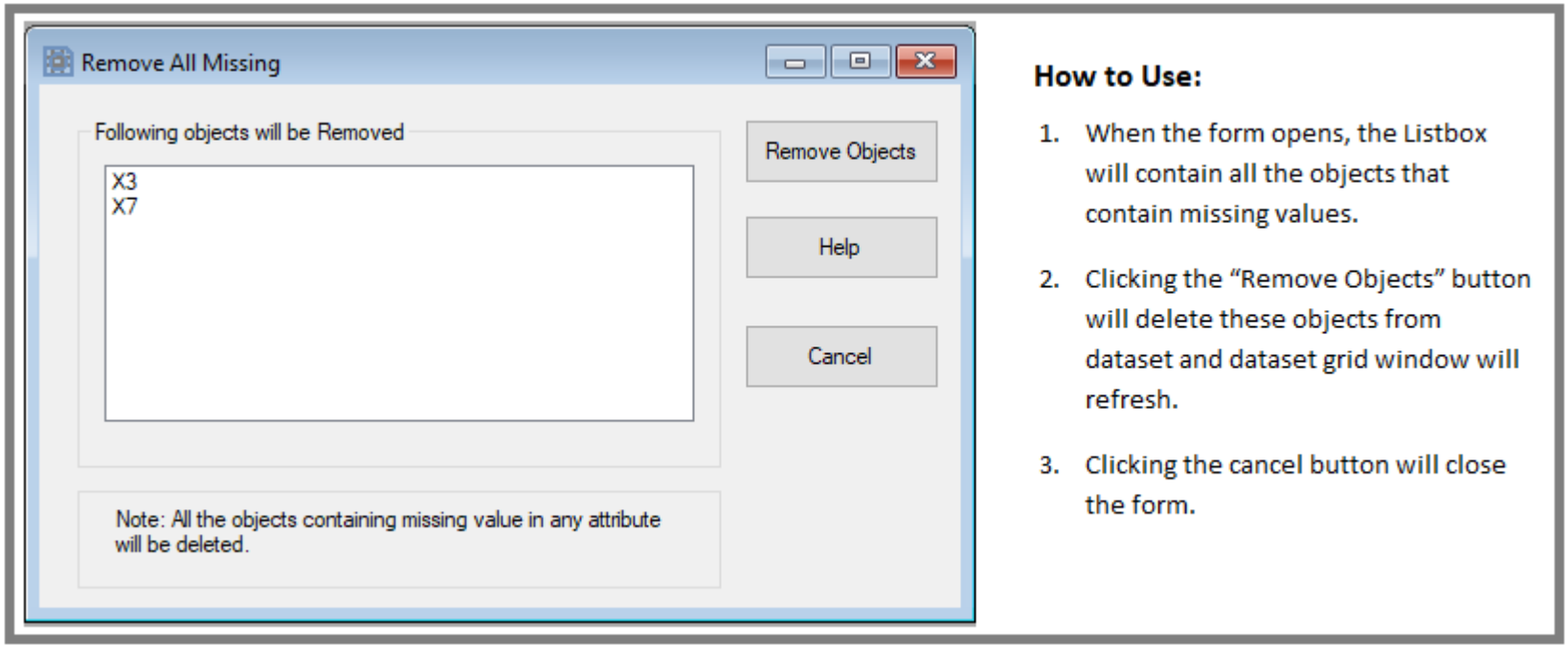
How to use each form

Now we will paste picture of each form along with sequence of steps to use that form.

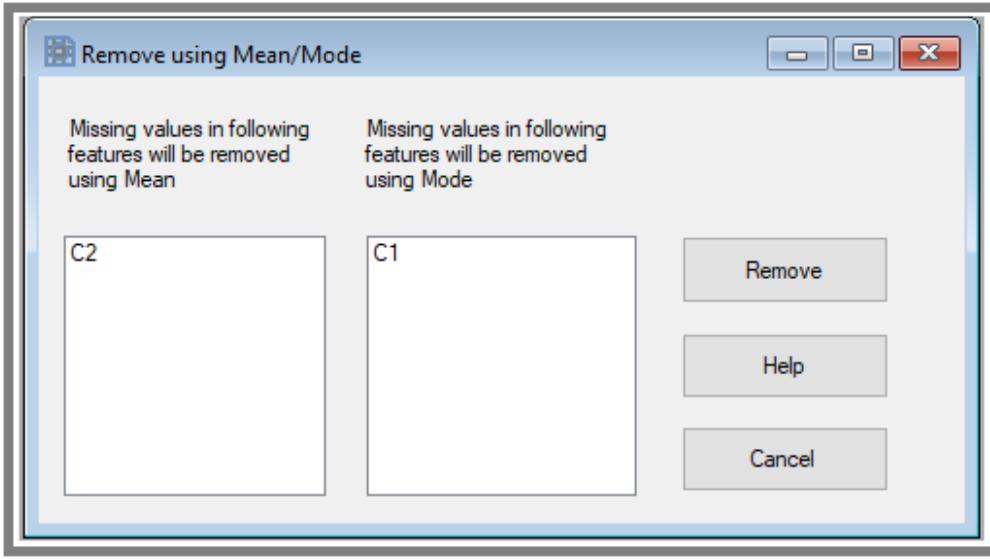
Form: Load Dataset



Form: Remove All Missing



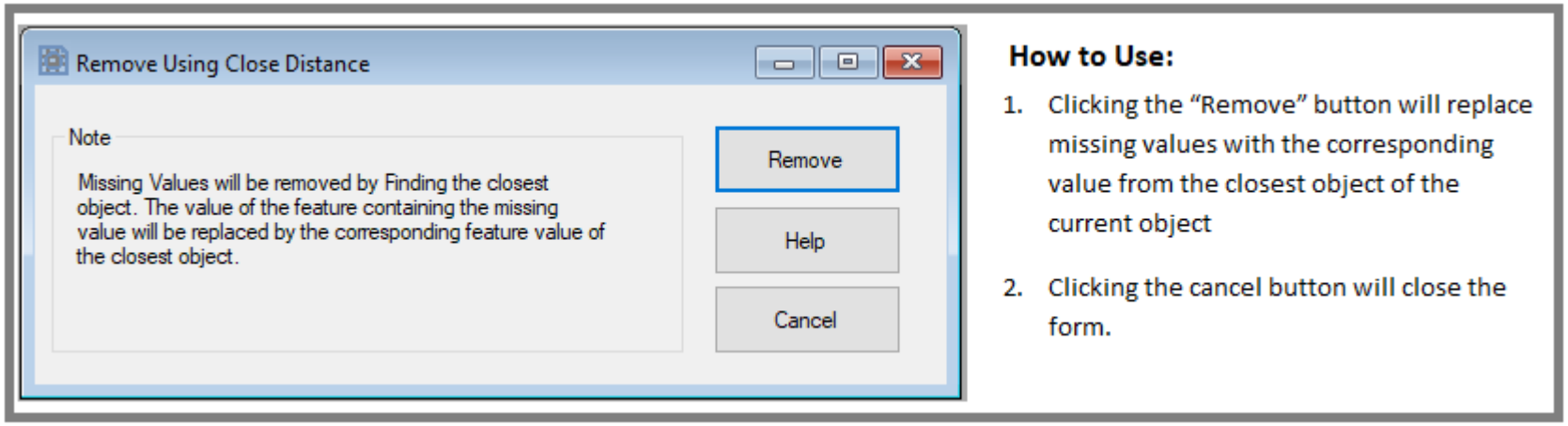
Form: Remove using Mean/Mode



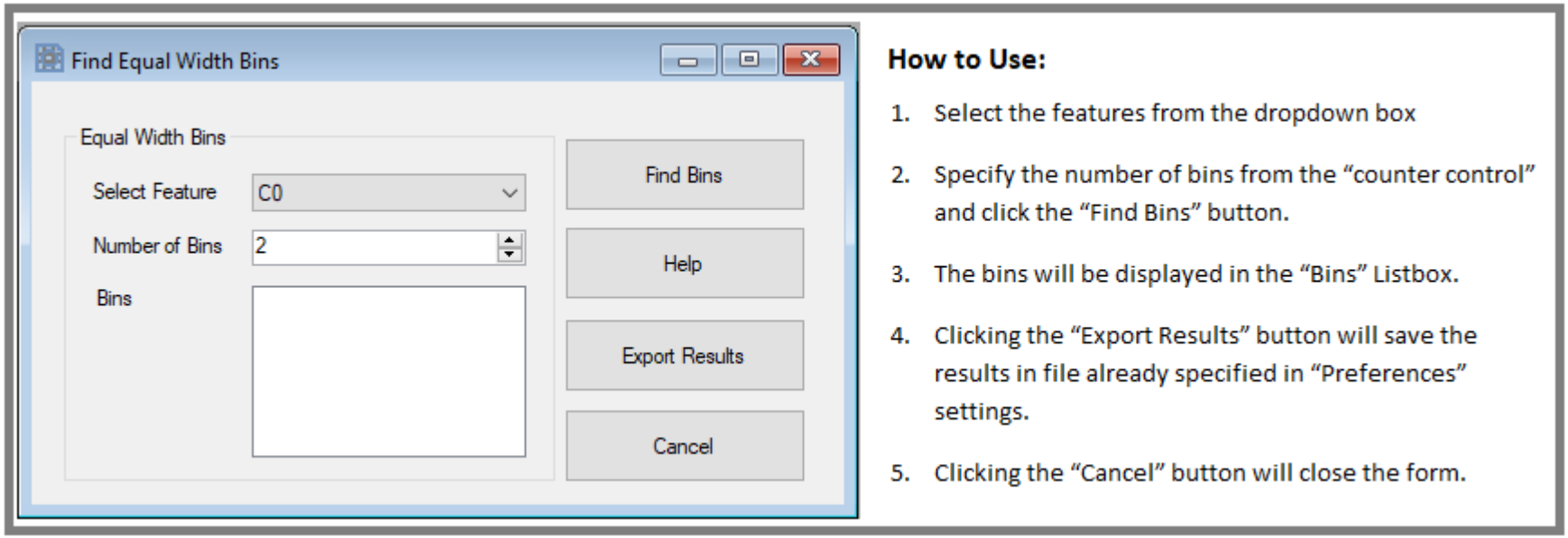
How to Use:

1. When the form opens, the first Listbox will contain all the features in which missing values will be removed using "Mean" operation. The second Listbox will contain all the features where missing values will be removed using "Mode" operation.
2. Clicking the "Remove Objects" button will replace missing values and dataset grid will be refreshed using updated values.
3. Clicking the cancel button will close the form.

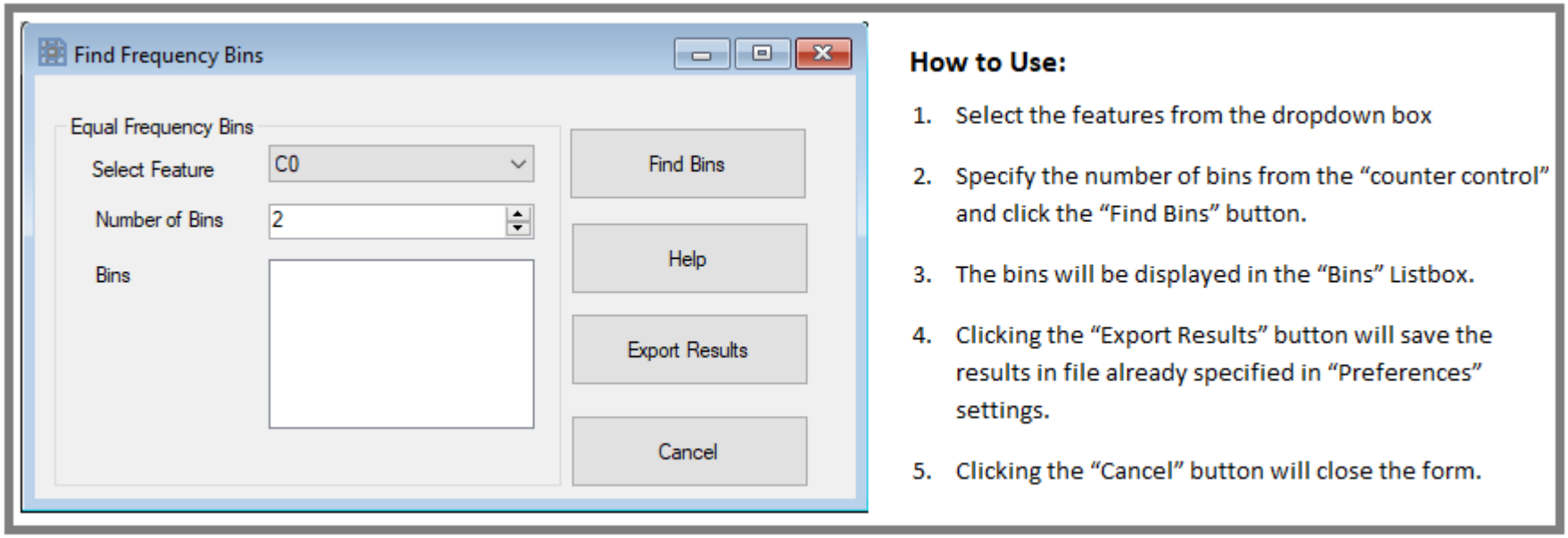
Form: Remove using Close Distance



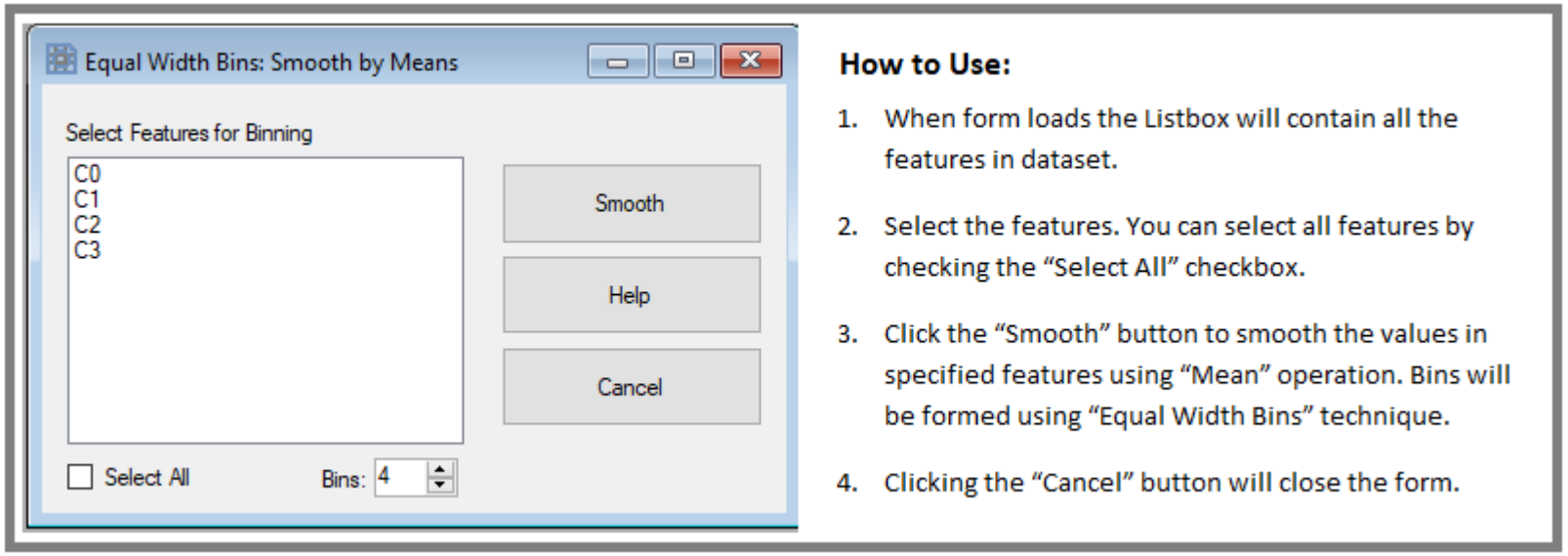
Form: Find Equal Width Bins



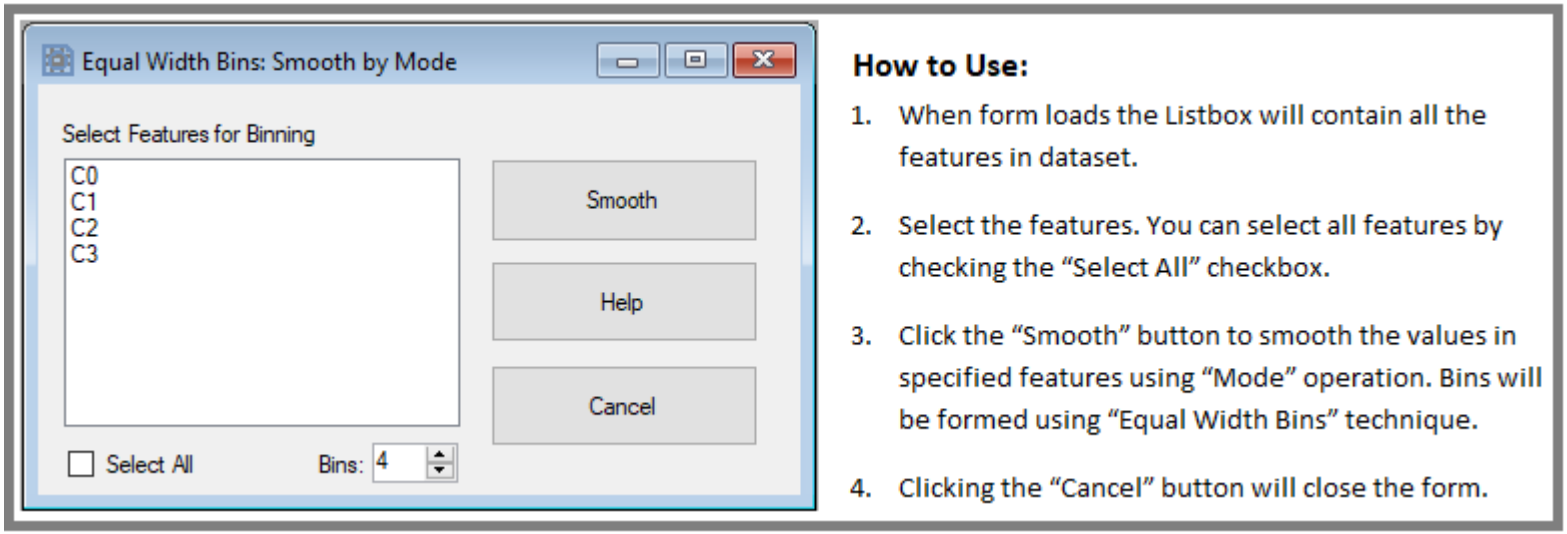
Form: Find Equal Frequency Bins



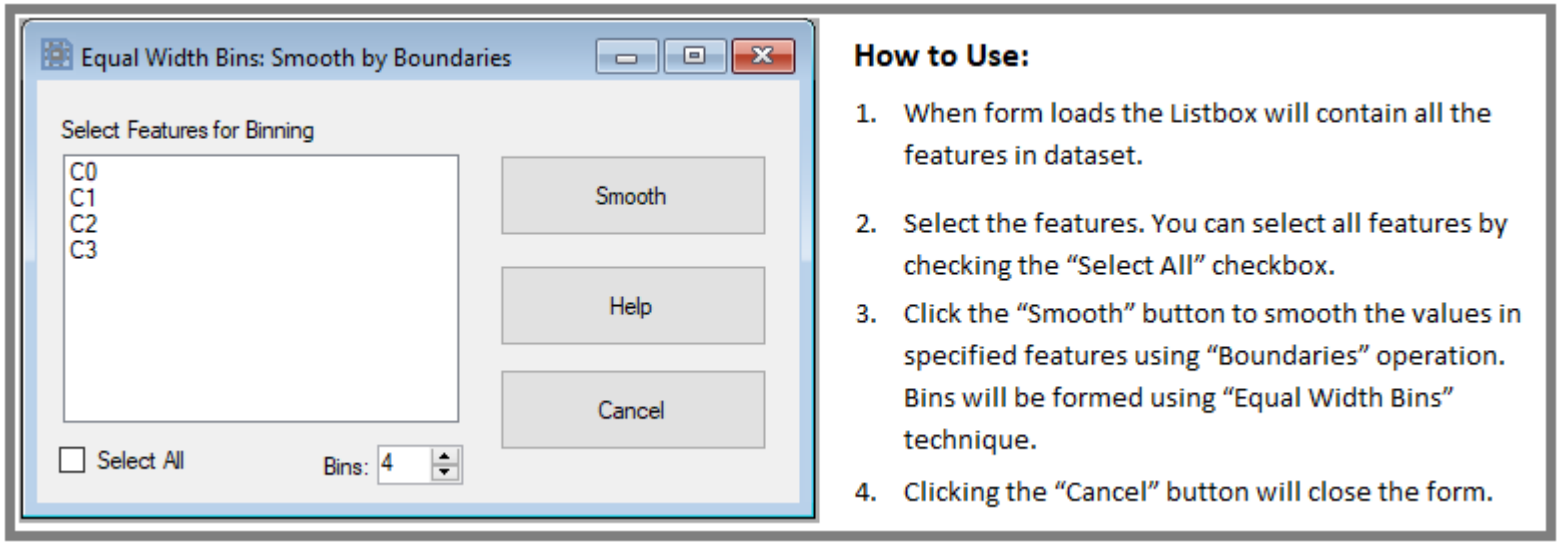
Form: Equal Width Bins: Smooth by Mean



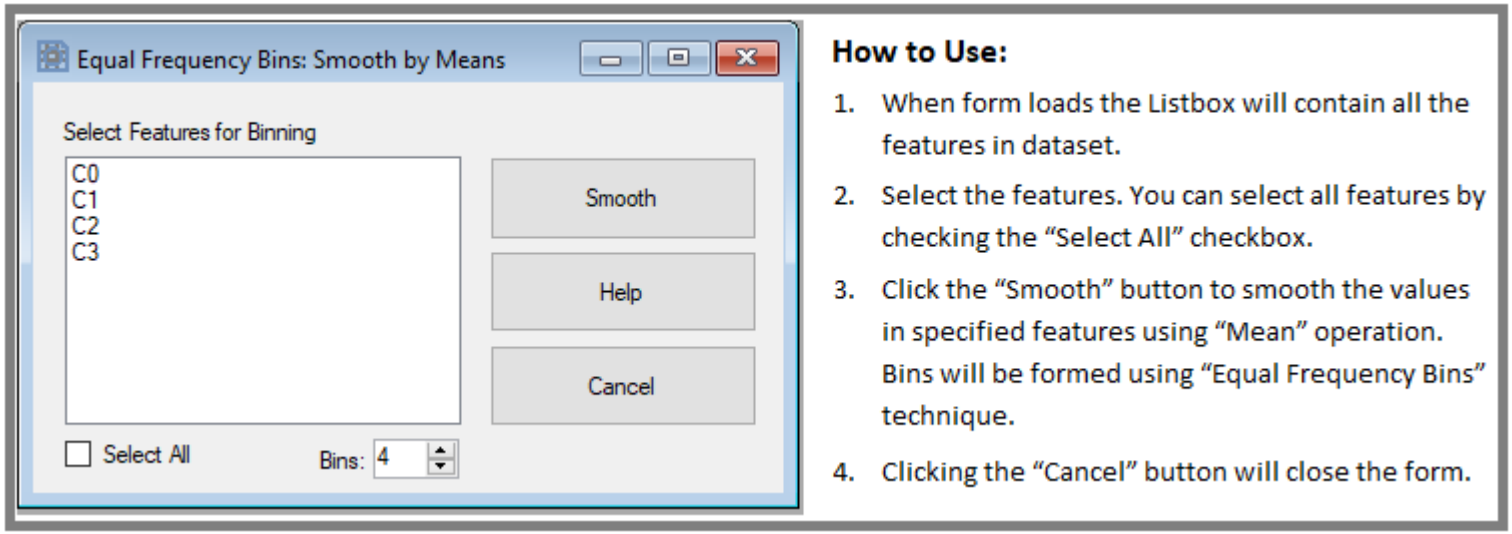
Form: Equal Width Bins: Smooth by Mode



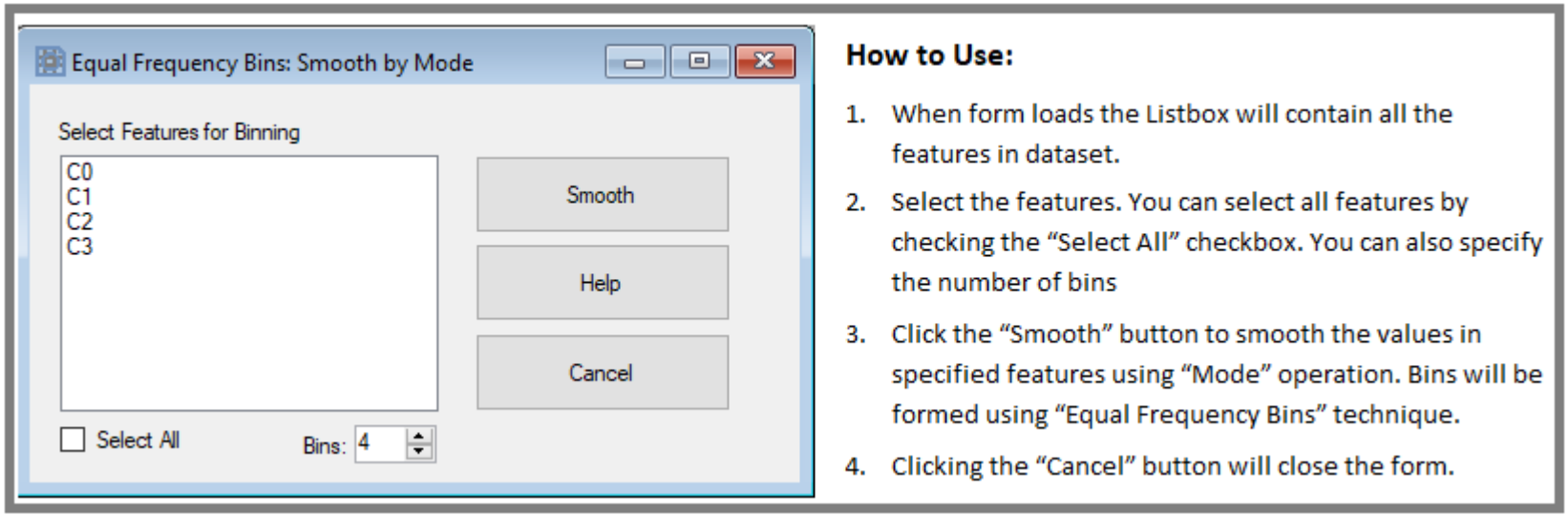
Form: Equal Width Bins: Smooth by Boundaries



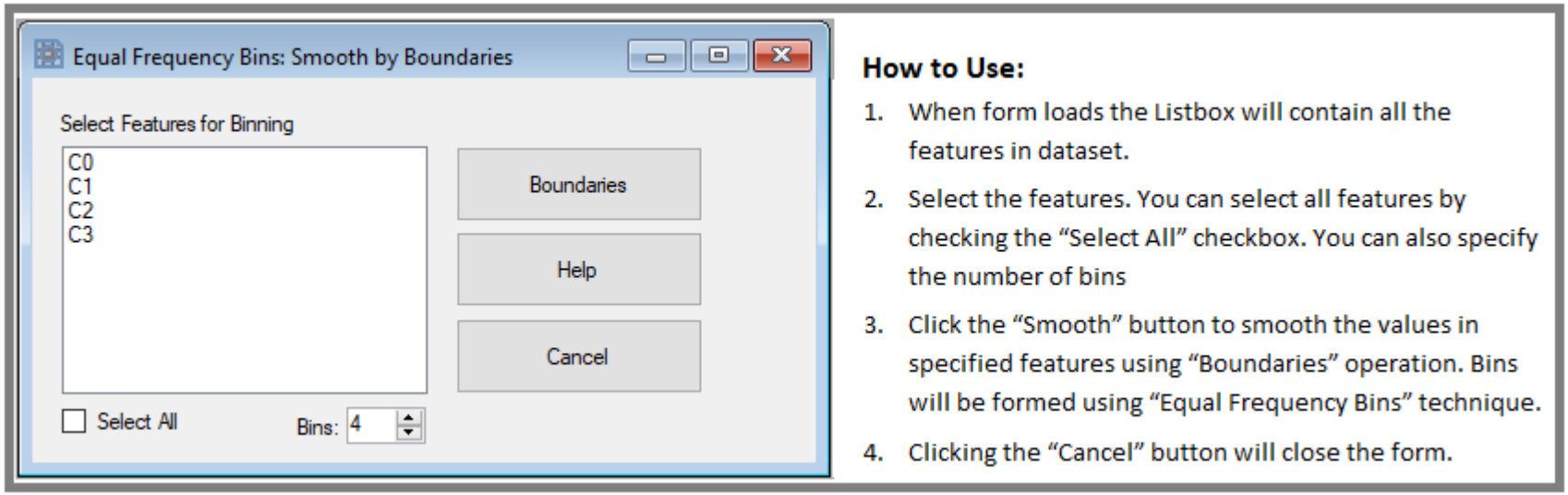
Form: Equal Frequency Bins: Smooth by Means



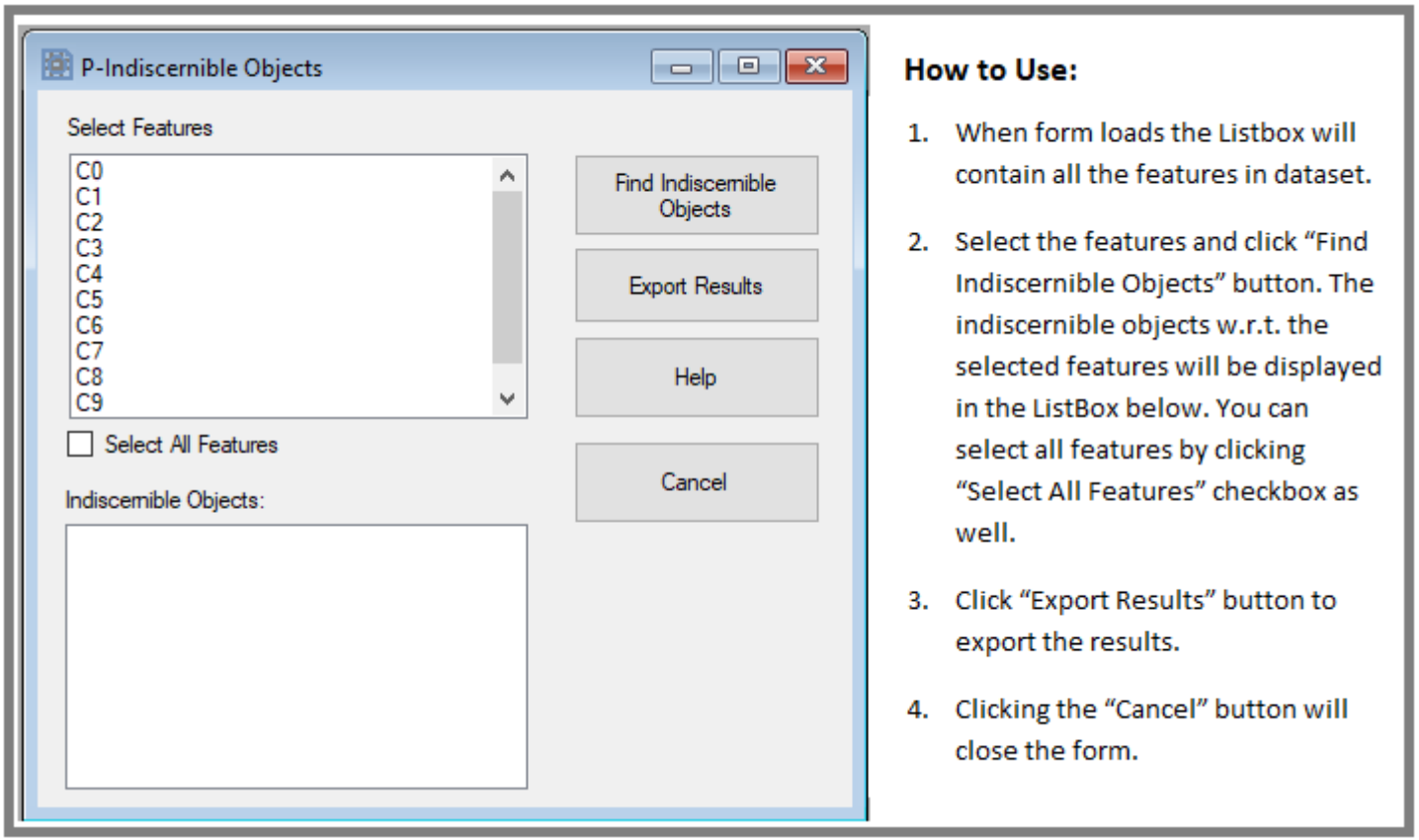
Form: Equal Frequency Bins: Smooth by Mode



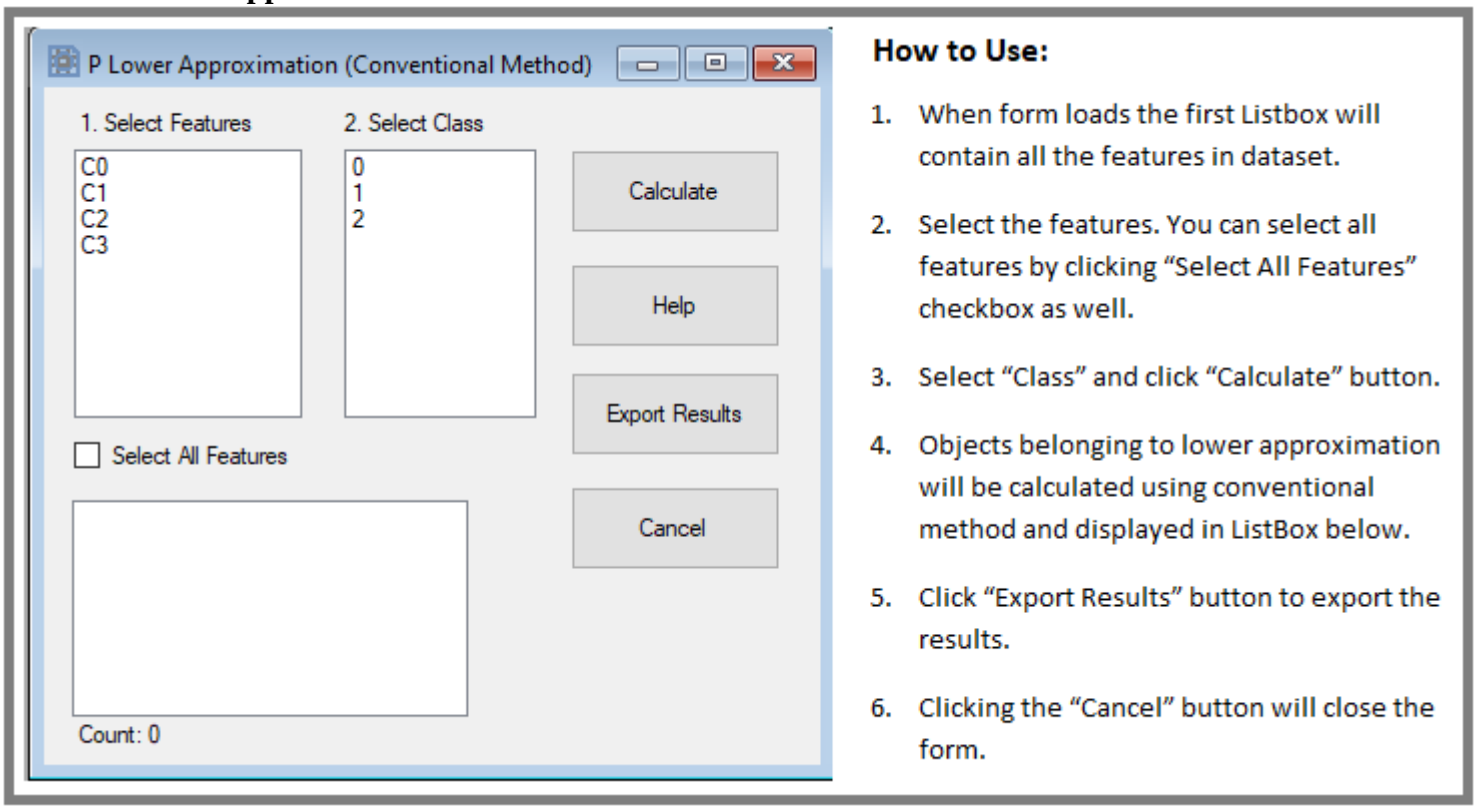
Form: Equal Frequency Bins: Smooth by Boundaries



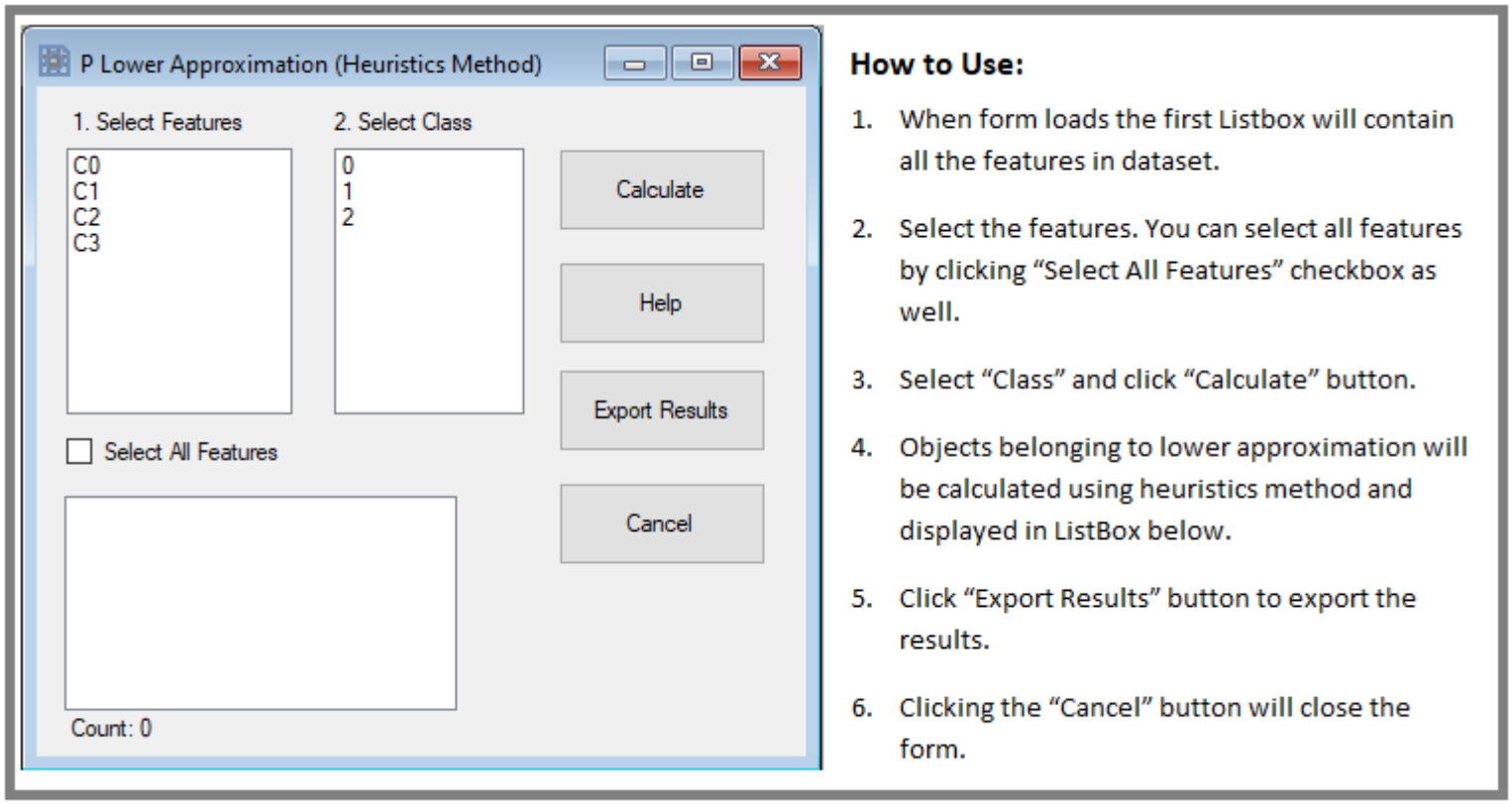
Form: P-Indiscernible Objects



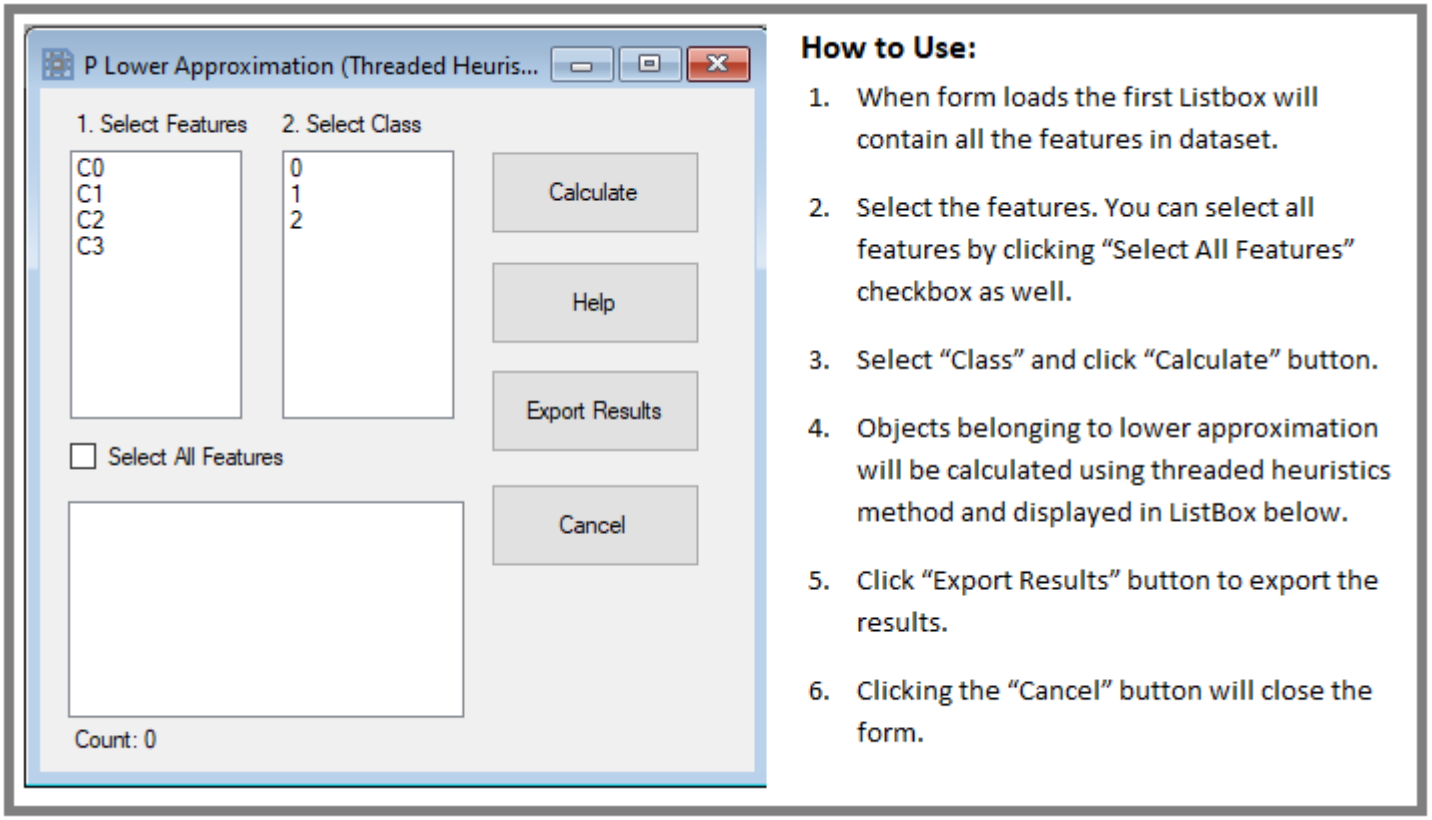
Form: P-Lower Approximation Coventional Method



Form: P-Lower Approximation Heuristics Method



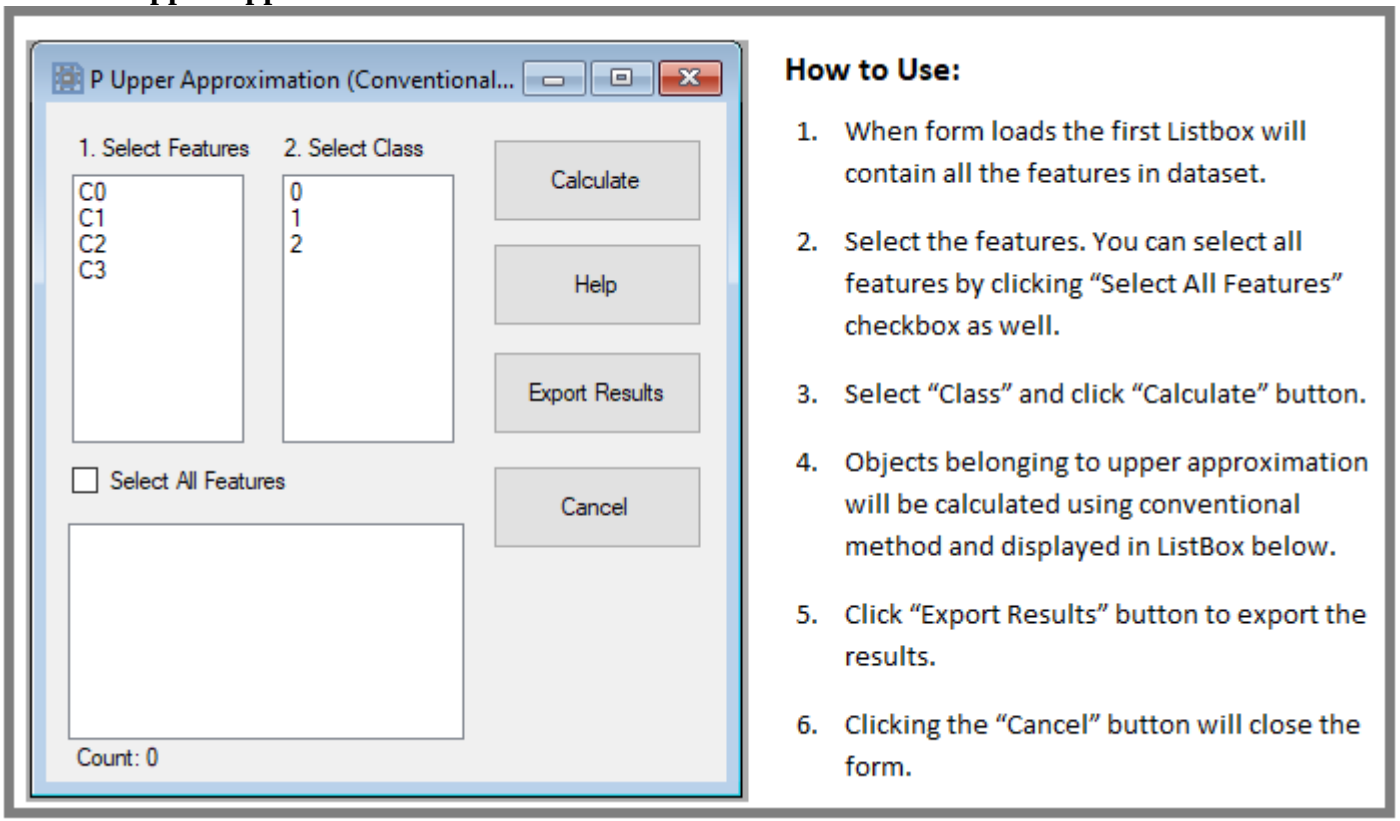
Form: P-Lower Approximation Threaded Heuristics Method



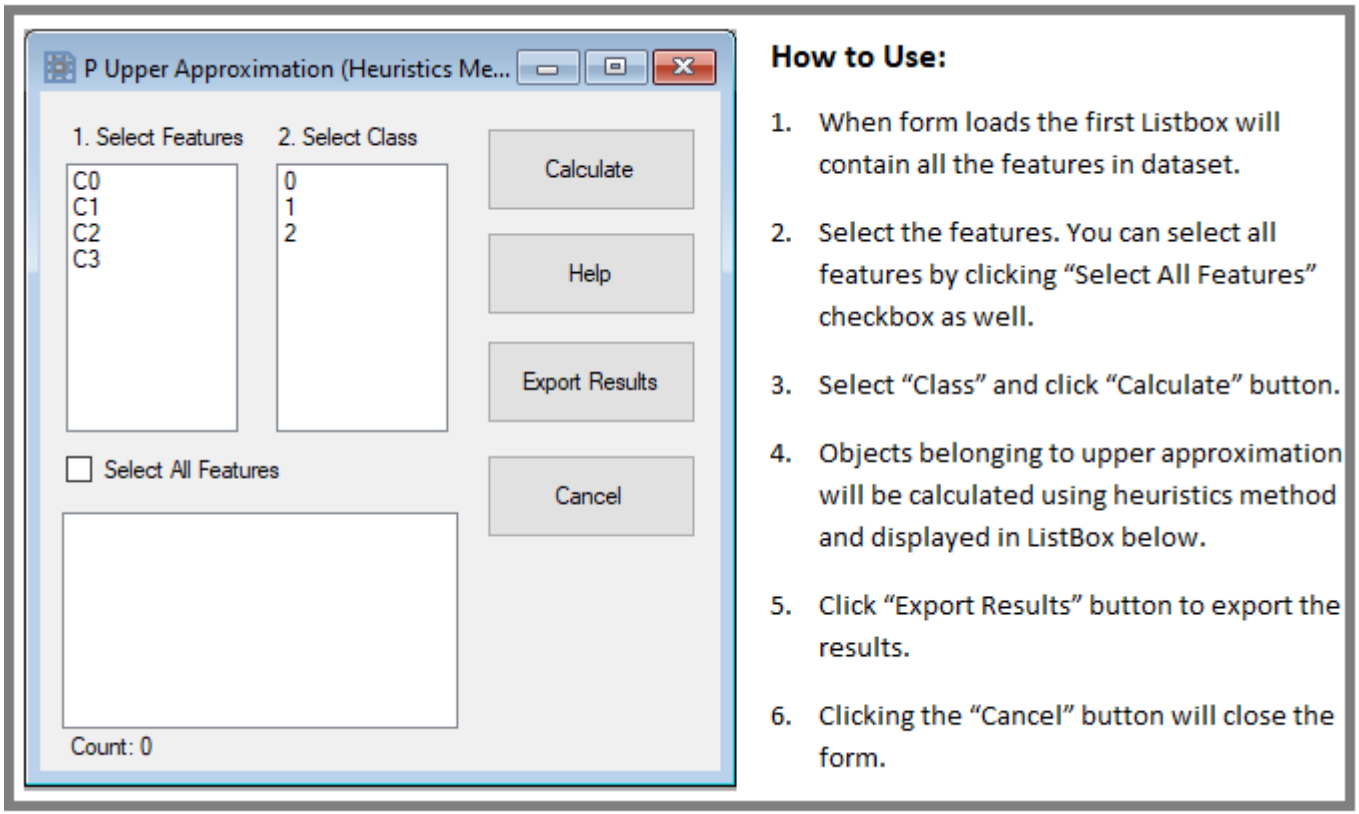
How to Use:

1. When form loads the first Listbox will contain all the features in dataset.
2. Select the features. You can select all features by clicking "Select All Features" checkbox as well.
3. Select "Class" and click "Calculate" button.
4. Objects belonging to lower approximation will be calculated using threaded heuristics method and displayed in ListBox below.
5. Click "Export Results" button to export the results.
6. Clicking the "Cancel" button will close the form.

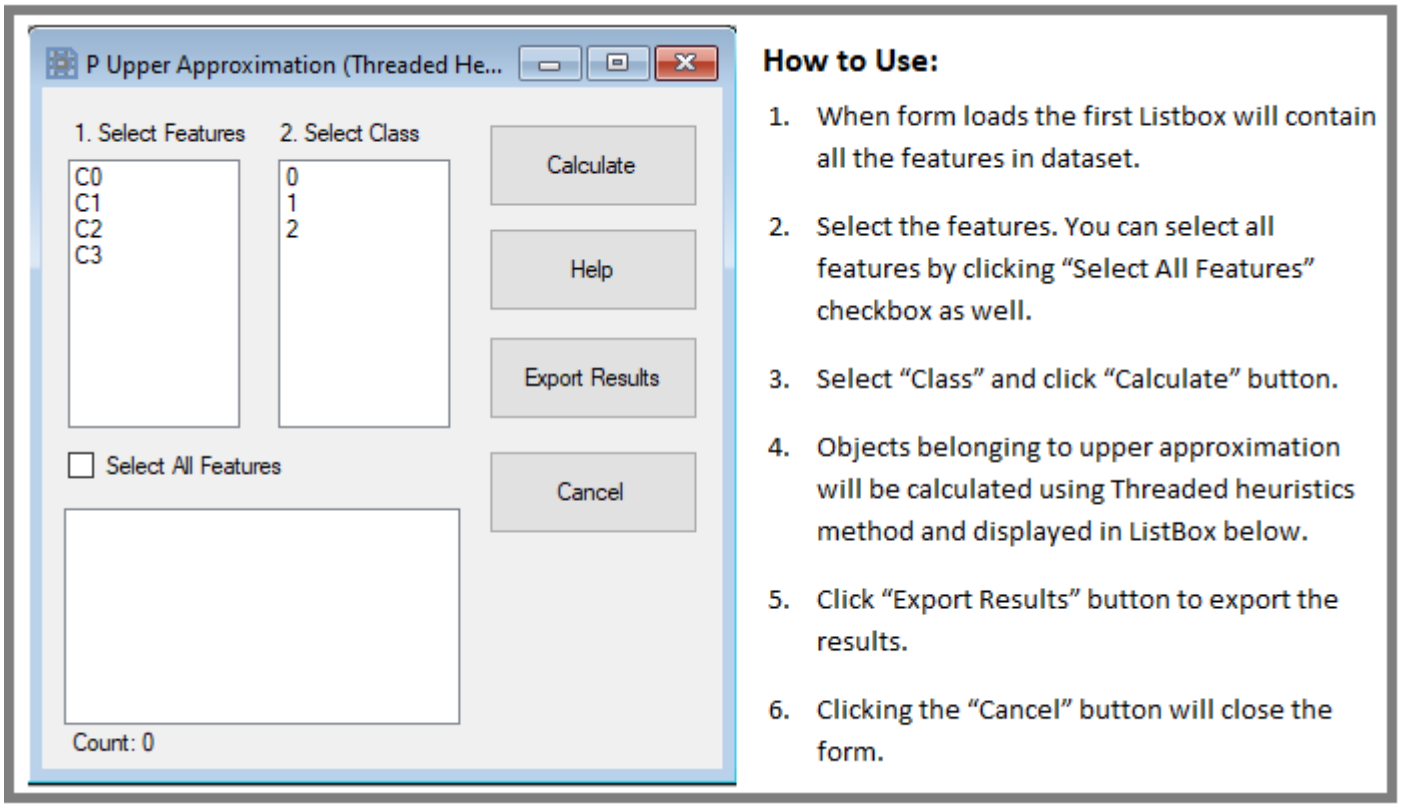
Form: P-Upper Approximation Coventional Method



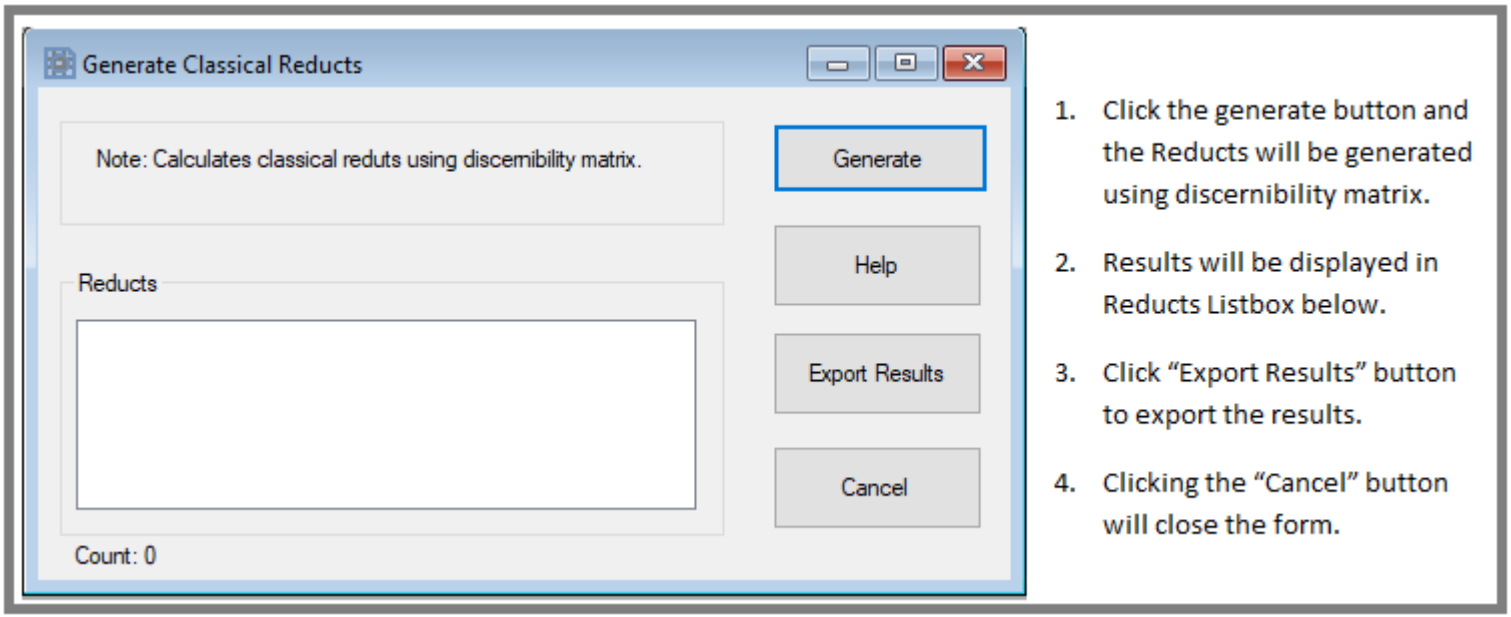
Form: P-Upper Approximation Heuristics Method



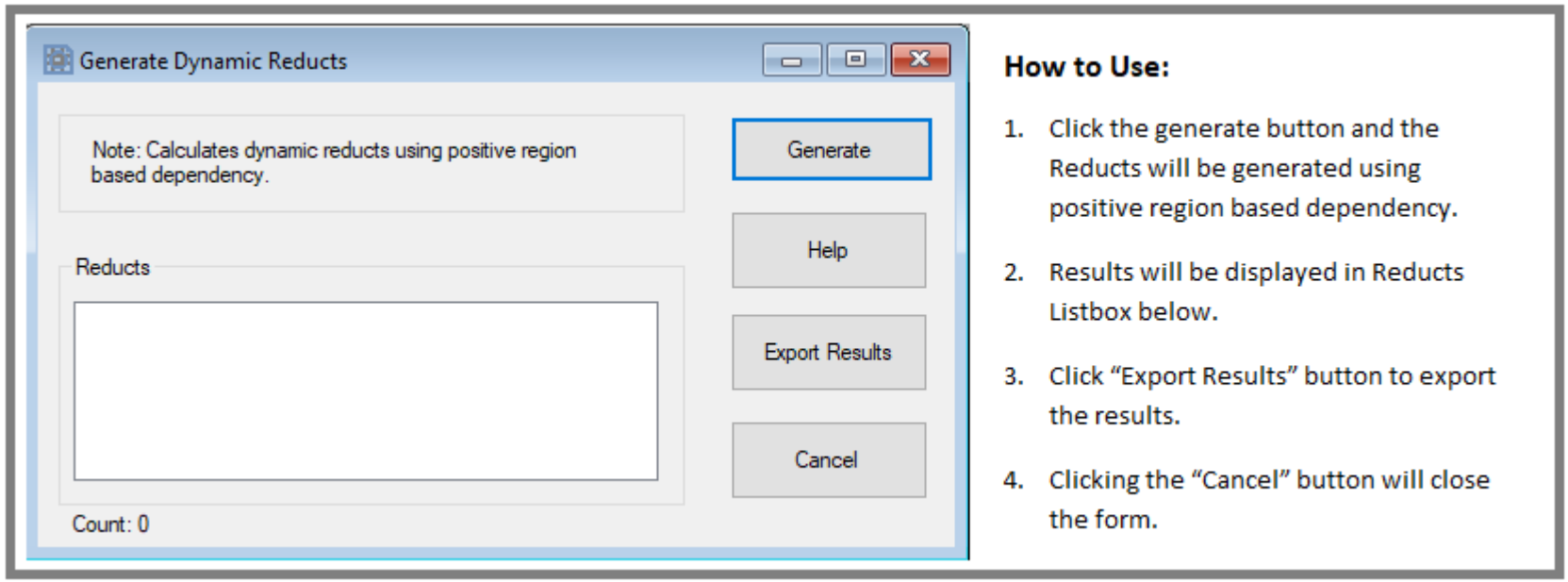
Form: P-Upper Approximation Threaded Heuristics Method



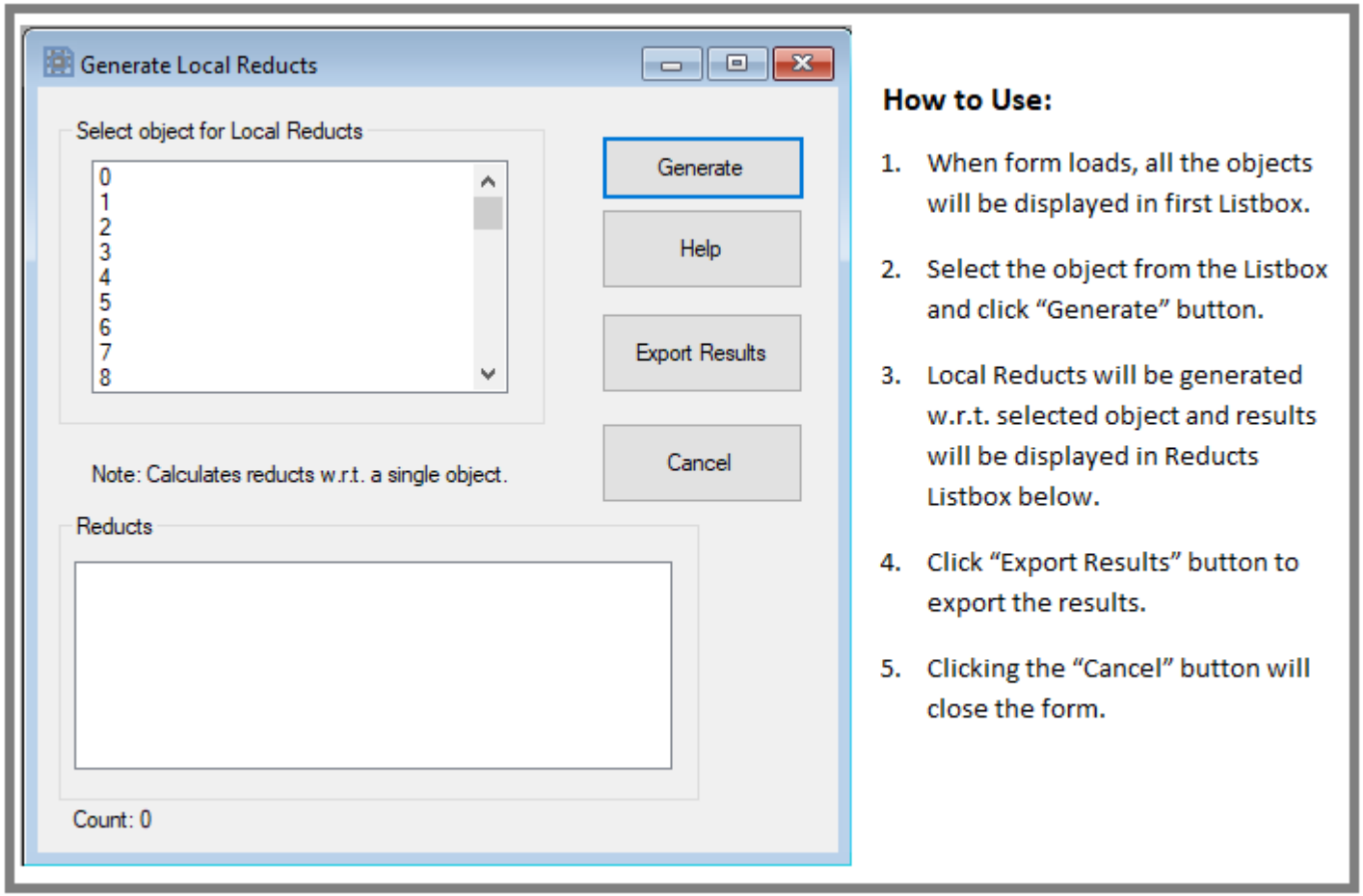
Form: Generate Classical Reducts



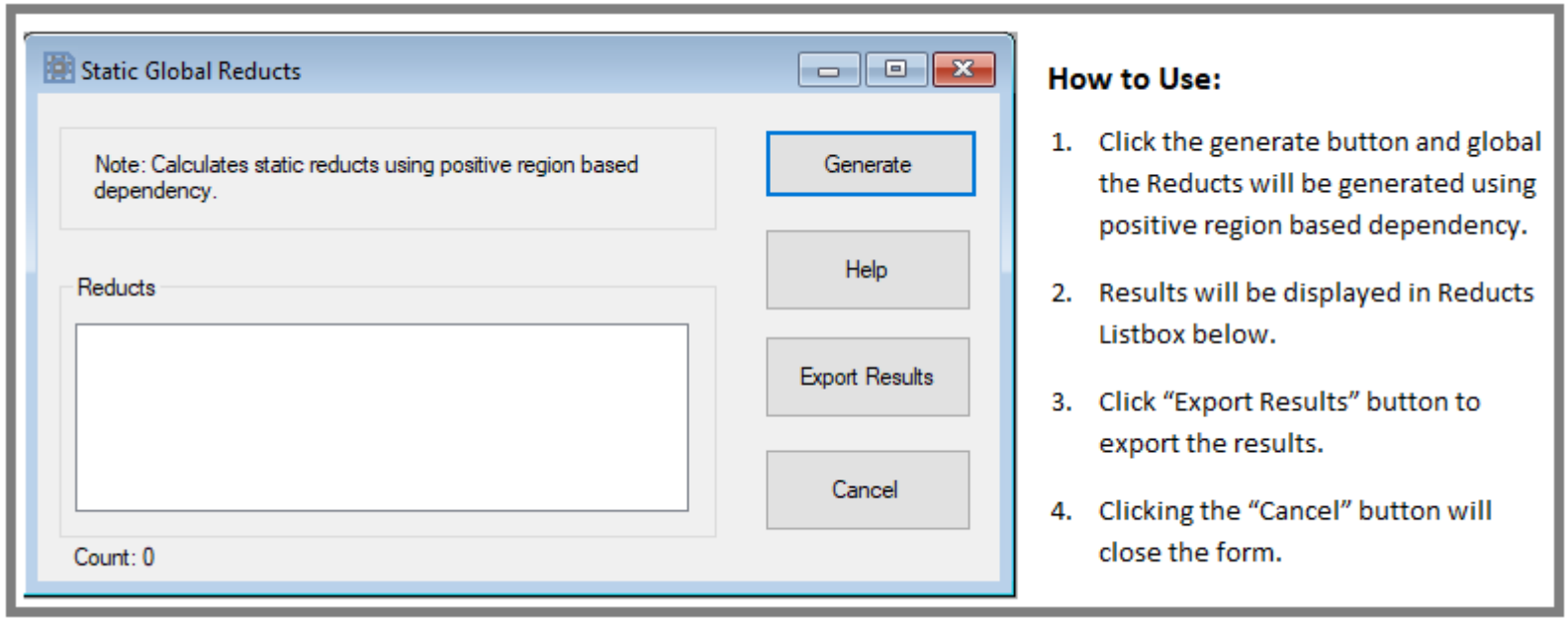
Form: Generate Dynamic Reducts



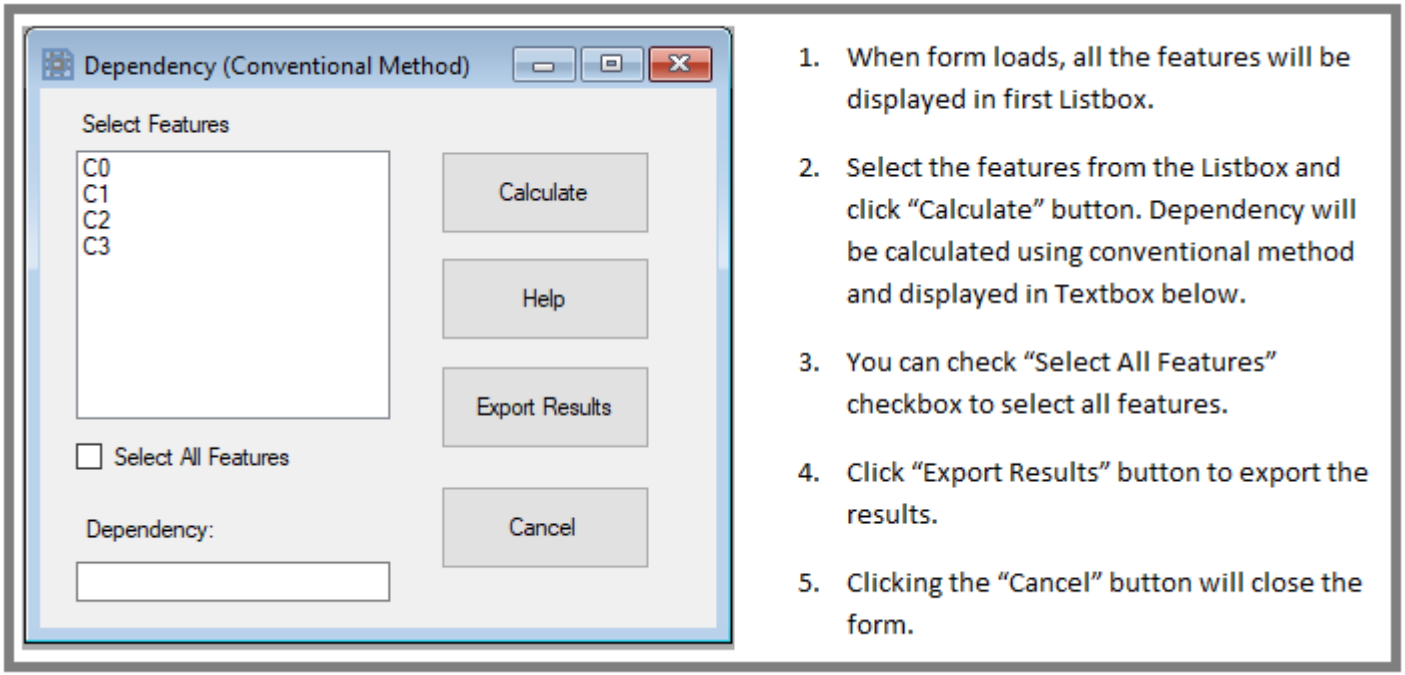
Form: Generate Local Reducts



Form: Static Global Reducts

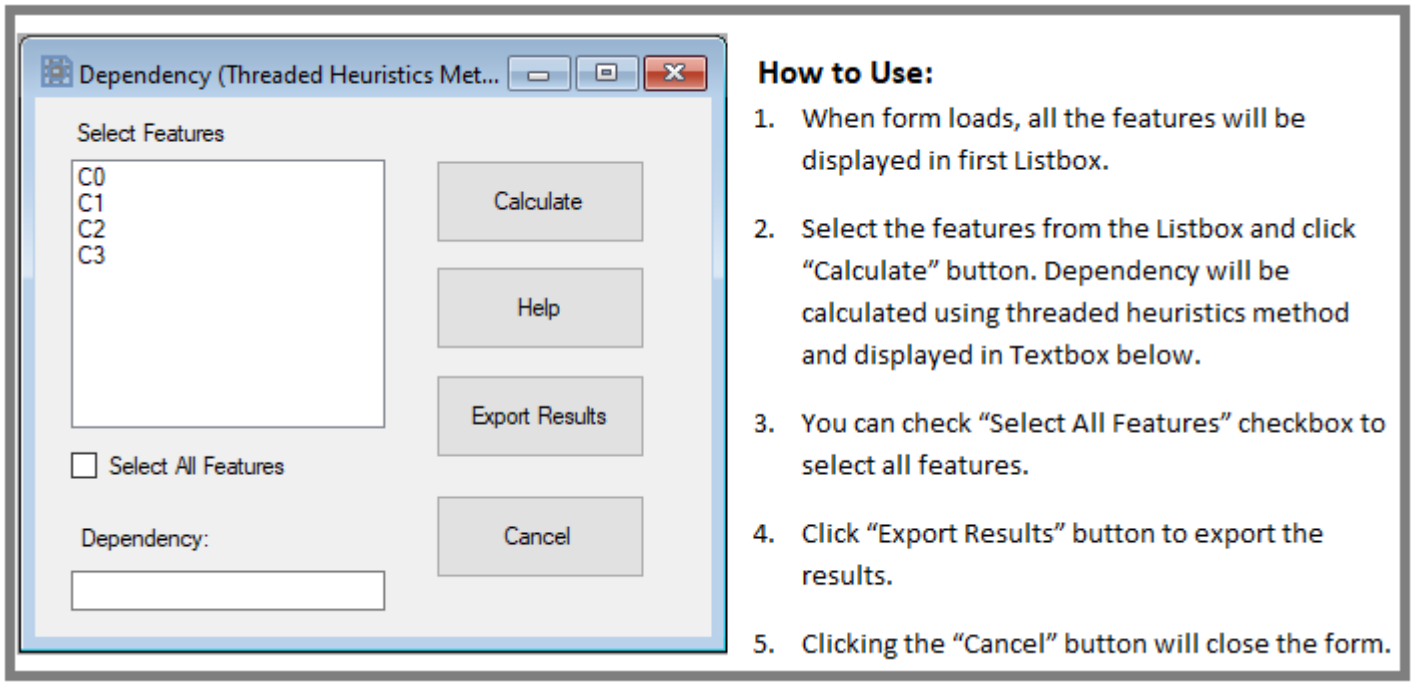


Form: Dependency (Conventional Method)

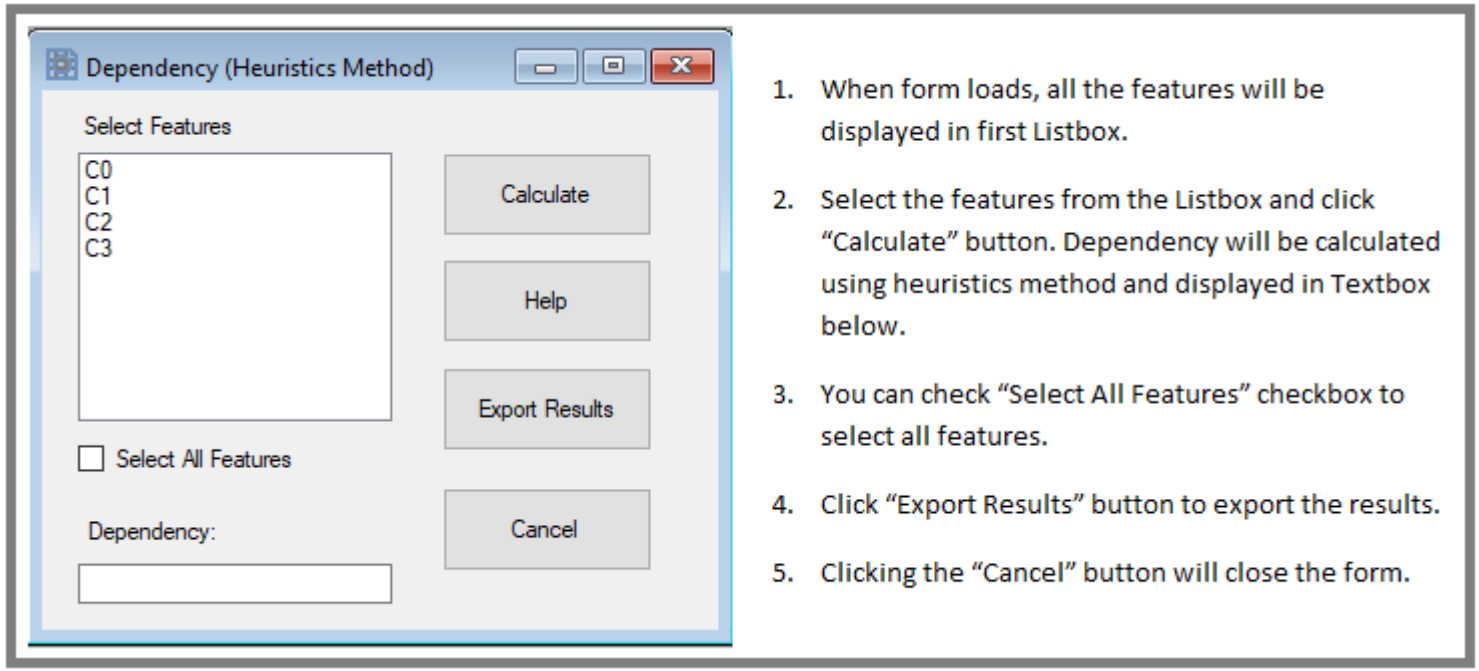


1. When form loads, all the features will be displayed in first Listbox.
2. Select the features from the Listbox and click "Calculate" button. Dependency will be calculated using conventional method and displayed in Textbox below.
3. You can check "Select All Features" checkbox to select all features.
4. Click "Export Results" button to export the results.
5. Clicking the "Cancel" button will close the form.

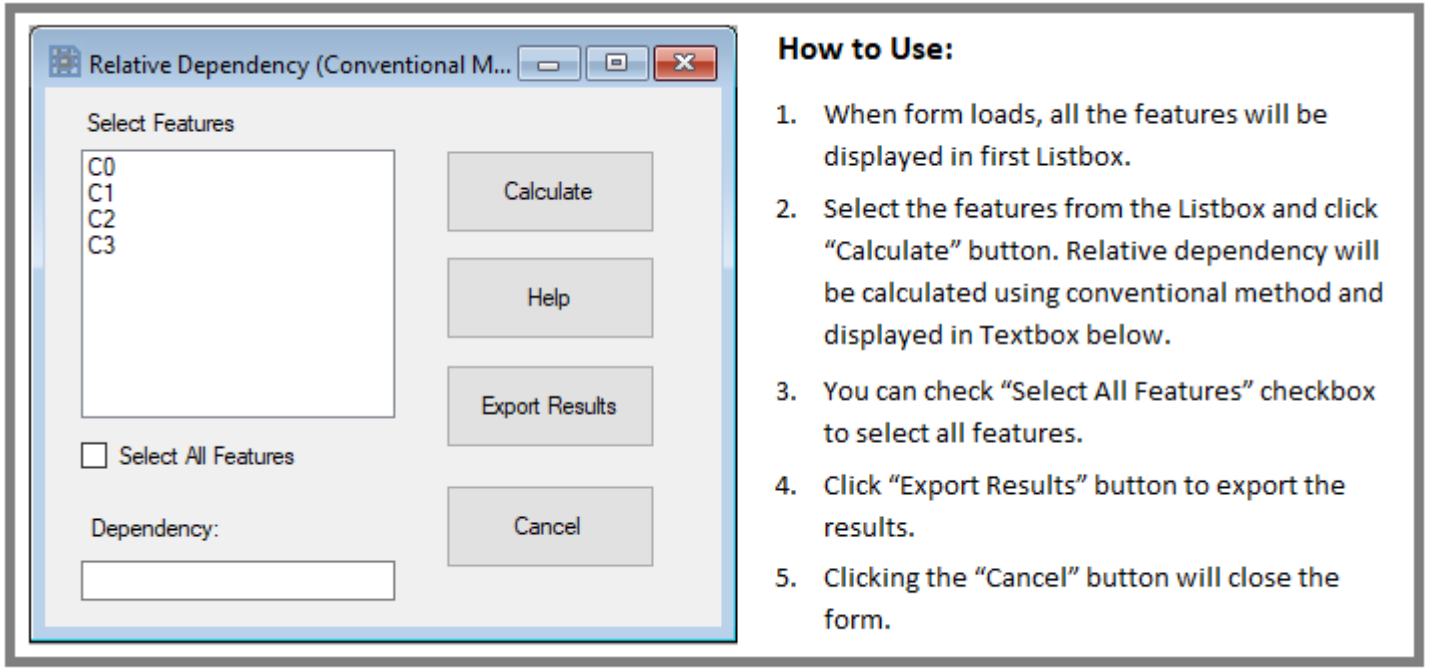
Form: Dependency (Threaded Heuristics Method)



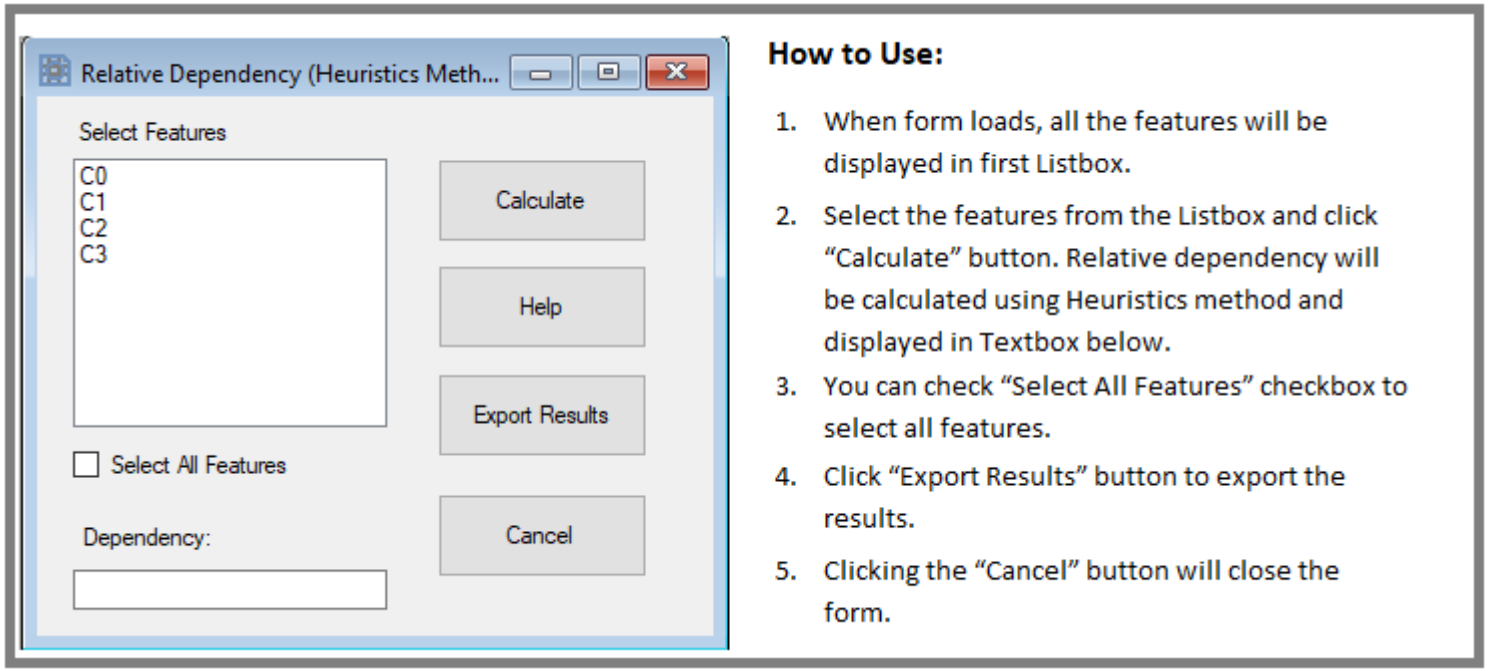
Form: Dependency (Heuristics Method)



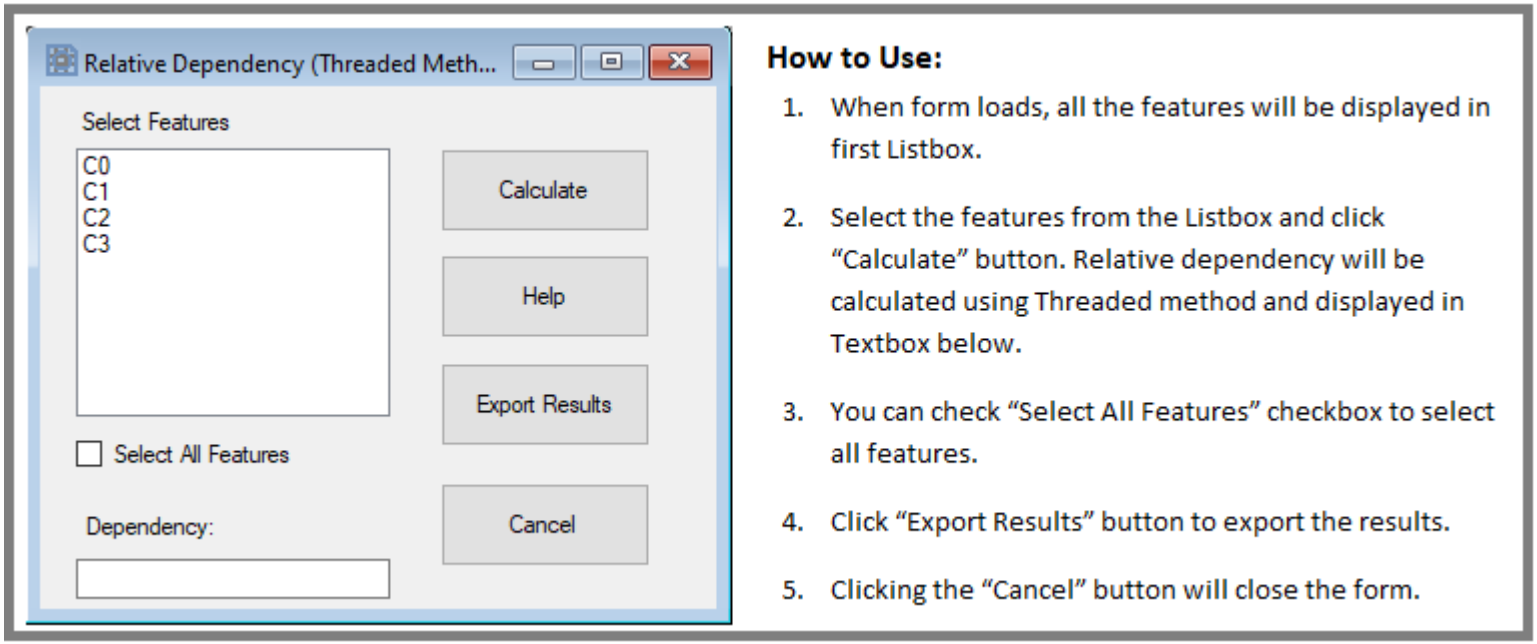
Form: Relative Dependency (Conventional Method)



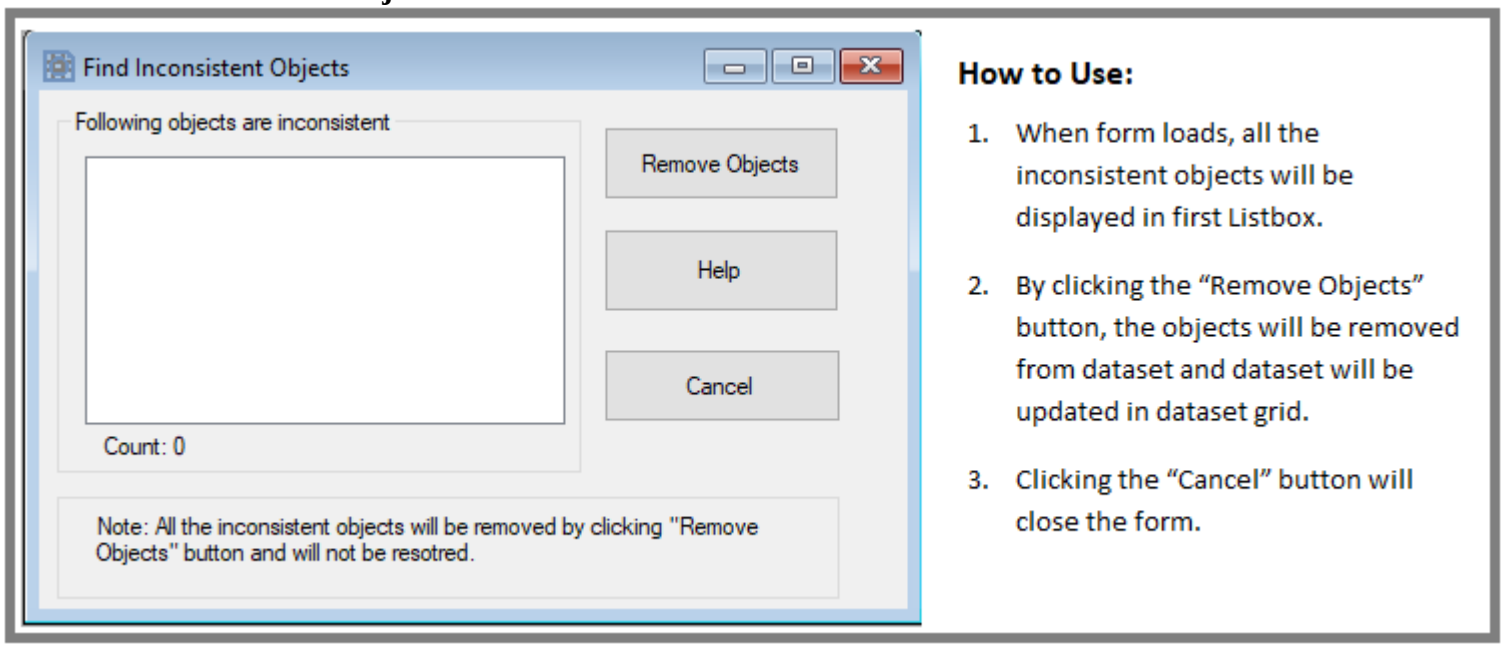
Form: Relative Dependency (Heuristics Method)



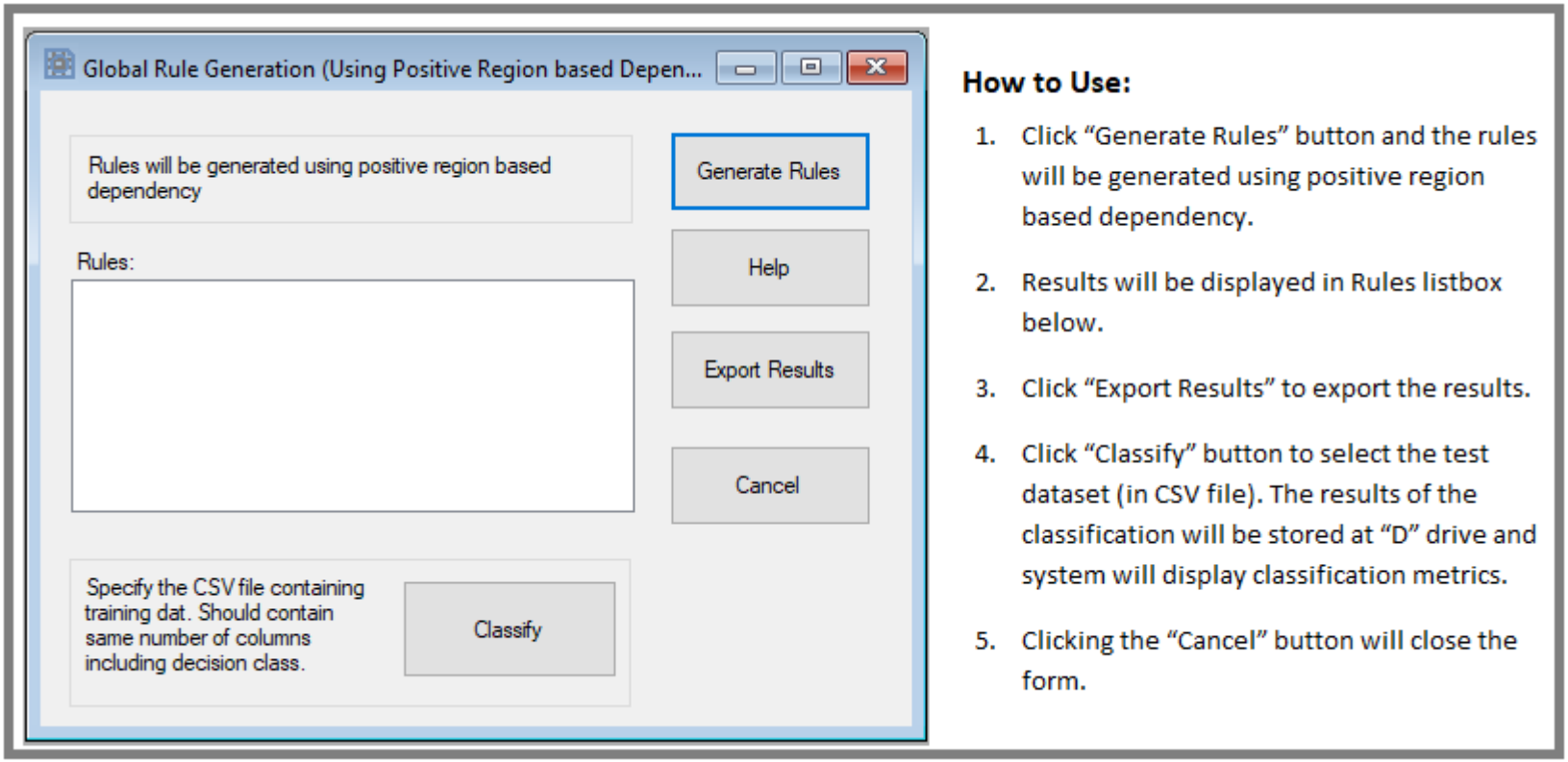
Form: Relative Dependency (Threaded Method)



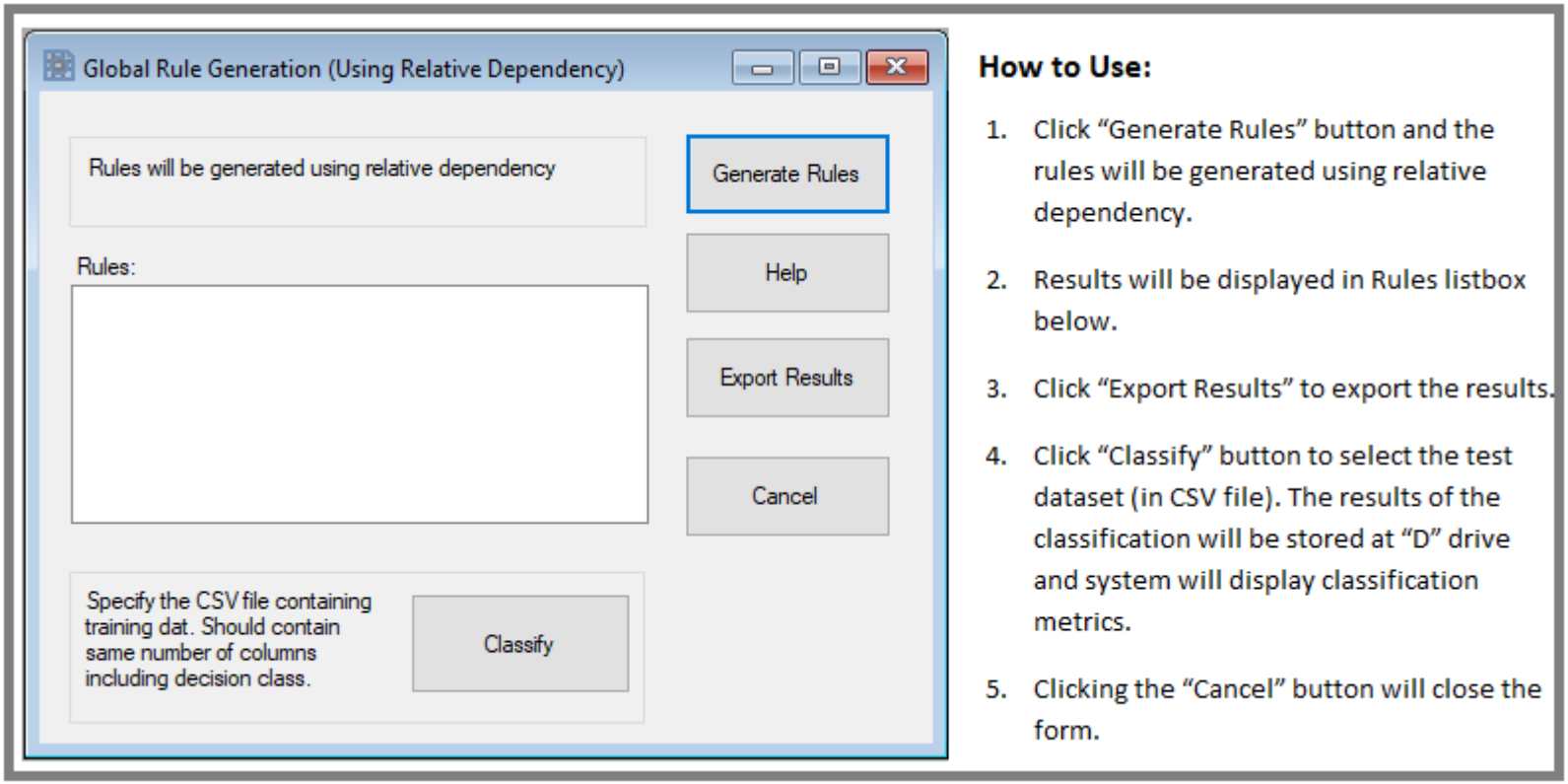
Form: Find Inconsistent Objects



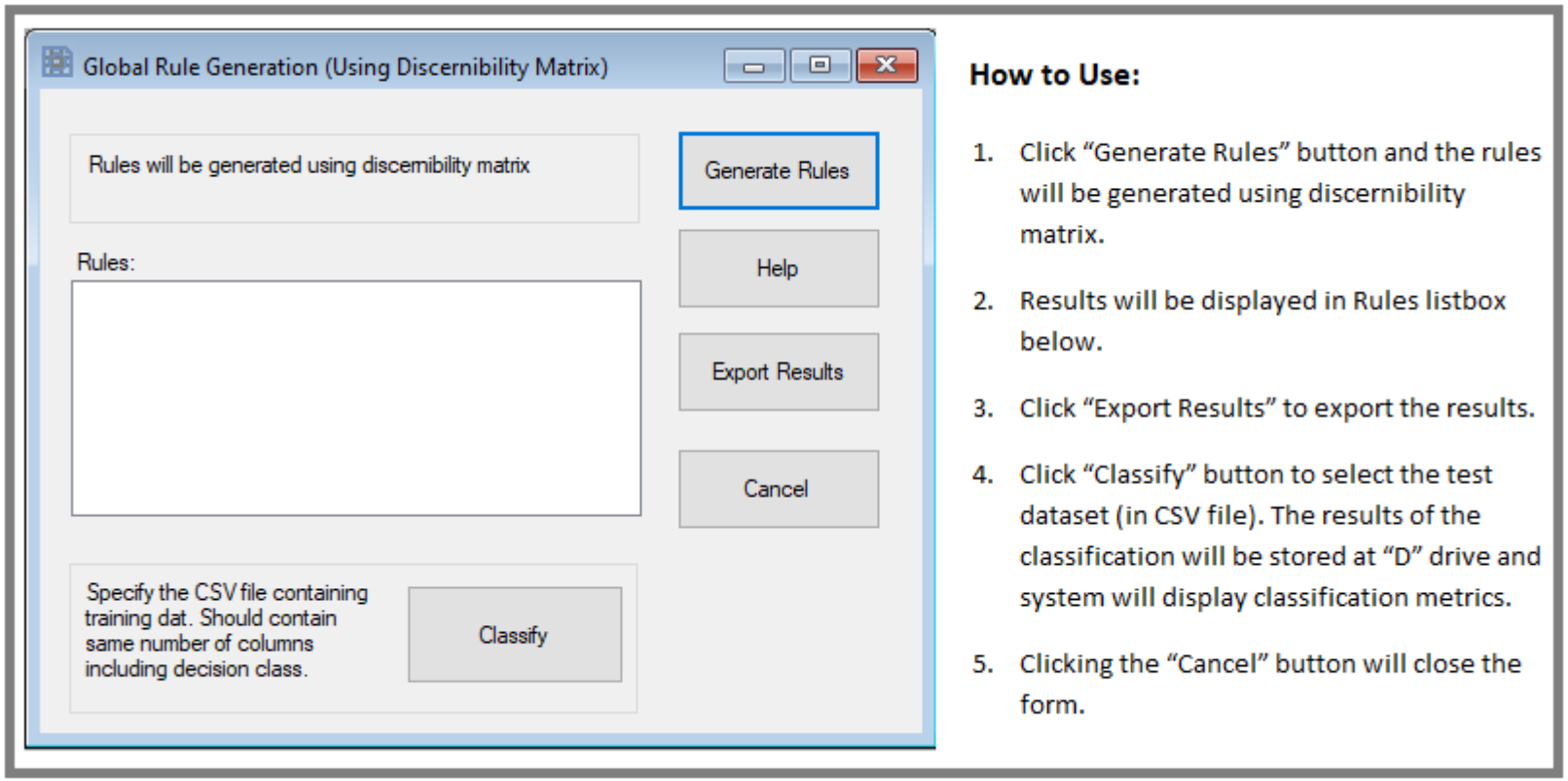
Form: Global Rule Generation (Using Positive Region based Dependency)



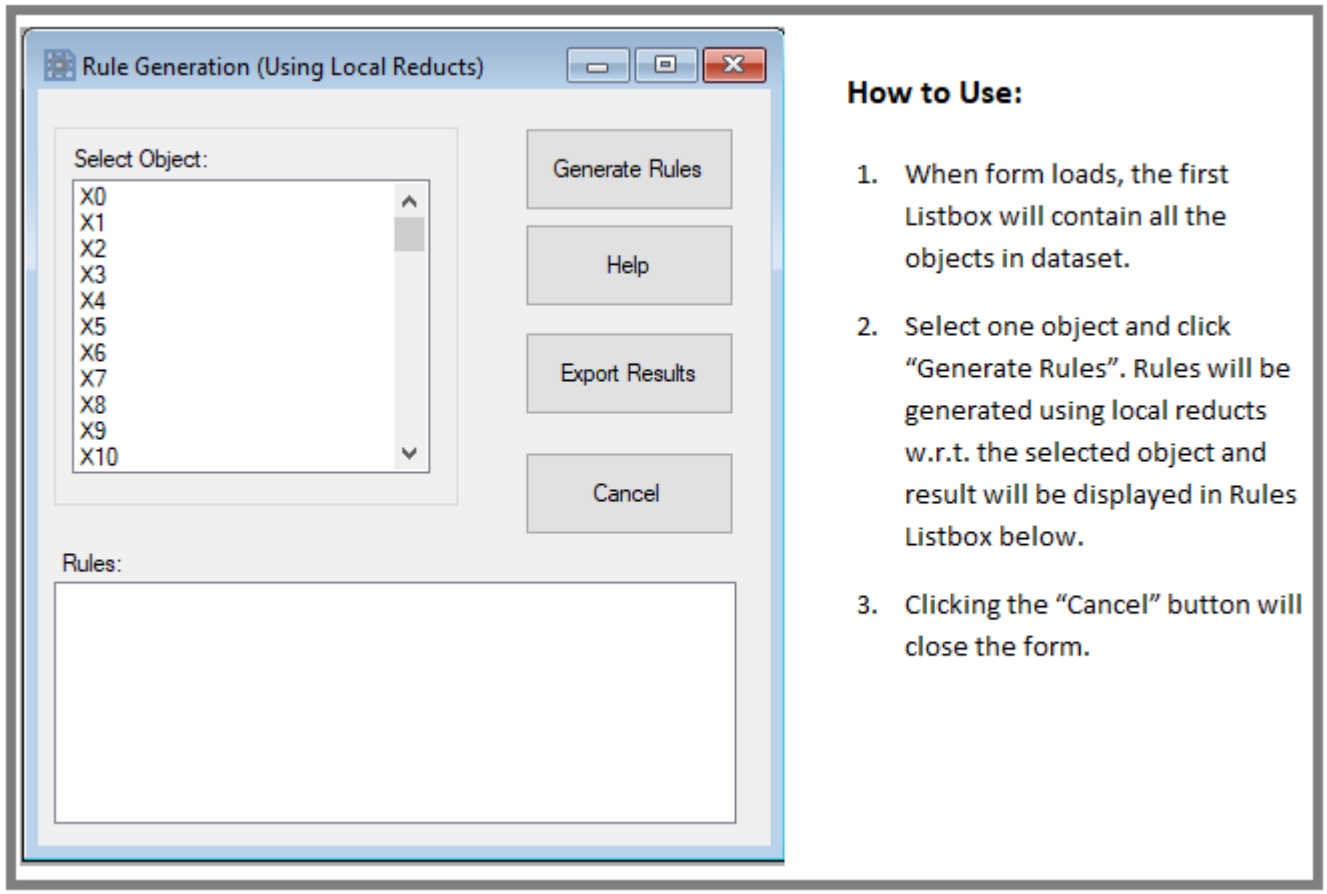
Form: Global Rule Generation (Using Relative Dependency)



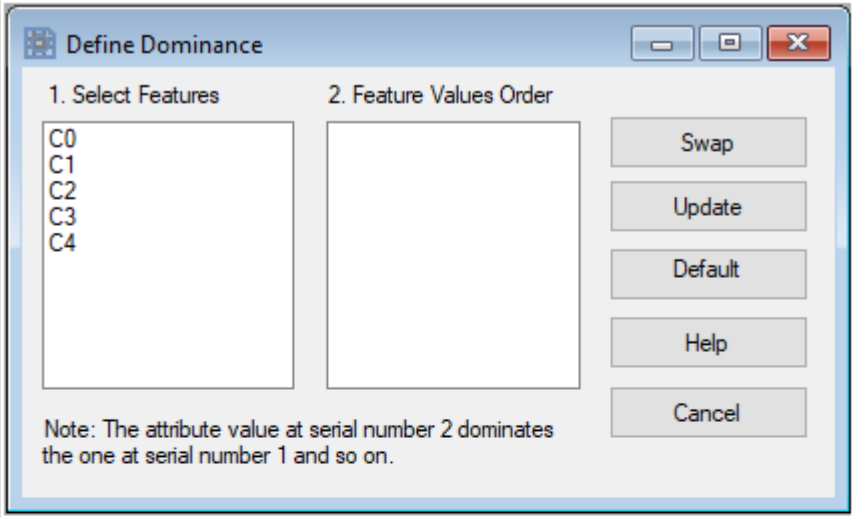
Form: Global Rule Generation (Using Discernibility Matrix)



Form: Rule Generation (Using Local Reducts)



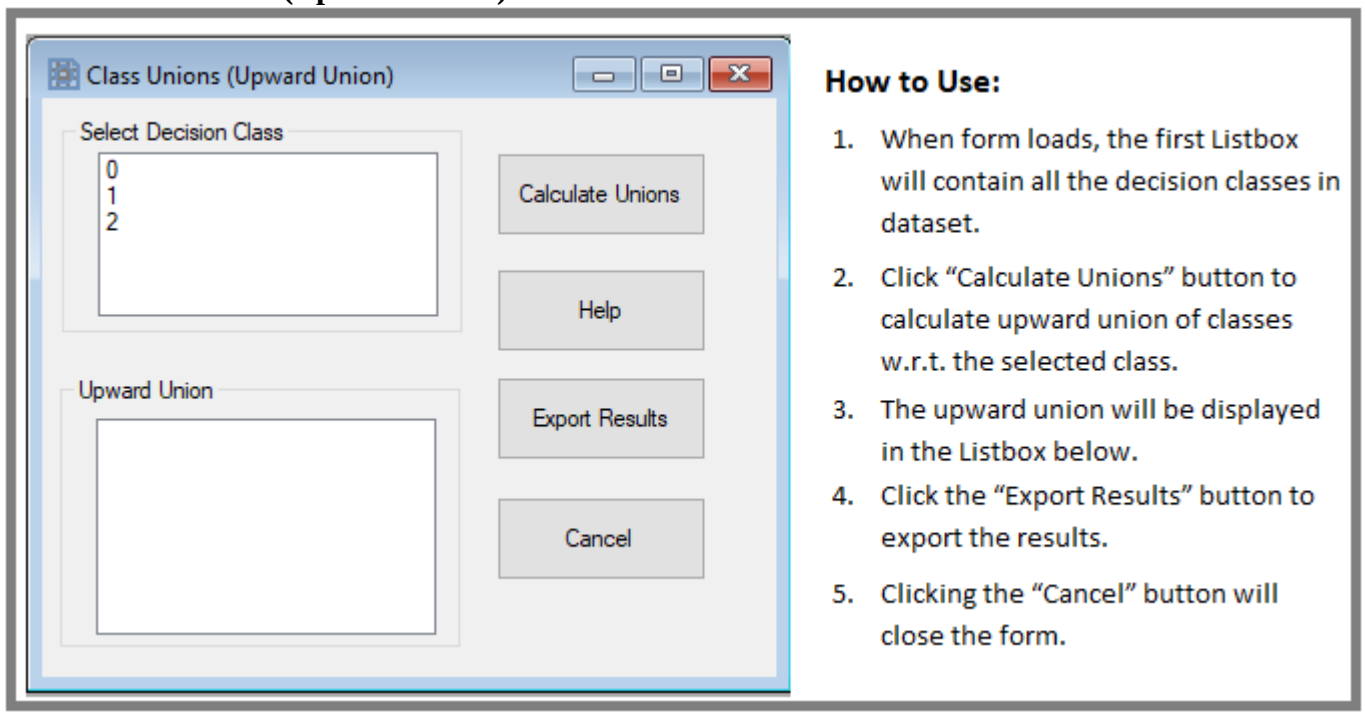
Form: Define Dominance



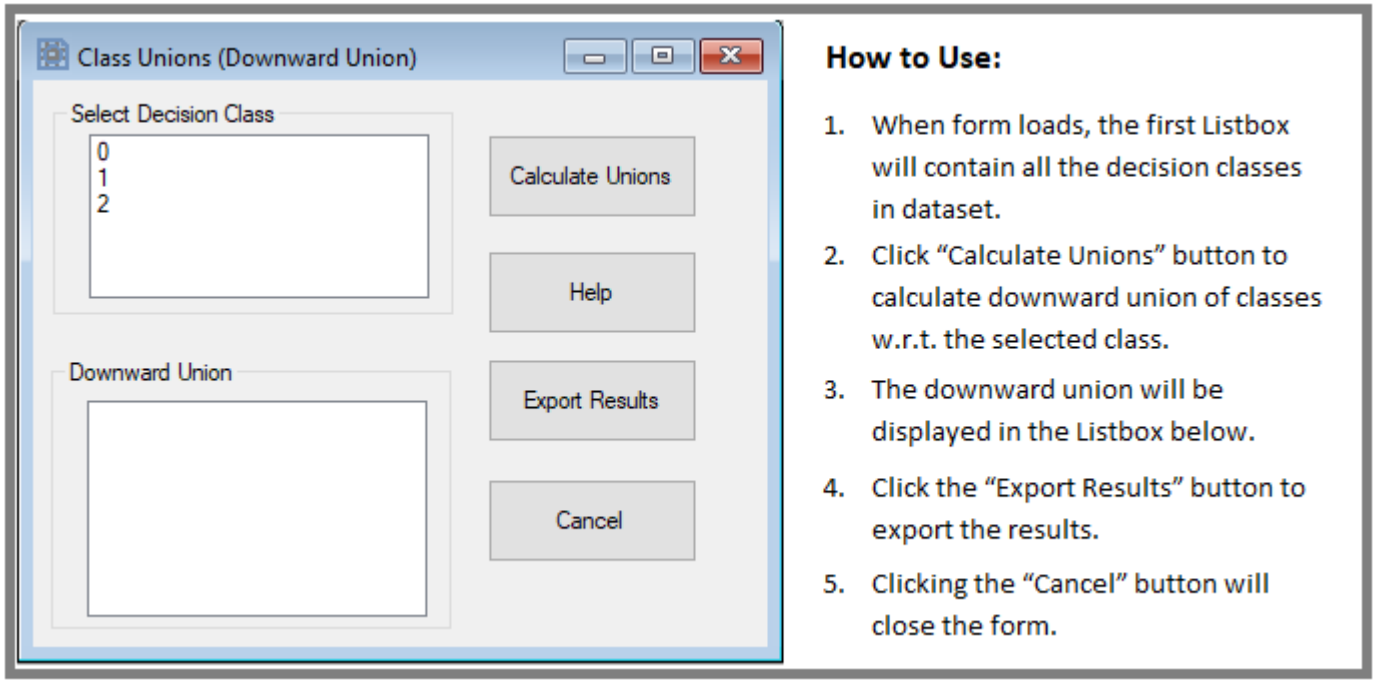
How to Use:

1. When form loads, the first Listbox will contain all the features in dataset.
2. Select one feature and its values will be displayed in "Feature Values Order" Listbox.
3. To sort a value, select that value and click the "Swap" button. The value will be swapped with the value above it. Click the "Update" button and the updated sort order will be saved. Clicking the "Default" button will sort the values in default order.
4. Clicking the "Cancel" button will close the form.

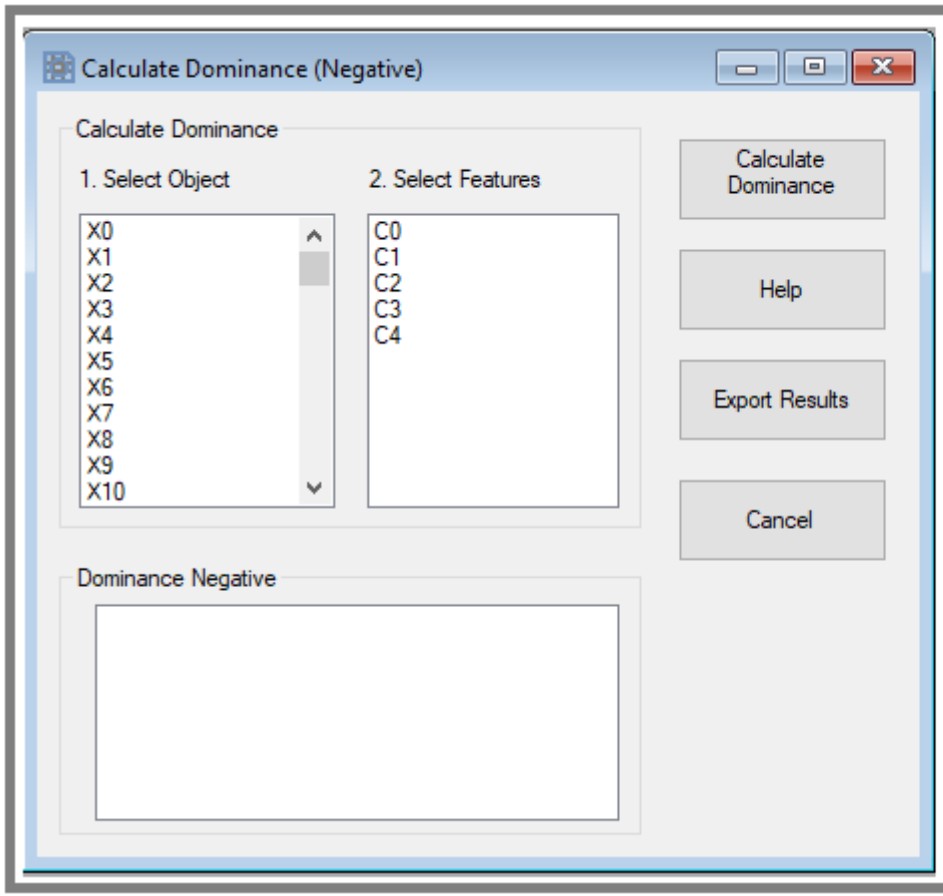
Form: Class Unions (Upward Union)



Form: Class Unions (Downward Union)



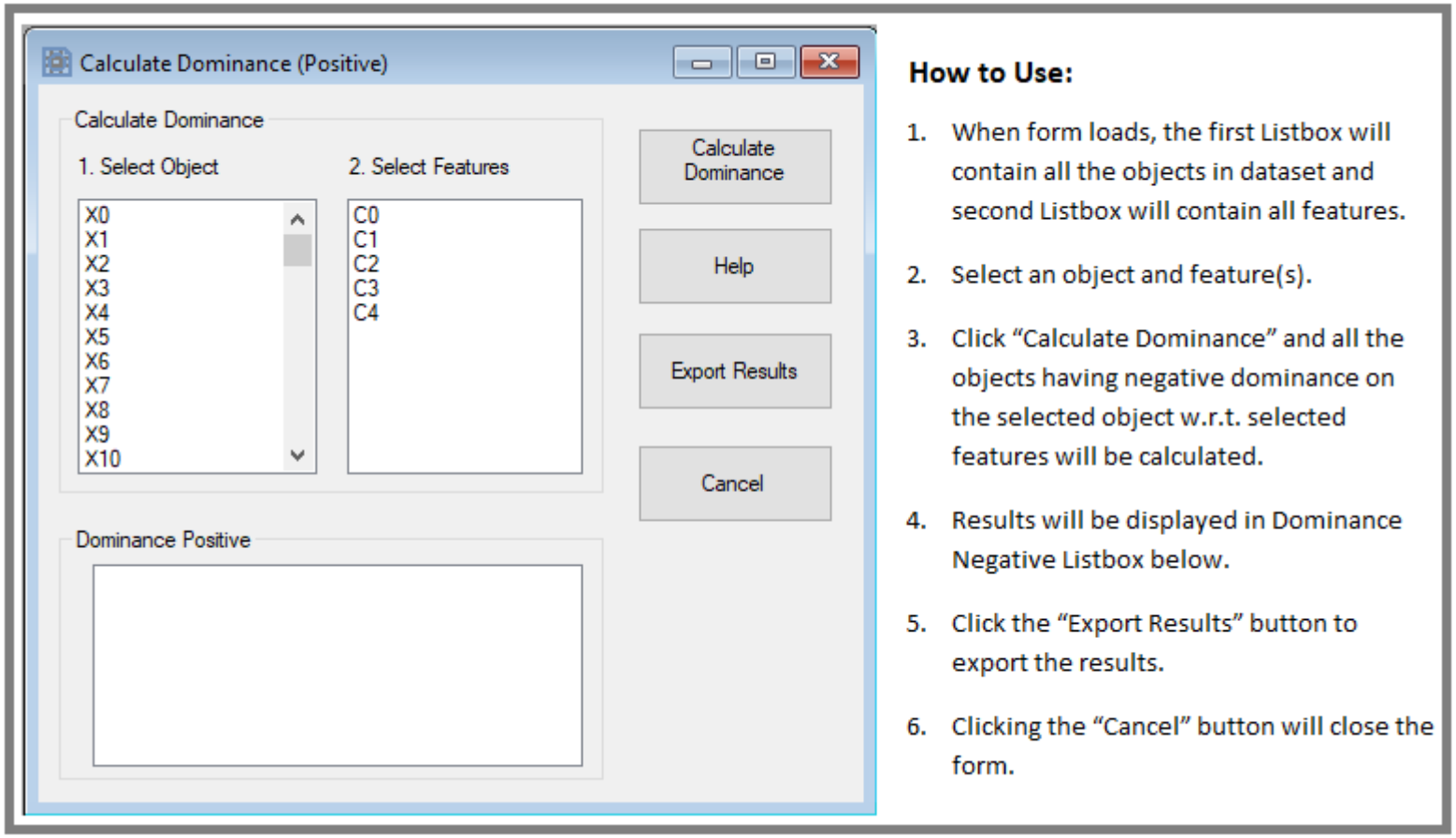
Form: Calculate Dominance (Negative)



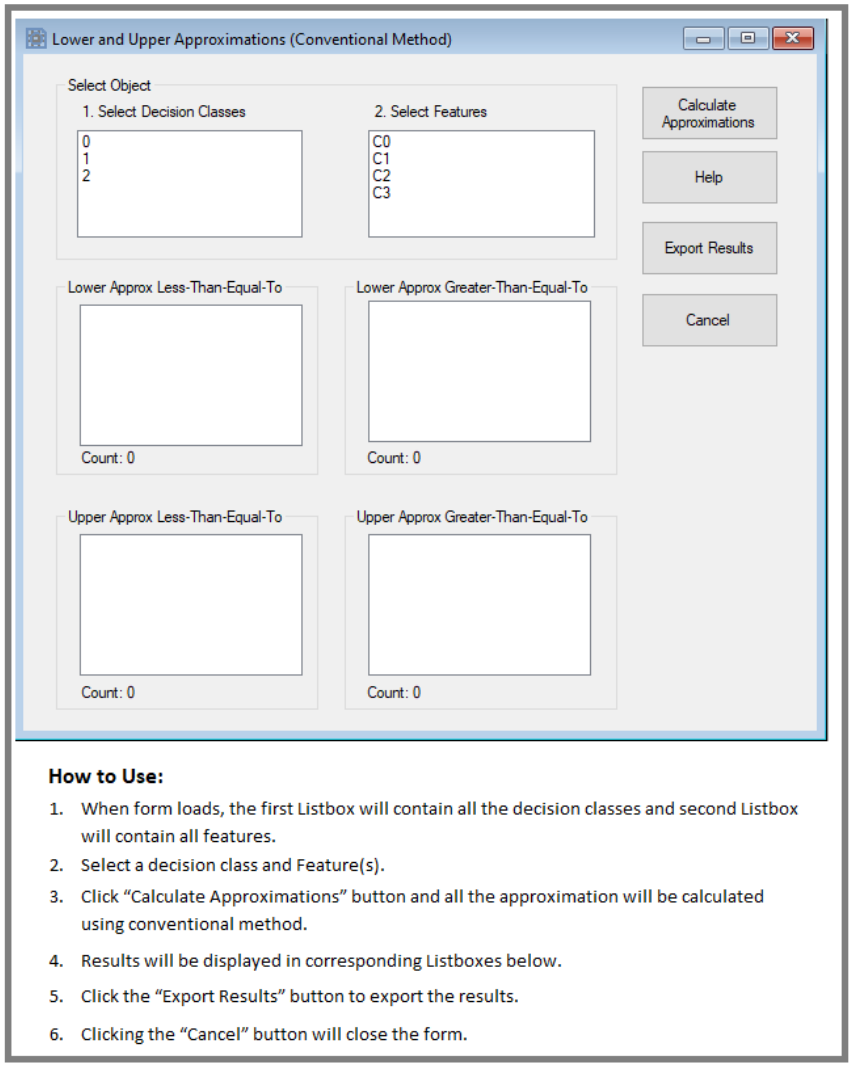
How to Use:

1. When form loads, the first Listbox will contain all the objects in dataset and second Listbox will contain all features.
2. Select an object and feature(s).
3. Click "Calculate Dominance" and all the object having positive dominance on the selected object w.r.t. selected features will be calculated.
4. Results will be displayed in Dominance Negative Listbox below.
5. Click the "Export Results" button to export the results.
6. Clicking the "Cancel" button will close the form.

Form: Calculate Dominance (Positive)



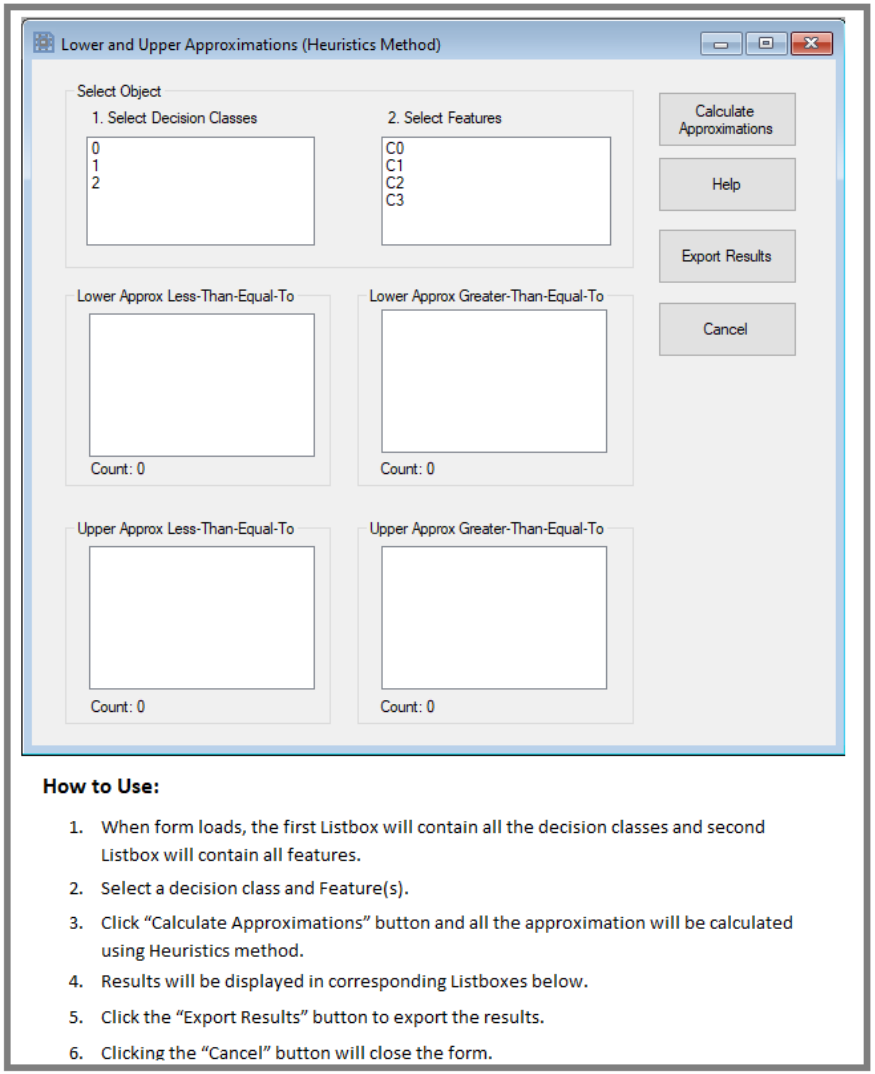
Form: Lower and Upper Approximations (Conventional Method)



How to Use:

1. When form loads, the first Listbox will contain all the decision classes and second Listbox will contain all features.
2. Select a decision class and Feature(s).
3. Click “Calculate Approximations” button and all the approximation will be calculated using conventional method.
4. Results will be displayed in corresponding Listboxes below.
5. Click the “Export Results” button to export the results.
6. Clicking the “Cancel” button will close the form.

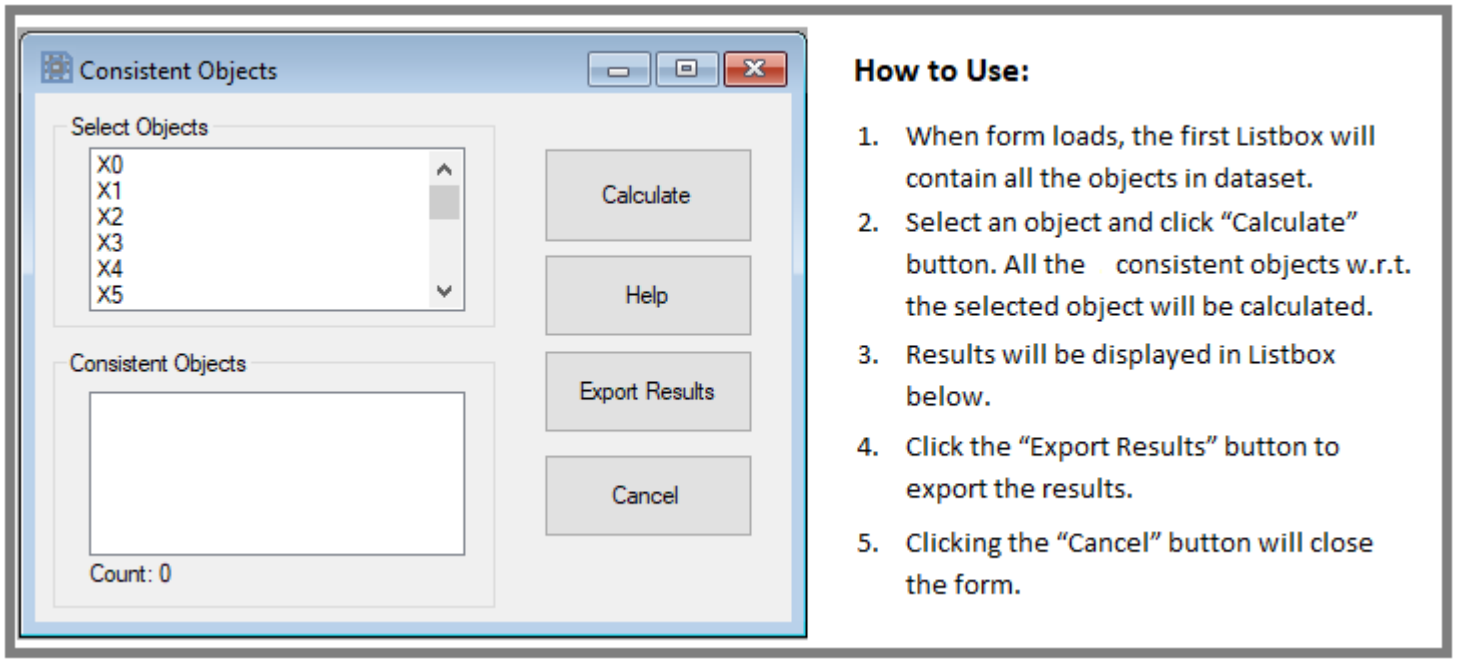
Form: Lower and Upper Approximations (Heuristics Method)



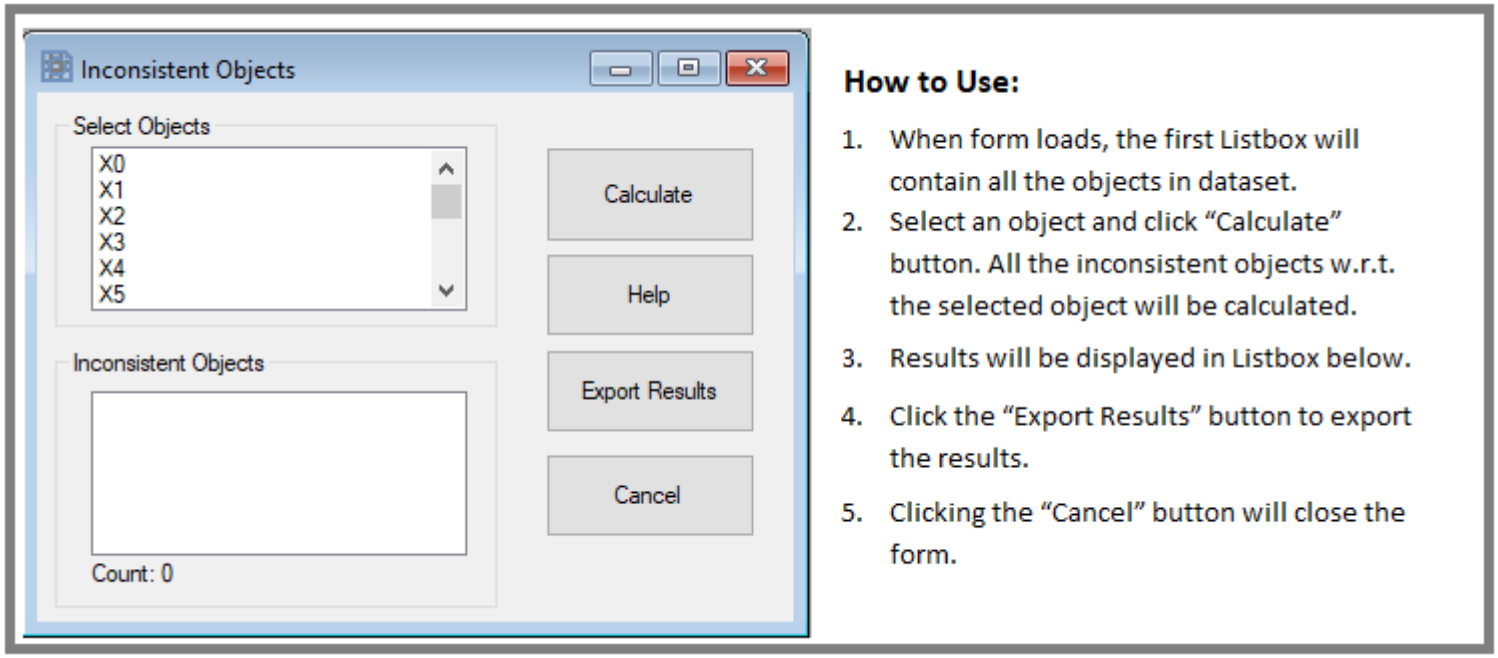
How to Use:

1. When form loads, the first Listbox will contain all the decision classes and second Listbox will contain all features.
2. Select a decision class and Feature(s).
3. Click “Calculate Approximations” button and all the approximation will be calculated using Heuristics method.
4. Results will be displayed in corresponding Listboxes below.
5. Click the “Export Results” button to export the results.
6. Clicking the “Cancel” button will close the form.

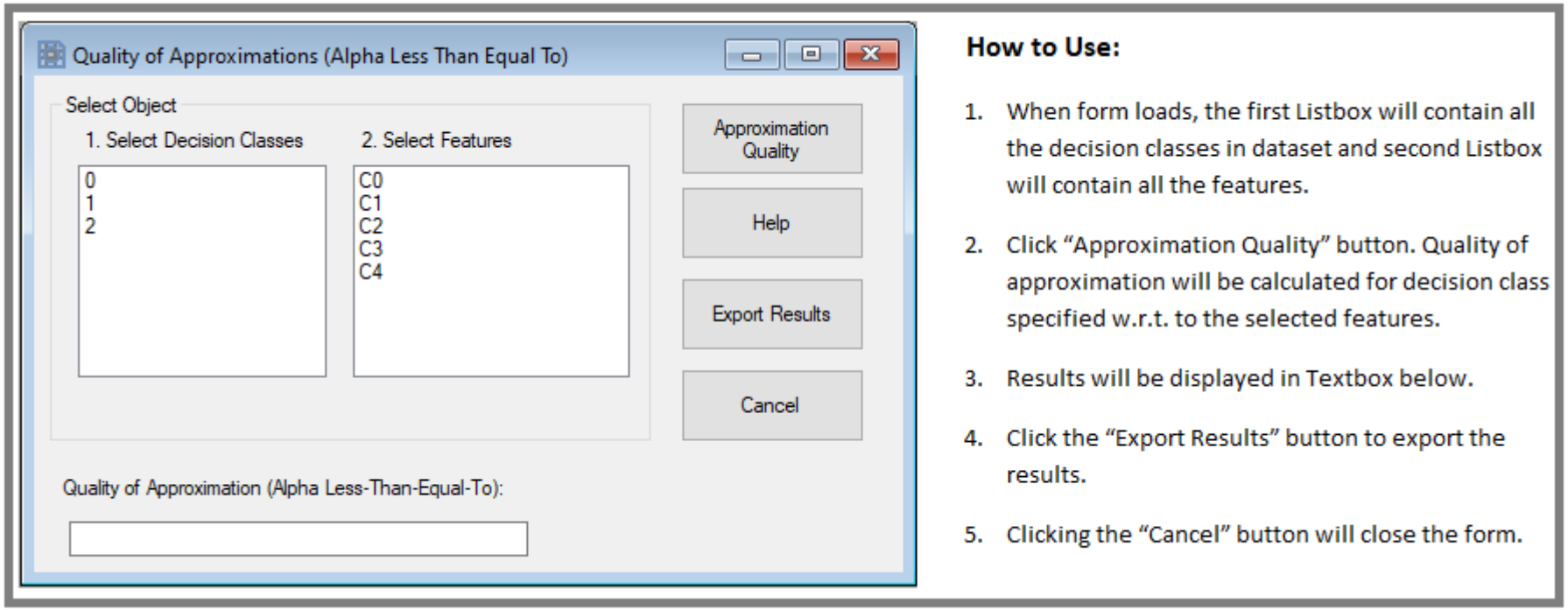
Form: Consistent Objects



Form: Inconsistent Objects



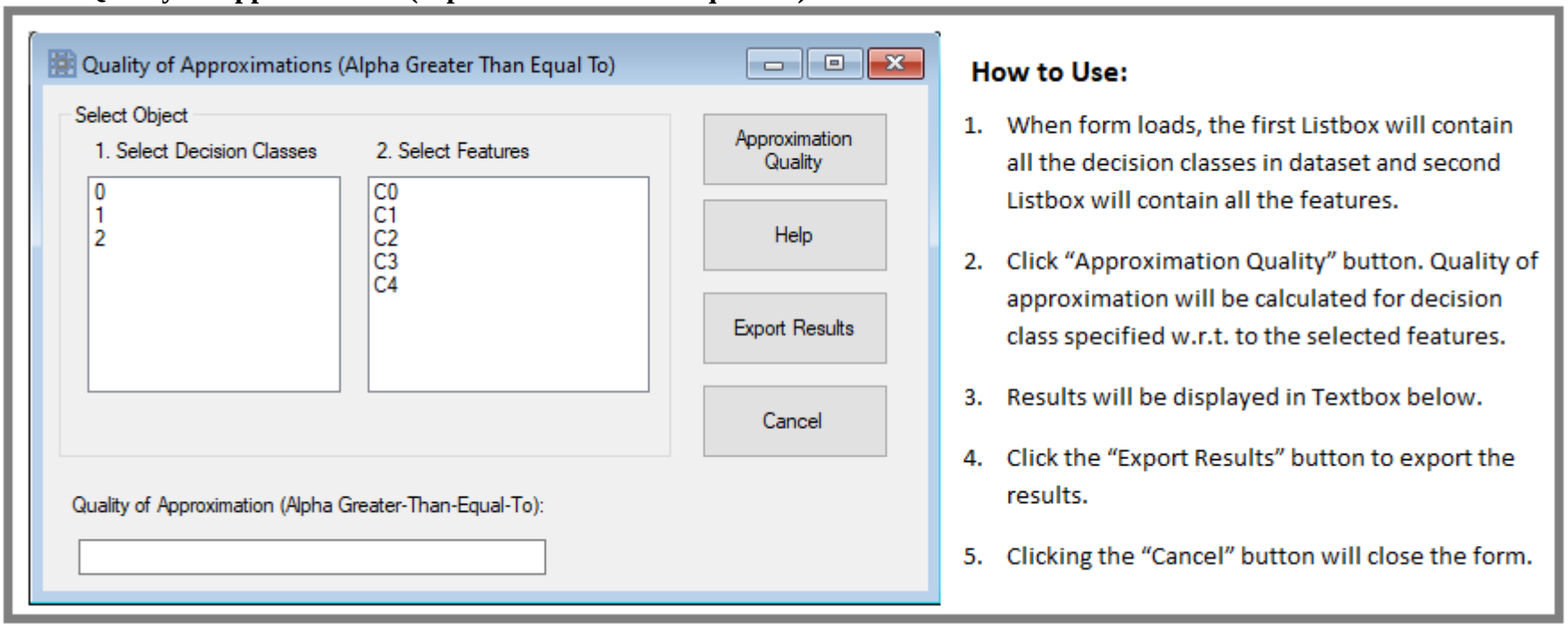
Form: Quality of Approximation (Alpha Less-Than-Equal-To)



How to Use:

1. When form loads, the first Listbox will contain all the decision classes in dataset and second Listbox will contain all the features.
2. Click "Approximation Quality" button. Quality of approximation will be calculated for decision class specified w.r.t. to the selected features.
3. Results will be displayed in Textbox below.
4. Click the "Export Results" button to export the results.
5. Clicking the "Cancel" button will close the form.

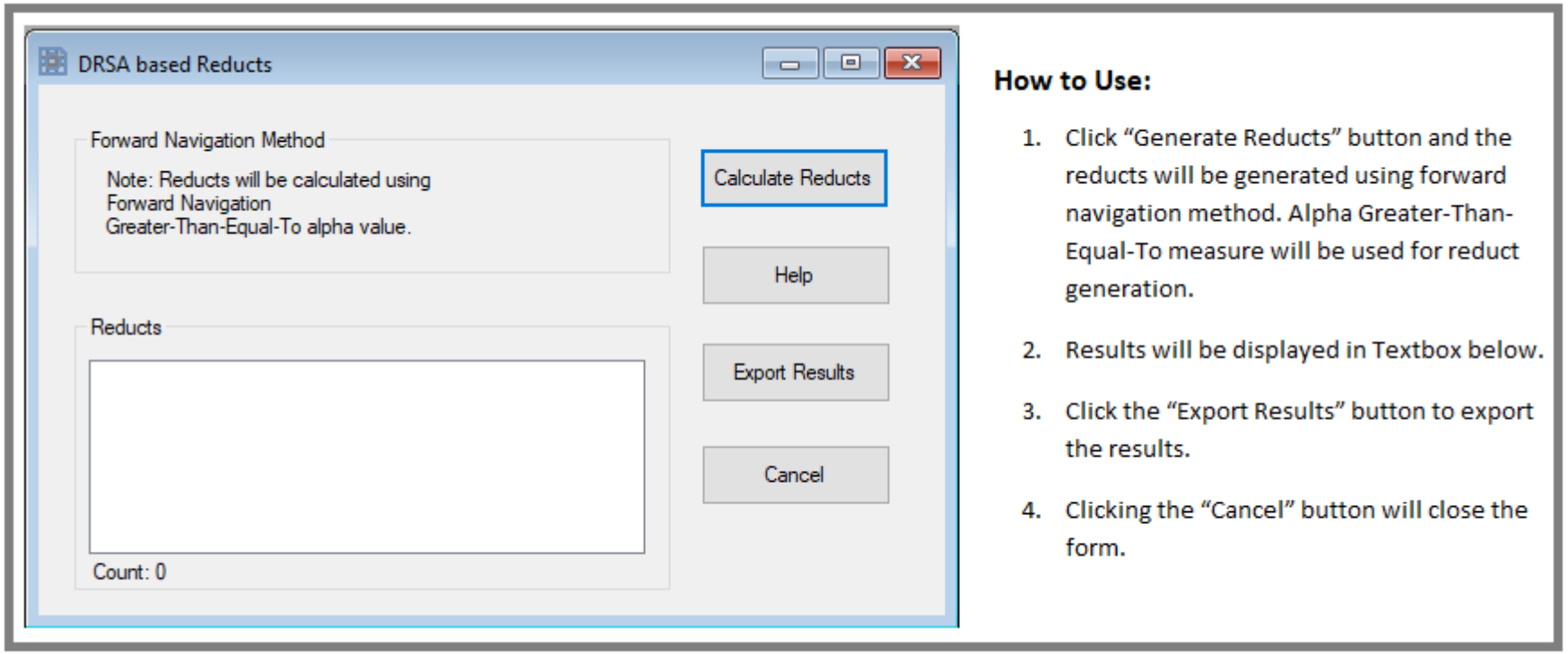
Form: Quality of Approximation (Alpha Greater-Than-Equal-To)



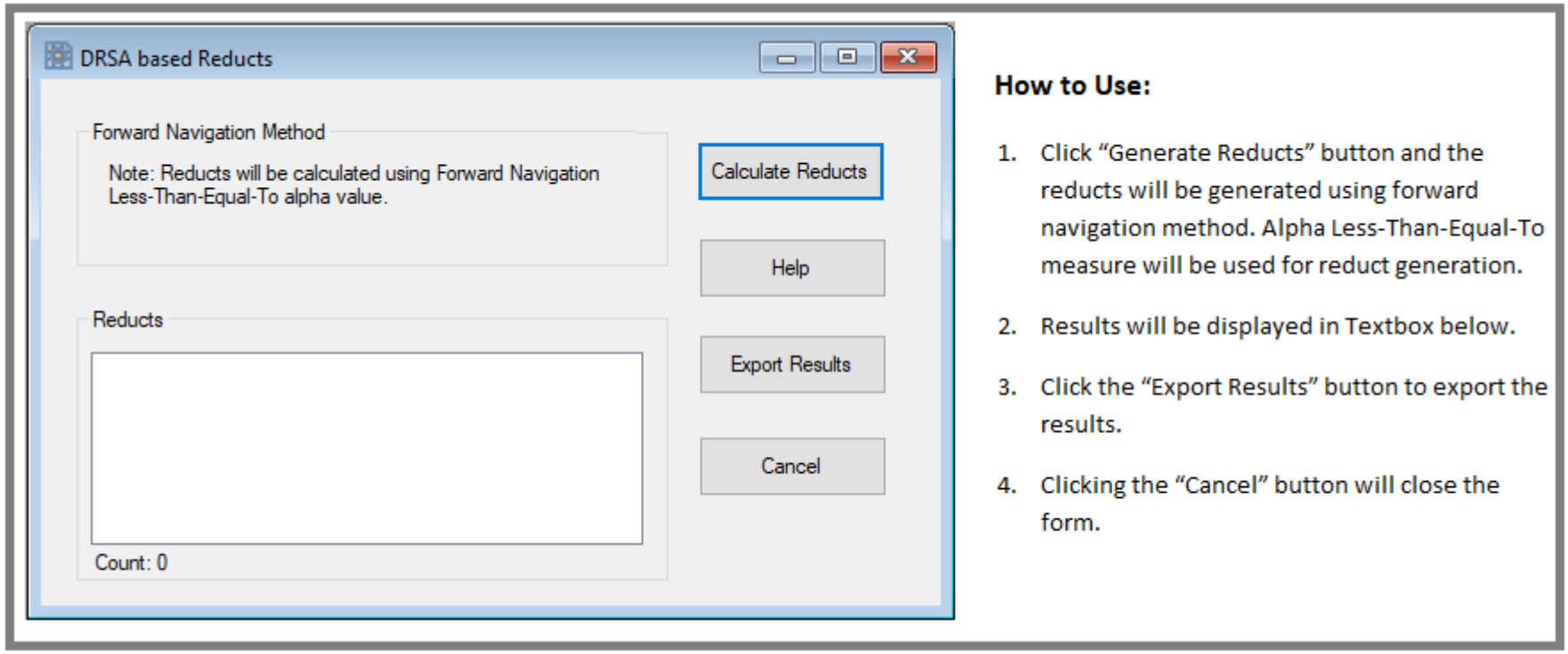
How to Use:

1. When form loads, the first Listbox will contain all the decision classes in dataset and second Listbox will contain all the features.
2. Click "Approximation Quality" button. Quality of approximation will be calculated for decision class specified w.r.t. to the selected features.
3. Results will be displayed in Textbox below.
4. Click the "Export Results" button to export the results.
5. Clicking the "Cancel" button will close the form.

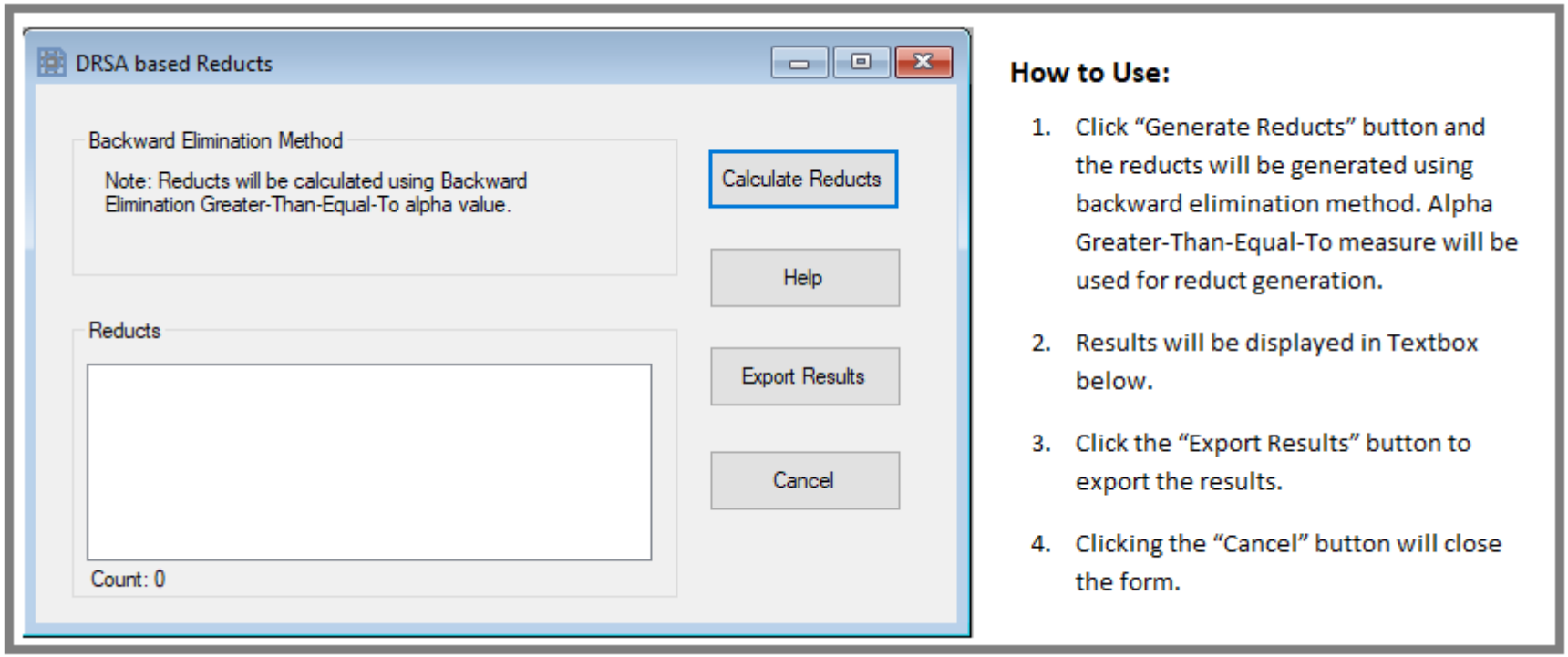
Form: DRSA based Reducts (using forward navigation Greater-Than-Equal-to alpha value)



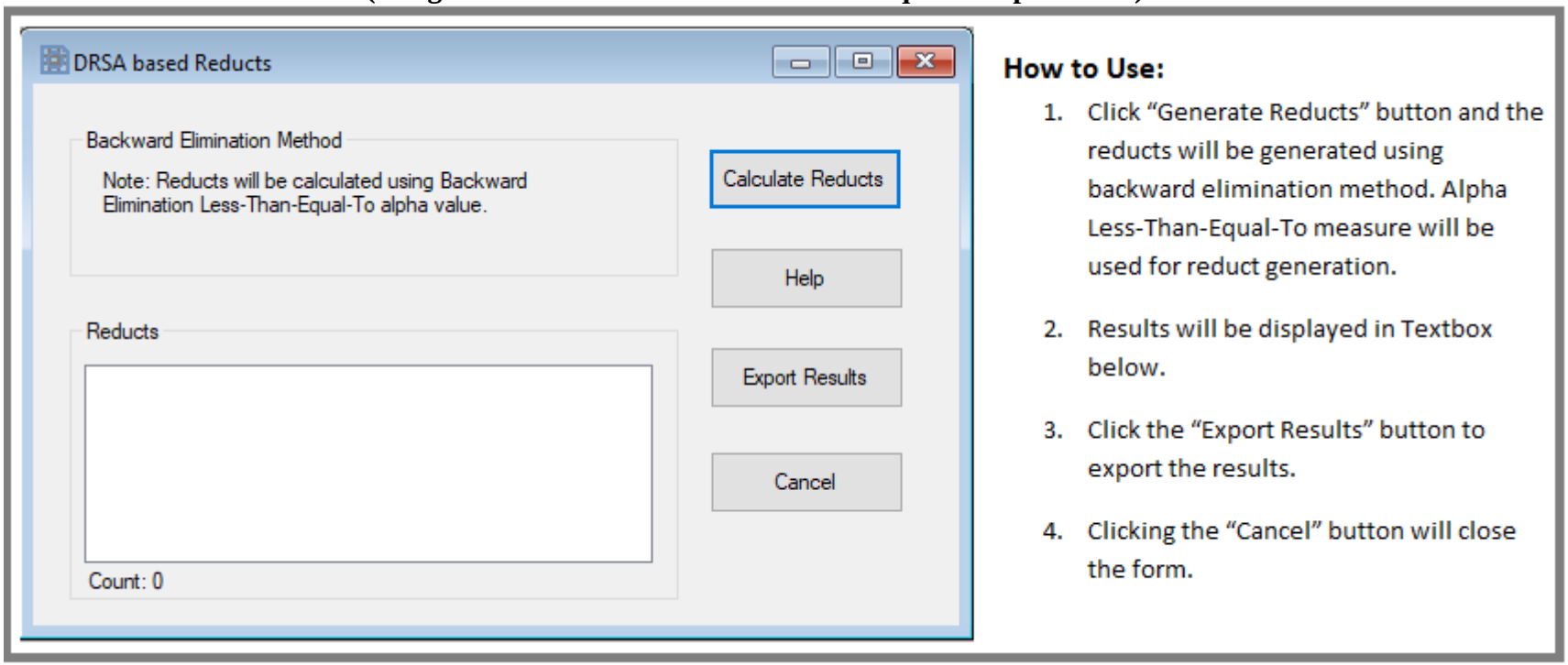
Form: DRSA based Reductions (using forward navigation Less-Than-Equal-to alpha value)



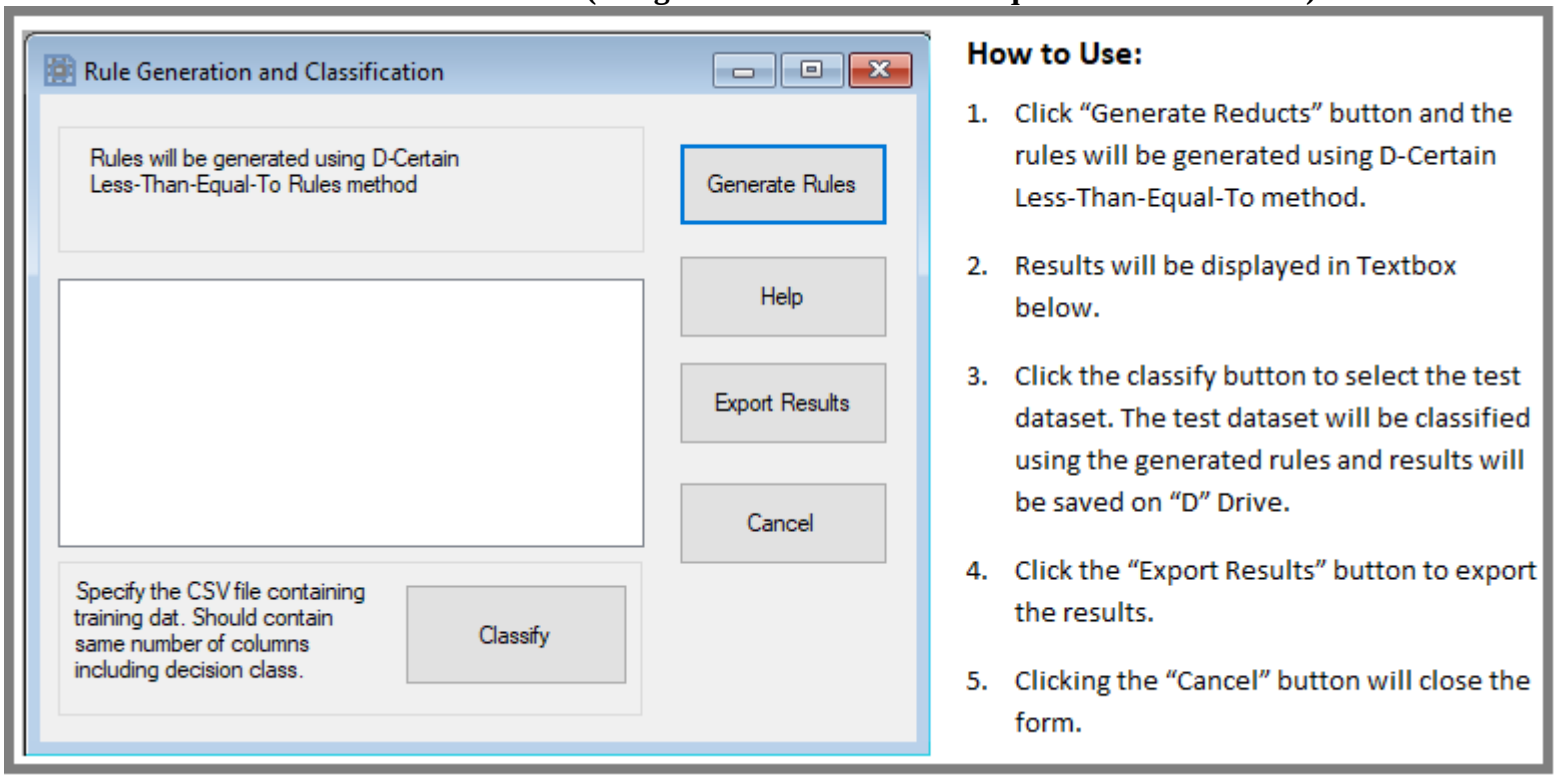
Form: DRSA based Reducts (using backward elimination Greater-Than-Equal-to alpha value)



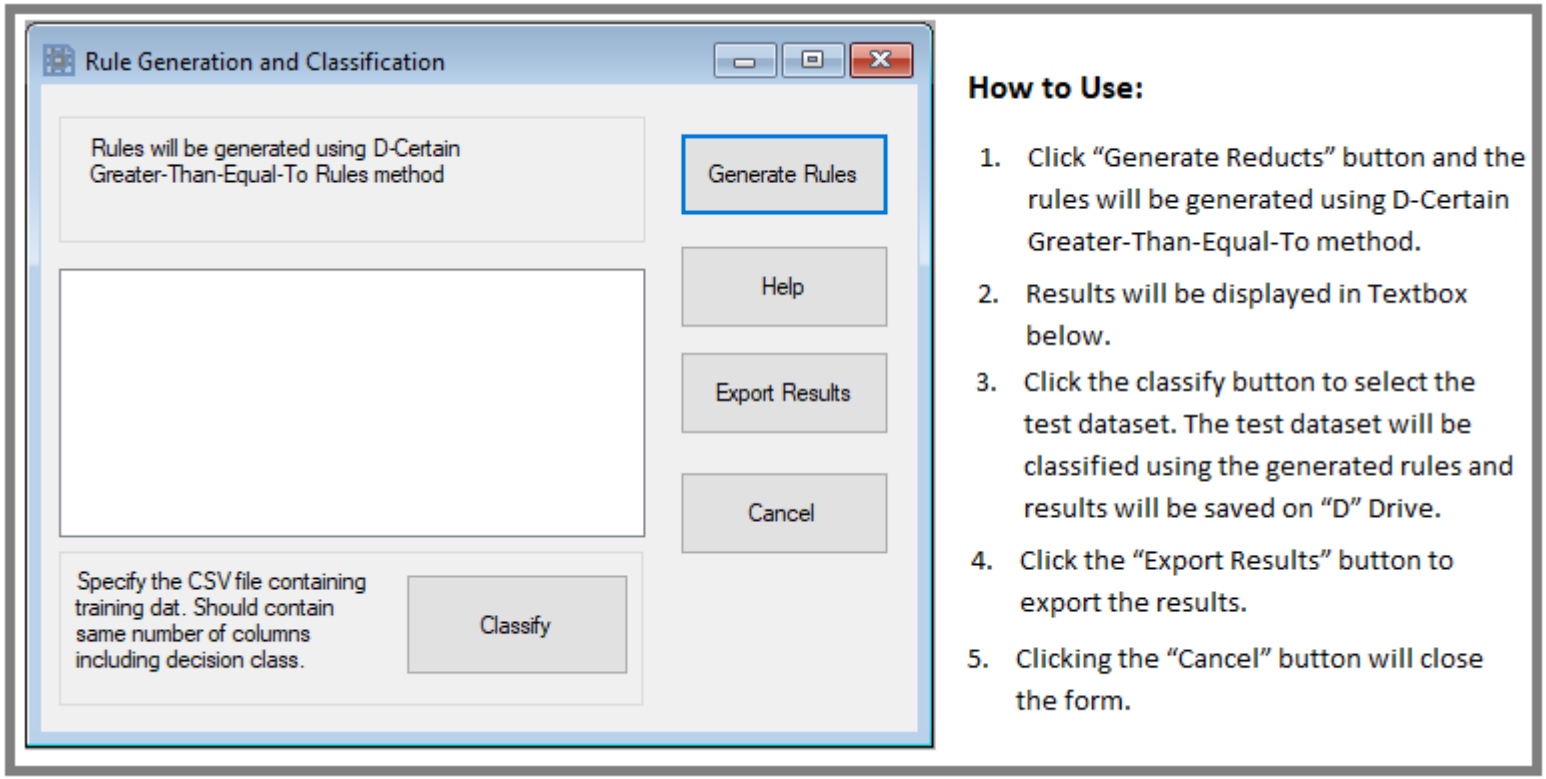
Form: DRSA based Reducts (using backward elimination Less-Than-Equal-to alpha value)



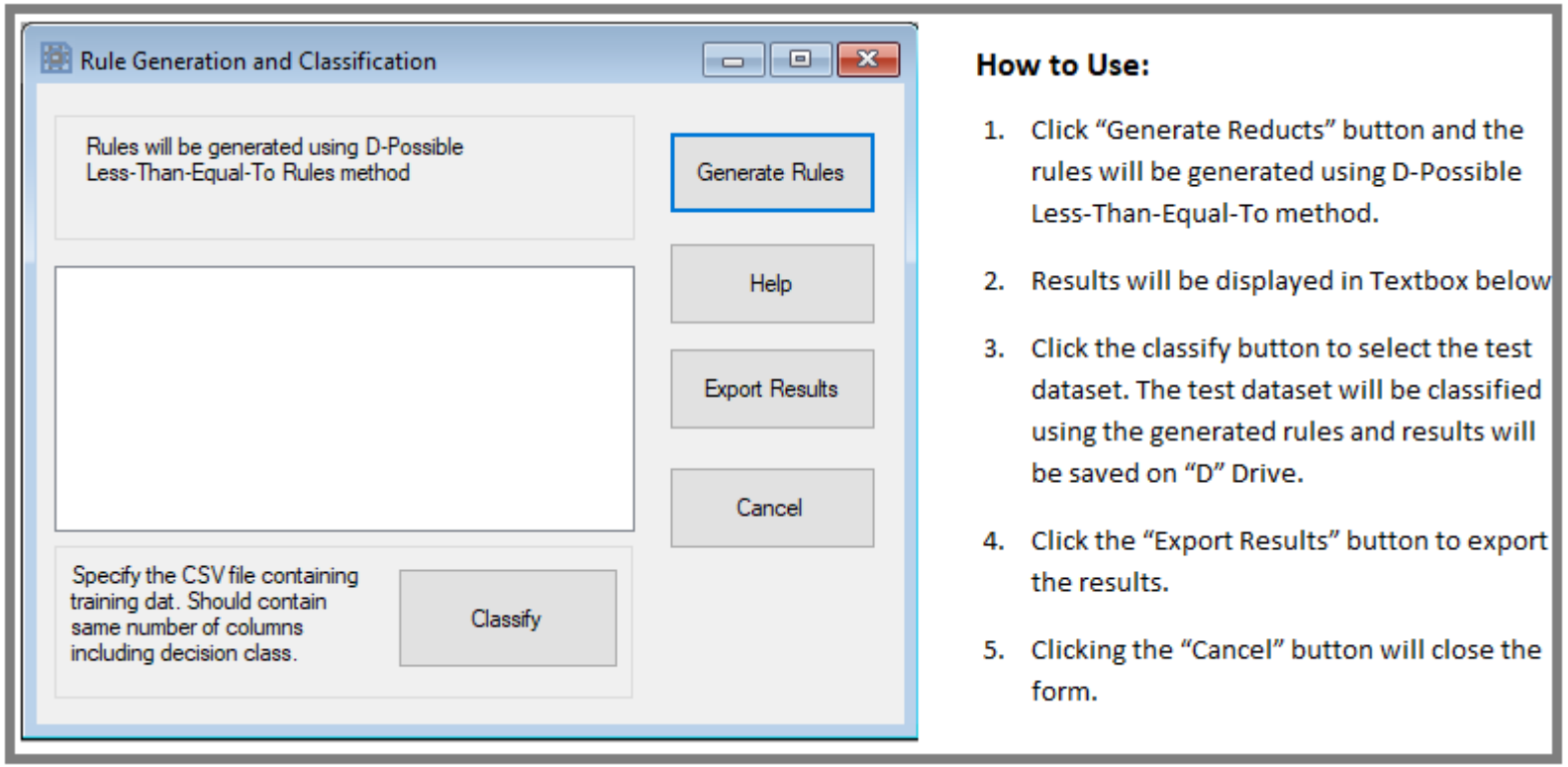
Form: Rule Generation and Classification (using D-Certain Less-Than-Equal-To Rules method)



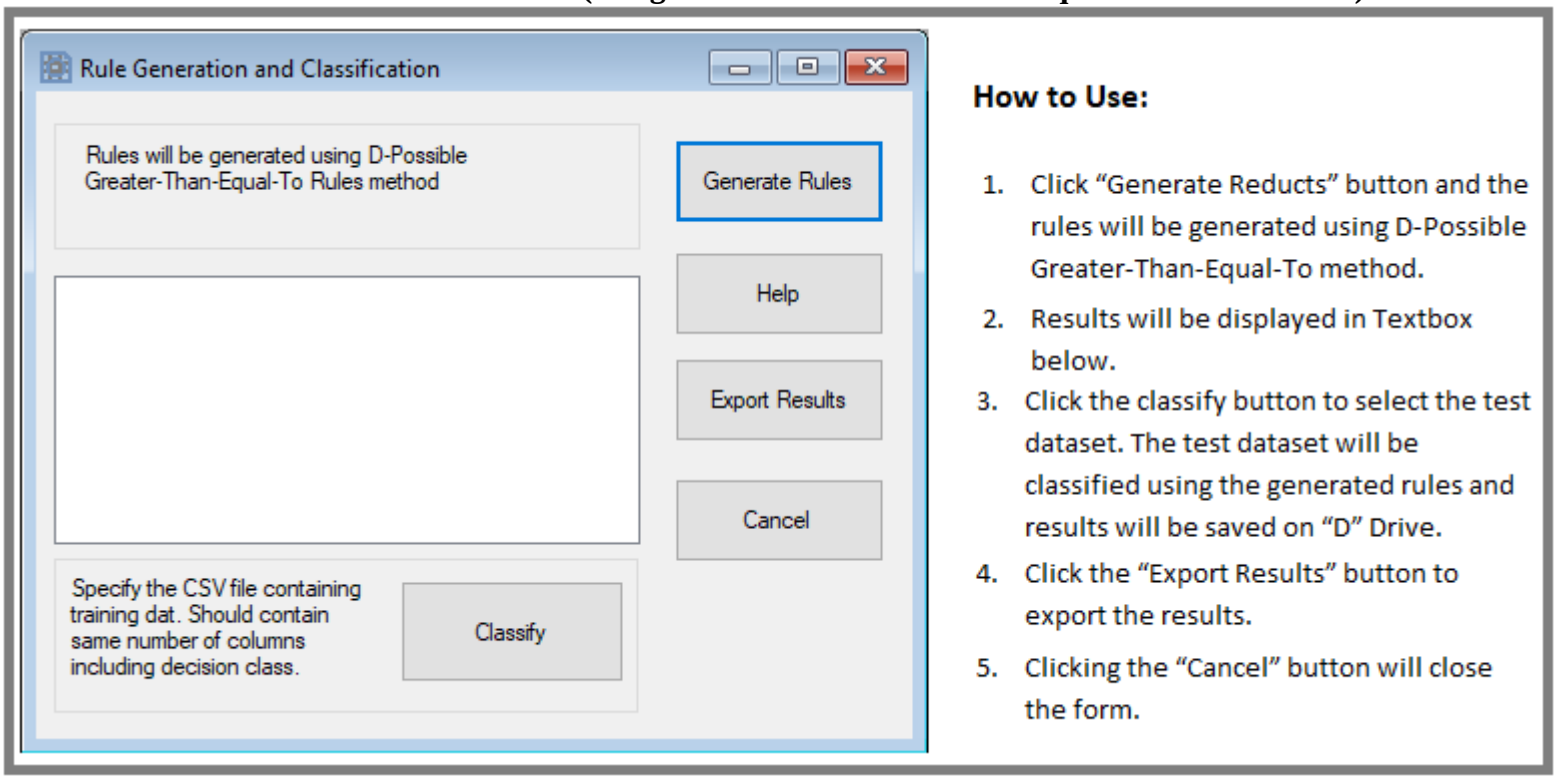
Form: Rule Generation and Classification (using D-Certain Greater-Than-Equal-To Rules method)



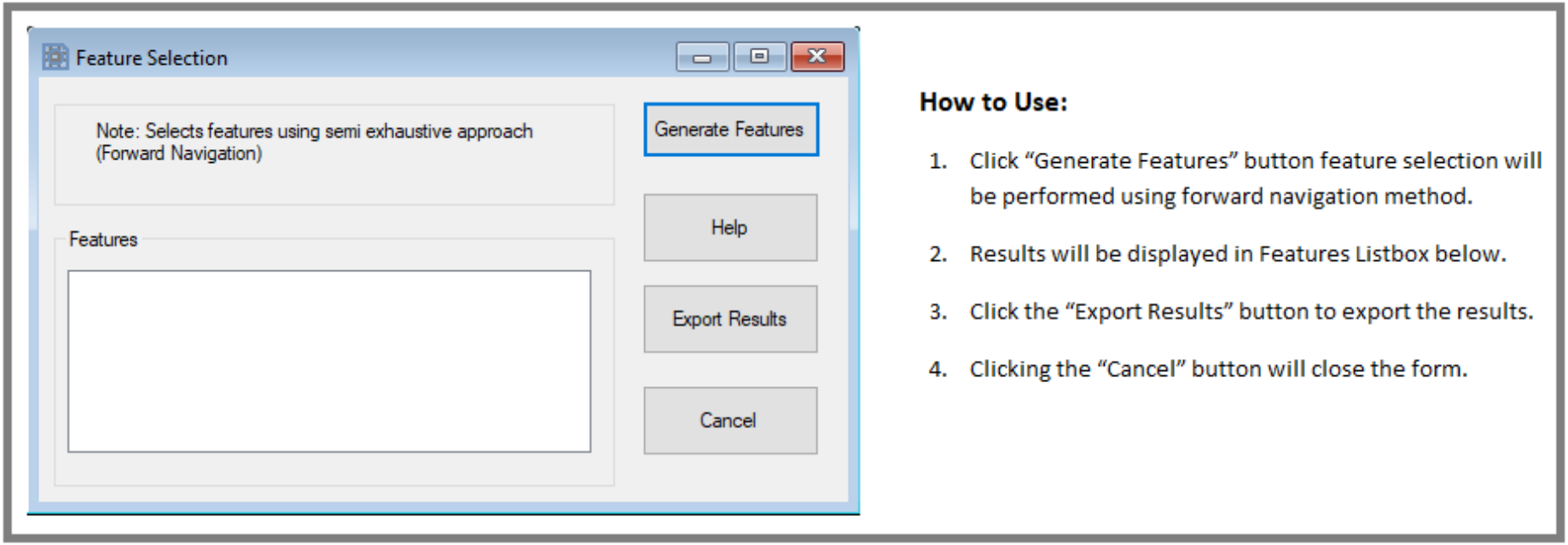
Form: Rule Generation and Classification (using D-Possible Less-Than-Equal-To Rules method)



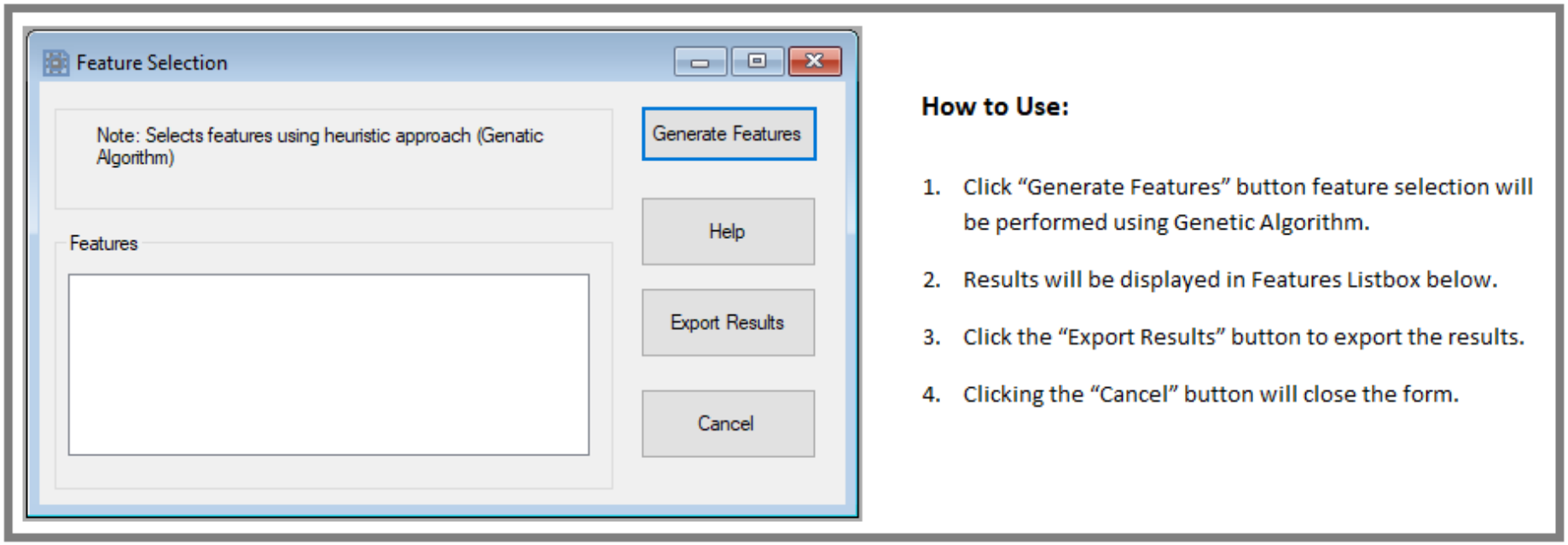
Form: Rule Generation and Classification (using D-Possible Greater-Than-Equal-To Rules method)



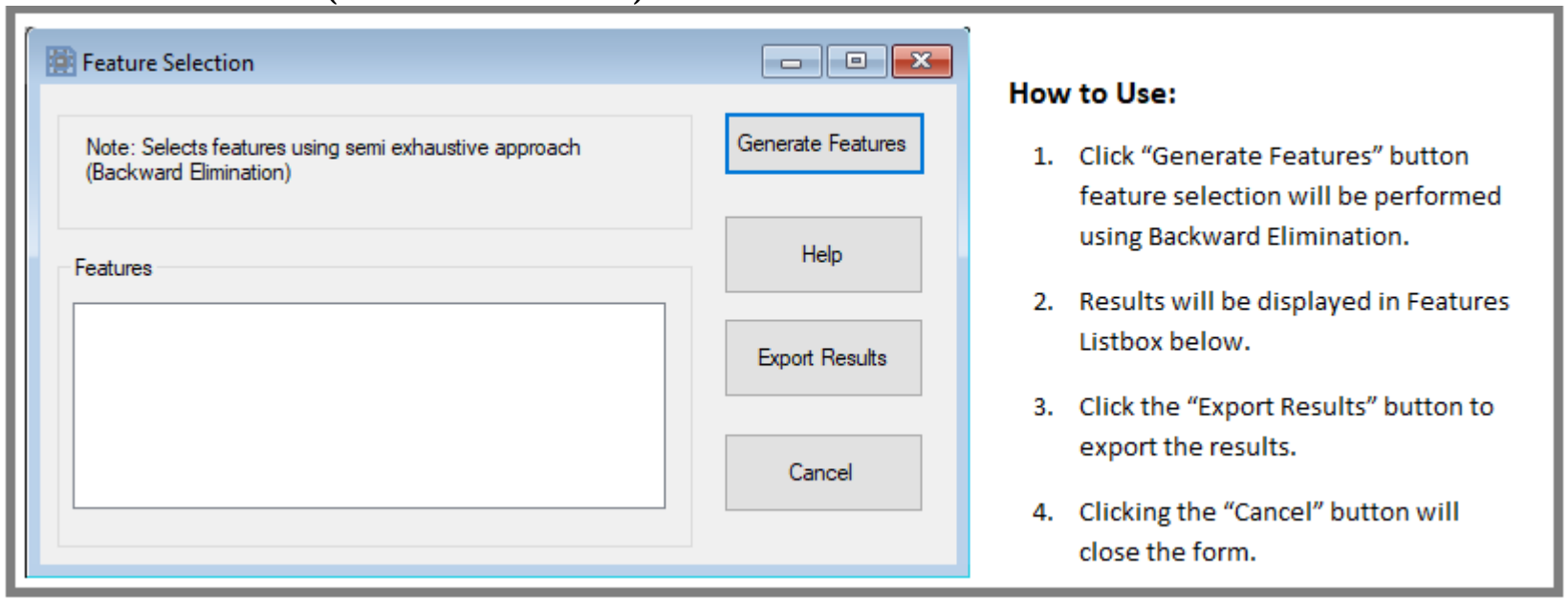
Form: Feature Selection (Forward Navigation)



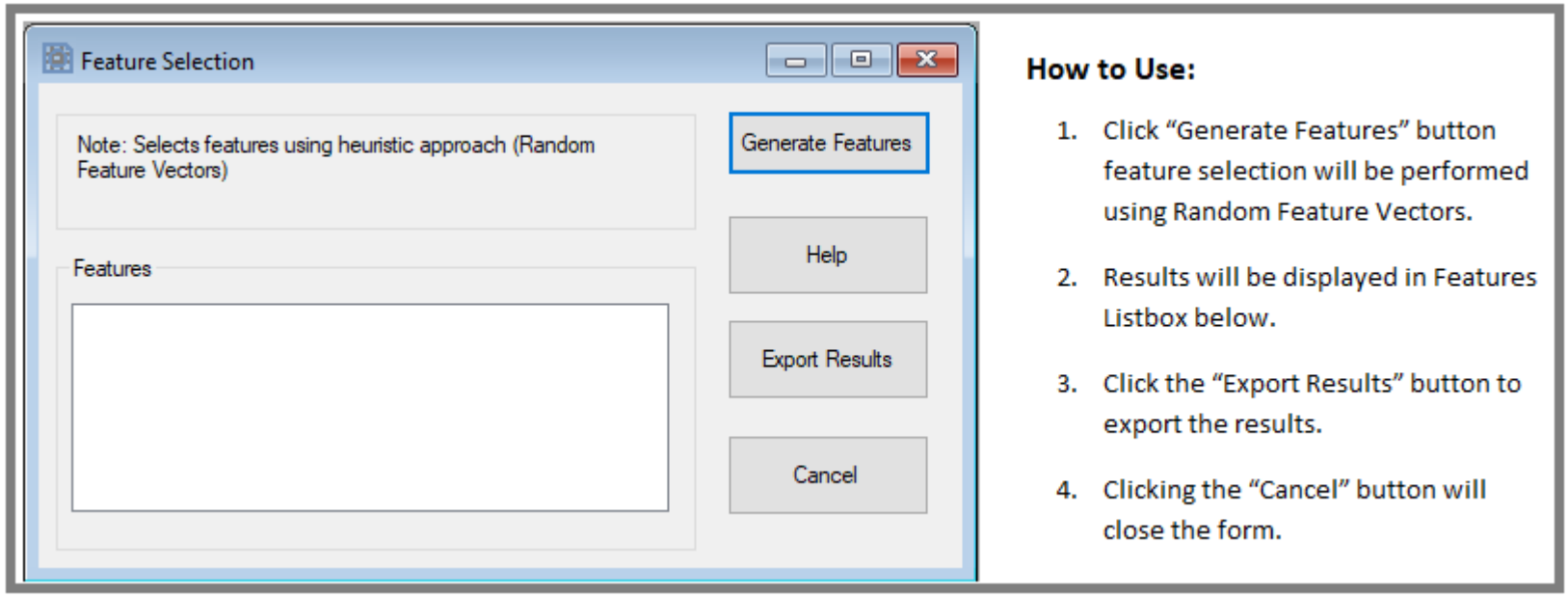
Form: Feature Selection (Genetic Algorithm)



Form: Feature Selection (Backward Elimination)



Form: Feature Selection (Random Feature Vectors)



Technical Aspects

Technical Aspect	Tool Support
• **RoSDACore:** RoSDA Core is the main library that will implement all the above-mentioned functionalities. The library will be implemented using .Net framework and will be deployed in the form of Dynamic Link Libraries (DLLs) for security reasons.	All the functionality is provided in the form of RoSDACore.DLL file.
• **Graphical User Interface (GUI):** The GUI will be provided in all versions of the tool. User through provided interface will be able to perform any type of available tasks. The interface will also provide the facility to export the results at specified path.	User Interface has also been provided.
• **Application Programmer Interface (APIs):** Apart from the provided GUI, the RoSDA core will be available in the form of APIs as well. The APIs can be used by the user in his own source code, e.g., a user can use the RST based dependency measure by calling the appropriate API in his own coded	API have been provided which user can use in his own code. List of classes and API is given in next

algorithm.	
The tool will provide a comparison framework for empirical evaluation of algorithms. The empirical evaluation will comprise of the following two metrics: • **Execution Time:** the total time taken by the algorithm to complete its task right from the input of data to output. • **Memory Used:** The total amount of memory taken by the algorithm to perform its tasks.	Timer Class has been provided DLL file to calculate the execution time of any algorithm. Dataset, RST and DRSA objects provide "GetSize" API which can be used to calculate the total memory of the object used in any algorithm.

APIs

Class: Dataset

API	Details
public Dataset(String path, char separation, char missingchar)	Default constructor, calls the read function to read the data from CSV file.
public Dataset(String path, char separation, char missingchar, HookClass hCs)	Overloaded constructor, calls the read function to read the data from CSV file. takes HookClass object so that user could call his hook function before and after loading the data.
public Dataset Clone()	Returns a copy of current object
public long GetSize()	Returns the size of current object
public int ReadDataSet(String path, char separationchar, char missingchar)	Reads the CSV file.
public int ReadDataSet(String path, char separationchar, char missingchar, HookClass hCs)	Reads the CSV file, user can provide his HookClass object to call hook functions.
public void NormalizeData(int i)	Defines default dominance order depending upon the data type of the feature.
public int GetError()	If dataset is not loaded, this function return “1” (as true).
public string GetElementAt(int row, int col)	Returns data value at specified row and column
public int GetDataSetRows()	Returns total number of rows in a dataset
public int GetDataSetCols()	Returns total number of columns in a dataset
public void SetMissingChar(char missingchar)	Sets value of missing character, this value will be used to identify missing values in dataset
public char GetSeparationChar()	Returns the character used to separate data values.
public bool ContainsMissingValues()	This function is used to check if dataset contains missing values.
public int GetMissingValuesCount()	Returns total number of missing values.
public List<List<String>> GetMissingVaules()	Return the list used to store the row and column number of missing

	values.
public int GetDictionaryRows()	Returns serial number of last row in dictionary
public bool DataSetLoaded()	Returns true if dataset is successfully loaded
private int UpdateFeatureType(string previousftype, string value)	Returns the new datatype of feature
public void RemoveObject(int ObjID)	Deletes the specified object from dataset
public List<List<String>> GetMissingValues()	Returns the reference of MissingValues list
public string GetDictionaryElement(int row, int col)	Returns dictionary element at specified row and column. Note: a feature and its values are stored in a row in dictionary.
public void SetElementIndexInData(int row, int col, int val)	Updates a specified index in data.
public int GetElementIndexInData(int row, int col)	Returns current index value stored in data at specified row and column
public void SetContainsMissingValues(bool hmv)	Sets the "MissingValue" flag.
public void SetMissingValuesCount(int mvc)	Updates the Missing value count
public void UpdateDictionaryElement(int row, int col, string value)	Updates a dictionary element at specified row and column
public void ClearMissingValues()	Sets missing value array as empty.
public int GetFeatureType(int FID)	Returns the datatype of specified feature.
public int SearchDictionary(int col, String Element)	Searches the dictionary. "col" means a column number of a feature.
public void SortIndexes(int FID, List<string> newvalues)	Updates feature dominance order according to the newvalues specified.
public bool FeatureHasMissingValues(int FID)	Return true if feature has missing values.
public bool MatchObjects(int o1, int o2)	Returns true if two objects are equal.

Class: RST

API	**Details**
public RST(Dataset dt)	Default constructor
public RST(Dataset dt, HookClass HCls)	Overloaded constructor, user can specify HoockClass object
public int GetDecisionClassID()	Returns the serial number of decision class feature.
public void SetDecisionClassID(int DID)	Sets the serial number of decision class feature.

public int GetDecisionClassCount()	Returns number of decision classes
public List<int> GetDecisionClassesIndeces()	Returns index number of decision classes in dictionary
public List<string> GetDecisionClasses()	Returns the decision classes
public List<string> GetFeatureValueSet(int FID)	Returns the value set of a feature
public List<int> GetX(int X)	Returns the objects that fulfills the concept X
public List<int> GetX(String X)	Overloaded function, returns the object that fulfills the concept X
public List<List<int>> GetPIndiscernibleObjects(List<int> P)	Returns indiscernible object w.r.t. P
public List<int> GetConPLowerAprox(String X, List<int> P)	Returns lower approximation using conventional approach
public List<int> GetHeuPLowerAprox(string X, List<int> P)	Returns lower approximation using heuristics-based approach
public List<int> GetTHeuPLowerAprox(String X, List<int> P)	Returns lower approximation using parallel heuristics-based approach
public float CalculateCDependency(List<int> P)	Calculates positive region based dependency using conventional approach
public float CalculateHDependency(List<int> P)	Calculates positive region based dependency using heuristics based approach.
public float CalculateTHDependency(List<int> P)	Calculates positive region based dependency using threaded heuristics based approach.
public List<int> GetConPUpperAprox(String X, List<int> P)	Calculates upper approximation using conventional approach
public List<int> GetHeuPUpperAprox(string X, List<int> P)	Calculates upper approximation using heuristics based approach.
public List<int> GetTHeuPUpperAprox(string X, List<int> P)	Calculates upper approximation using threaded heuristics based approach.
public bool IsDatasetConsistent()	Returns true if dataset is consistent
public List<int> FindInconsistentObjects()	Returns the set inconsistent objects
public Dataset MakeDatasetConsistent()	Makes dataset consistent by removing inconsistent objects
public double CalculateRDependency(List<int> P)	Calculates relative dependency using conventional approach.
public double CalculateRDOnePass(List<int> P)	Calculate relative dependency using heuristics based approach
public double CalculateTRDependency(List<int> P)	Calculate relative dependency using parallel heuristics based approach
public string GetDiscernibilityMatrix(List<int> P)	Returns reducts using discernibility matrix

public List<List<string>> GenerateRulesUsingGlobalReducts(int RuleGenOption)	Returns two dimensional list containing the rules generated using global reducts
public List<List<string>> GenerateRulesUsingLocalReducts(int OID)	Returns two dimensional list containing the rules generated using local reducts
public string QuickReduct()	Returns reducts using QuickReduct algorithm.
public string RelaiveReduct()	Returns reducts using Relative dependency.
public string LocalReducts(int OID)	Returns local reducs w.r.t. to specified object
public string DynamicReducts()	Returns dynamic reducts.
public string ConDepReductFN()	Returns reduct using forward navigation algorithm (conventional dependency based)
public string ConDepReductBE()	Returns reduct using backward elimination algorithm (conventional dependency based)
public string GenaticAlgorithm(int NumberofGenerations, int PopulationSize)	Returns reducts using genetic algorithm
public string RandomFeatureVectors(int NumberofIterations)	Returns reducts using random feature vectors
public string GetDecisionClass(int OID)	Returns decision class of specified object
public List<List<string>> ClassifyTestData(Dataset TDts, int RuleGenerationOption)	Performs classification of the specified test dataset. Returns two-dimensional list containing the actual classes and predicted classes
public string[,] CalculateConfusionMatrix(List<string> Actual, List<string> Predicted)	Constructs and returns confusion matrix.
public int FN(string[,] CM, string Cls)	Calculates and returns false negatives using mentioned confusion matrix and decision class.
public int FP(string[,] CM, string Cls)	Calculates and returns false positives using mentioned confusion matrix and decision class.
public int TP(string[,] CM, string Cls)	Calculates and returns Returns true positives using mentioned confusion matrix and decision class.
public int TN(string[,] CM, string Cls)	Calculates and returns true negatives using mentioned confusion Calculates and returns and decision class.
public double Precision(string[,] CM, string Cls)	Calculates and returns precision using mentioned confusion matrix and decision class.

public double Recall(string[,] CM, string Cls)	Calculates and returns Recall using mentioned confusion matrix and decision class.
public double Accuracy(string[,] CM)	Calculates and returns Accuracy using mentioned confusion matrix.
public double Specificity(string[,] CM, string Cls)	Calculates and returns Specificity using mentioned confusion matrix and decision class.
public long GetSize()	Returns size of the current object

Class: DRSA

API	**Details**
public DRSA(Dataset dt)	Default constructor
public DRSA(Dataset dt, HookClass HCls)	Overloaded constructor, user can specify HookClass object
public List<int> UpWardUnion(String t)	Returns upward union of mentioned class
public List<int> DownWardUnion(String t)	Returns downward union of mentioned class
public List<int> DpPlus(int X, List<int> P)	Returns all objects Greater-Than-Equal-To object X
public bool YDpX(int Y, int X, List<int> P)	Returns true if object Y is Greater-Than-Equal-To object X
public bool XDpY(int X, int Y, List<int> P)	Returns true if object X is Greater-Than-Equal-To object Y
public List<int> DpMinus(int X, List<int> P)	Returns all objects Less-Than-Equal-To object X
public List<int> PL_GE(string t, List<int> P)	Returns P-Lower approximation for upward union of classes using conventional approach
public List<int> HPL_GE(string t, List<int> P)	Returns P-Lower approximation for upward union of classes using heuristics based approach
public List<int> PL_LE(string t, List<int> P)	Returns P-Lower approximation for downward union of classes using conventional approach
public List<int> HPL_LE(string t, List<int> P)	Returns P-Lower approximation for downward union of classes using heuristics based approach
public List<int> PU_GE(string t, List<int> P)	Returns P-Upper approximation for downward union of classes

	using conventional approach
public List<int> HPU_GE(string t, List<int> P)	Returns P-Upper approximation for upward union of classes using heuristics based approach
public List<int> PU_LE(string t, List<int> P)	Returns P-Upper approximation for downward union of classes using conventional approach
public List<int> HPU_LE(string t, List<int> P)	Returns P-Upper approximation for downward union of classes using heuristics based approach
public List<int> Subtract(List<int> Superset, List<int> Subset)	Subtracts subset from superset and returns the resultant set
public List<int> GetConsistentObjects(int OID)	Returns consistent objects w.r.t. specified object
public List<int> GetInconsistentObjects(int OID)	Returns inconsistent objects w.r.t. specified object
public double QualityofApproximationGE(string t, List<int> P)	Returns quality of approximation using alpha for upward union of classes
public double QualityofApproximationLE(string t, List<int> P)	Returns quality of approximation using alpha for downward union of classes
public List<string> GetSortedDCs()	Returns decision classes in sorted order
public float CalculateAlphaGE(List<int> P)	Calculates accuracy of approximation using upward union of lasses
public float CalculateAlphaLE(List<int> P)	Calculates accuracy of approximation using downward union of lasses
public string CalculateReductFNGE()	Calculates reducts using forward navigation algorithm (based on accuracy of approximation measure using upward union of lasses)
public string CalculateReductFNLE()	Calculates reducts using forward navigation algorithm (based on accuracy of approximation measure using downward union of lasses)
public string CalculateReductBEGE()	Calculates reducts using backward elimination algorithm (based on accuracy of approximation measure using upward union of lasses)
public string CalculateReductBELE()	Calculates reducts using backward elimination algorithm (based on accuracy of approximation measure using downward union of lasses)

public List<List<string>> DCertainGERules()	Returns D-Certain rules for upward union of classes
public List<List<string>> DPossibleGERules()	Returns D-Possible rules for upward union of classes
public List<List<string>> DCertainLERules()	Returns D-Certain rules for downward union of classes
public List<List<string>> DPossibleLERules()	Returns D-Possible rules for downward union of classes
public List<List<string>> ClassifyTestDataGE(Dataset TDts, int RuleGenerationOption)	Performs classification of the test dataset (based on the rules generated using upward union of classes) and returns two dimensional list containing actual and predicted decision classes
public List<List<string>> ClassifyTestDataLE(Dataset TDts, int RuleGenerationOption)	Performs classification of the test dataset (based on the rules generated using downward union of classes) and returns two dimensional list containing actual and predicted decision classes
public long GetSize()	Returns size of the current object

Class: MissingValueManager

API	**Details**
public MissingValuesManager()	Default constructor
public Dataset RemoveMissing(Dataset ds)	Removes all objects containing missing values.
public Dataset RemovebyMeanMode(Dataset ds)	Replaces the missing values by Mean and Mode operation
public Dataset RemoveByCloseDistance(Dataset ds)	Replaces missing values by using closest objects
public List<Int32> GetMissingValuedObjects(Dataset ds)	Returns set of objects containing missing values
public List<Int32> GetMeanValuedCols(Dataset ds)	Returns features where missing values will be replaced using Mean operation
public List<Int32> GetModeValuedCols(Dataset ds)	Returns features where missing values will be replaced using Mode operation

Class: Discretizer

public Discretizer()	Default constructor
public List<List<int>> GetIntBinsEW(int FID, int	Returns equal width bins for specified feature

nBins, Dataset ds)	
public List<List<int>> GetIntBinsEF(int FID, int nBins, Dataset ds)	Returns equal frequency bins for specified feature
public Dataset SmoothinByBinMeans(int FID, int nBins, Dataset ds, int BinType)	Smooths (discretizes) feature by using Mean operation. BinType specifies the type (Equal width or Equal frequency) of bins that will be created.
public int FindBinID(List<List<int>> bins, double fValue)	Returns bin ID of the specified value.
public Dataset SmoothinByBinMode(int FID, int nBins, Dataset ds, int BinType)	Smooths (discretizes) feature by using Mod operation. BinType specifies the type (Equal width or Equal frequency) of bins that will be created.
public Dataset SmoothinByBinBoundaries(int FID, int nBins, Dataset ds, int BinType)	Smooths (discretizes) feature by using bin Boundaries operation. BinType specifies the type (Equal width or Equal frequency) of bins that will be created.

Class: Timer

API	**Details**
public Timer()	Default constructor
public void Start()	Starts the stopwatch
public void Stop()	Stops the stopwatch
public int GetHours()	Returns the number of Hours elapsed between start and stop operation.
public int GetMinutes()	Returns the number of Minutes elapsed between start and stop operation.
public int GetSeconds()	Returns the number of Seconds elapsed between start and stop operation.
public int GetMilliSeconds()	Returns the number of MilliSecons elapsed between start and stop operation.

Class: HookClass

API	Details
public abstract void HCPreDatasetLoad(Dataset ds);	Hook function to be called before loading the dataset
public abstract void HCPostDatasetLoad(Dataset ds);	Hook function to be called after loading the dataset
public abstract void HCPreRSTLoad(Dataset ds);	Hook function to be called from RST class constructor
public abstract void HCPreDRSALoad(Dataset ds);	Hook function to be called from DRSA class constructor

Class: Miscellaneous

API	Details
public Miscellaneous()	Default constructor
public static int FindFeatureType(string Item)	Returns the possible datatype of the item specified

Disclaimer:

If any damage is caused by RoSDA Version 1.1 to the computer system or loss of data occurs because of usage of RoSDA Version 1.1, the developers shall not be held responsible.