**PRACTICAL 2**

**AIM: Introduction to R programming: R-GUI, RStudio – Basic working & Commands**

**S/W: RStudio/RGUI**

**H/W: --**

**Theory:**

R  is a  programming  language and  software environment  for statistical analysis,  graphics representation and reporting. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team.

R  is freely  available under  the GNU General Public  License, and pre-compiled  binary versions are provided   for various operating systems   like Linux, Windows and Mac. This programming  language was named **R**,  based  on the  first letter  of first name  of the two R authors (Robert Gentleman and Ross Ihaka), and partly a play on the name of the Bell Labs Language **S**.

**Evolution of R:**

R was initially written by **Ross Ihaka** and **Robert Gentleman** at the Department of Statistics of the University of Auckland in Auckland, New Zealand. R made its first appearance in 1993.

* A large group of individuals has contributed to R by sending code and bug reports.
* Since  mid-1997  there has  been a core  group (the "R  Core Team") who  can modify the R source code archive.

**Features of R:**

As  stated  earlier,  R is a programming  language and software  environment for statistical  analysis,graphics representation and reporting. The following are the important features of R –

•    R   is a   well-developed,   simple and effective   programming language which   includes conditionals, loops, user defined recursive functions and input and output facilities.

* R has an effective data handling and storage facility,
* R provides a suite of operators for calculations on arrays, lists, vectors and matrices.
* R provides a large, coherent and integrated collection of tools for data analysis.
* R provides graphical facilities for data analysis and display either directly at the computer or printing at the papers.

There are many types of R-objects. The frequently used ones are −

* Vectors
* Lists
* Matrices
* Arrays
* Factors
* Data Frames

There are six data types and they are as follow:

* Logical
* Numeric
* Integer
* Complex
* Character
* Raw

**Commands:**

•  print() – to print any value/string.

•  classs() – to get the data type of the variable.

•  name() – assign names .

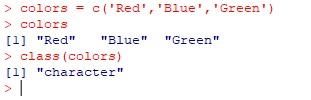
•  charToRaw() – convert string into hexa-decimal.

Assigning values to variables:

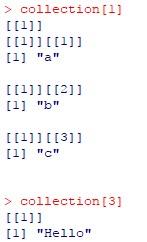
•  variable <- value

•  variable = value

**Vectors:** To create vector with one or more element use c() function.



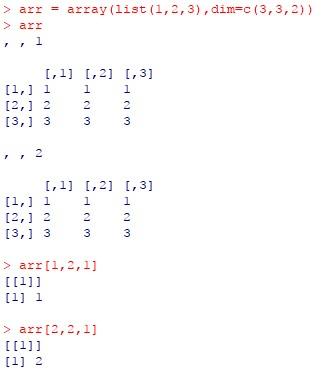
**Lists:** A list is an R-object which can contain many different types of elements inside it like vectors, functions and even another list inside it.

Accessing Element:

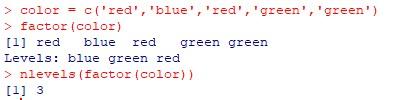
**Array:** While matrices are confined to two dimensions, arrays can be of any number of dimensions.

The array function takes a dim attribute which creates the required number of dimension.



**Factors:** Factors are the r-objects which are created using a vector. It stores the vector along with the distinct values of the elements in the vector as labels. The labels are always character irrespective of whether it is numeric or character or Boolean etc. in the input vector. They are useful in statistical modeling.

Factors are created using the factor() function. The nlevels functions gives the count of levels.



**Matrix:**

The basic syntax for creating a matrix in R is −

matrix(data, nrow, ncol, byrow, dimnames)

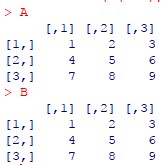
Following is the description of the parameters used −

 data is the input vector which becomes the data elements of the matrix.

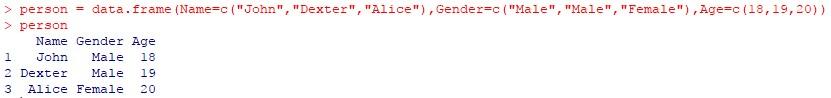
 nrow is the number of rows to be created.

 ncol is the number of columns to be created.

 byrow is a logical clue. If TRUE then the input vector elements are arranged by row



**Data Frame:** Data frames are tabular data objects. Unlike a matrix in data frame each column can contain different modes of data. The first column can be numeric while the second column can be character and third column  can be logical. It is a list of vectors of equal length. Data Frames are created using the data.frame() function.



**Conclusion:**

In this practical we have learned about basic of R language and its implementation using R GUI.

**PRACTICAL 3**

**AIM: Write a program to Perform various steps of preprocessing on the given relational database/ warehouse/ files. (Data Preparator)**

**S/W:** **Data Preparator**.

**H/W: --**

**Theory:**

**Data Preparator**

* DataPreparator is a free software tool designed to assist with common tasks of [data preparation](http://www.datapreparator.com/what_is_data_preparation.html) (or data pre-processing) in data analysis and data mining.

**DataPreparator provides:**

* A variety of techniques for data cleaning, transformation, and exploration
* Chaining of pre-processing operators into a flow graph (operator tree)
* Handling of large volumes of data (since data sets are not stored in the computer memory)
* Stand-alone tool independent of any other tools
* User friendly graphical user interface

**Features**

* Data access from text files, relational databases, and Excel workbooks
* Handling of large volumes of data (since data sets are not stored in the computer memory, with the exception of Excel workbooks and result sets of some databases where database drivers do not support data streaming)
* Stand-alone tool, independent of any other tools
* User friendly graphical user interface
* Operator chaining to create sequences of pre-processing transformations (operator tree)
* Creating of model tree for test/execution data

**Data Cleaning**

Data cleaning facilities include character removal, text replacement, and date conversion.

**Data Import/Export**

Data Preparator can be used to import data from a database and export them to a file and vice versa.

**Data Integration**

Two operators, Append and Merge, can be used to combine data from different data sources.

**Data Reduction**

Data reduction can be achieved using sampling and record selection.

**Data Transformation**

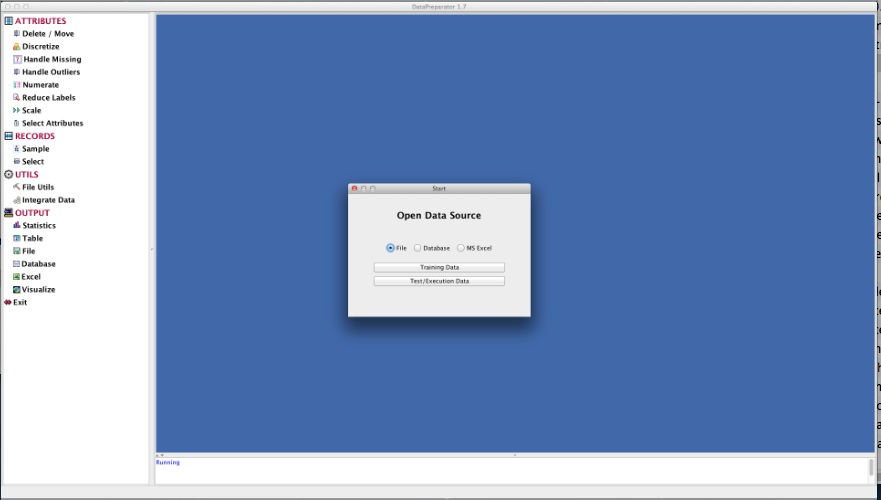
Data Preparator can be used to pre-process data for data mining. It transforms training data using a series of transformations and in the process creates a model which can be used to transform corresponding test/execution data.

Data Preparator provides many operators for transforming data.

**Data Visualization**

Data visualization can be performed using a variety of statistical plots.

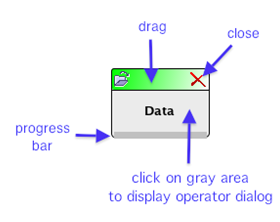
**Main Window**



The initial window. Open a data source from the Start dialog, then drag operators from the list on the left to the panel on the right.

**Operator Tree**

An operator tree is a tree of operators (preprocessing transformations) that are to be applied to the data. The nodes of the tree represent the operators and the links between the nodes show dependencies between the operators. The root of the tree --- the *Data* node --- is created automatically after opening a data source. With each node is associated an operator dialog which is displayed when the user clicks on the gray area of the node. Operators are initialized by entering required details into operator dialogs.



## Creating Nodes

To create a new node, drag an operator name from the list of names on the left hand side of the main window and drop it on the display pane on the right hand side

## Connecting Nodes

To link two nodes by an arrow, press the mouse button on the gray area of the first node and drag the mouse to the second node. Release the mouse button on the gray area of the second node.

## Moving Nodes

To move a node to a different location on the display pane, press the mouse button on the coloured bar at the top of the node and drag the node to the desired location.

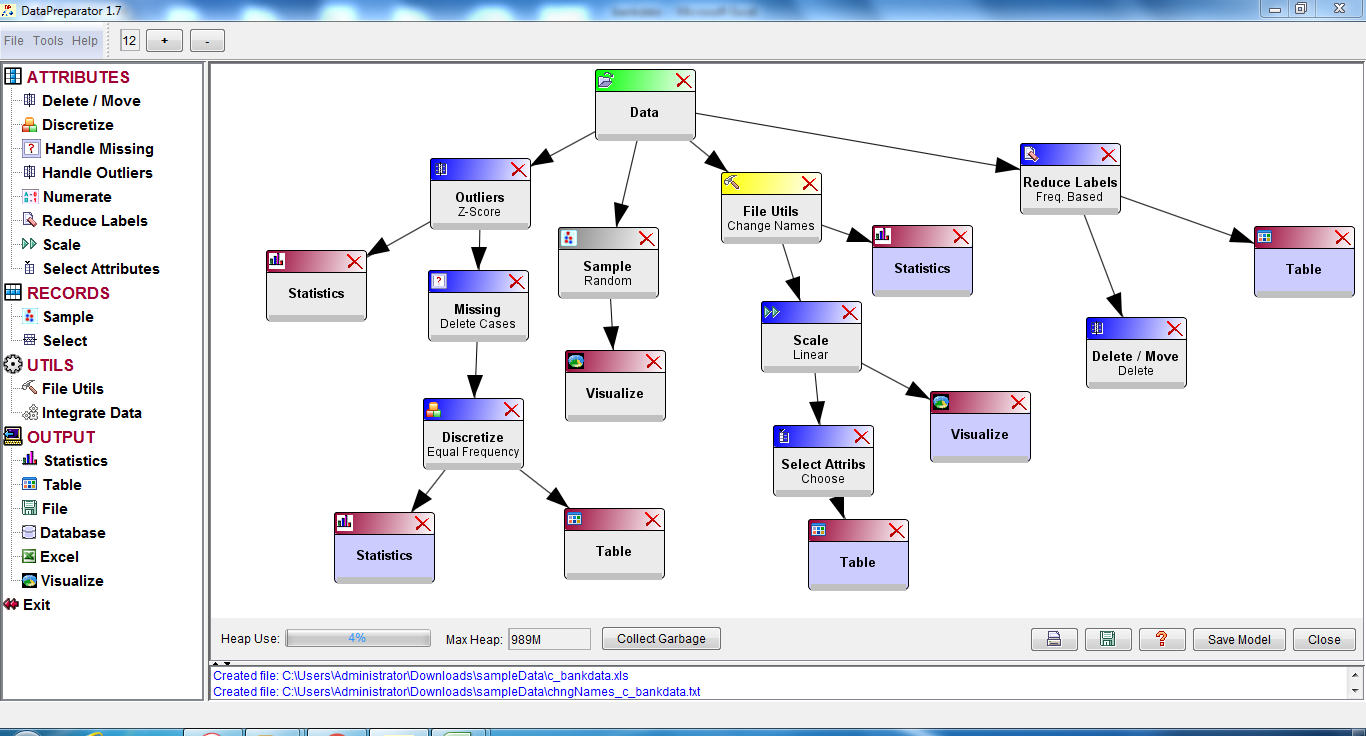
## Displaying Operator Dialog

After creating a new node, the node color is purple. Click on the purple area to display the associated operator dialog and enter the desired values. Press OK, Execute or Close as appropriate. This marks the node as processed and changes its color to gray. Clicking on the gray area displays the dialog.

## Types of Nodes

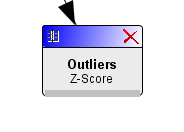
There are five types of nodes, distinguished by the colour of the bar at the top of the node icon.

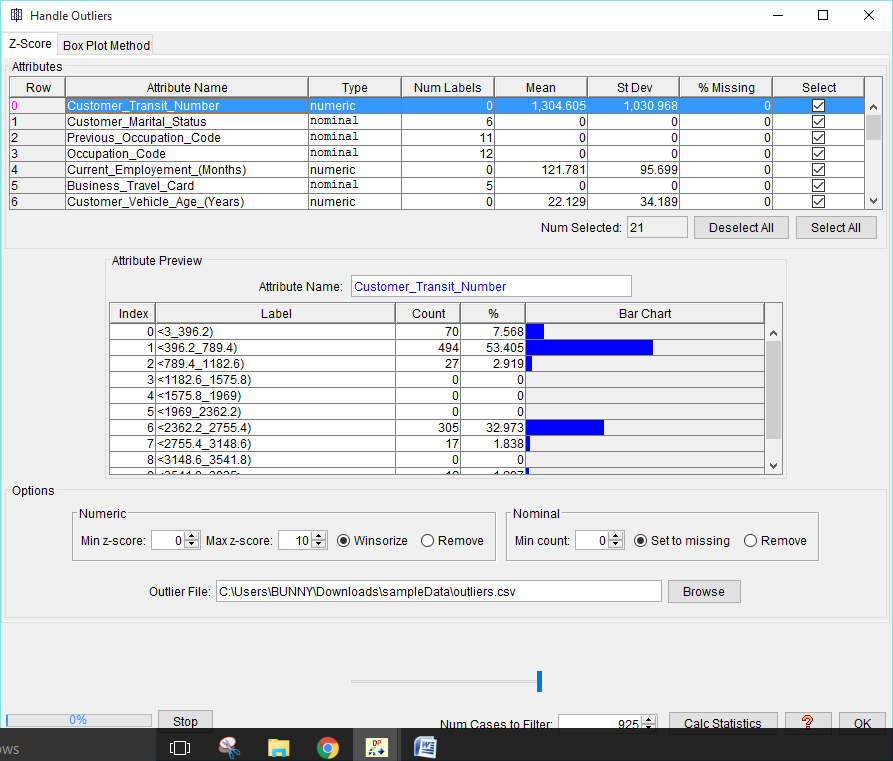
* **Green**node is the Data node. It is the root of the operator tree. There can be only one green node.
* **Blue** nodes are preprocessing nodes that will be included in the corresponding [Model Tree](about:blankmodelTree.html). They represent the transformations that will also be performed on the test or execution data sets.
* **Red**nodes are output nodes that display or save results. They cannot have descendants.
* **Yellow** nodes are file utils nodes which can only have the Data node or another File Utils node as the parent node. However, they can have other nodes as descendants.



**1.Attribute Operator**

**1) Outlier**

****



# Z-Score Method

This operator uses the Z-Score method to handle outliers in numeric attributes, and a frequency based approach to handle outliers in nominal attributes.

## Numeric Attributes

The Z-Score method uses the zscore statistic defined as:

zscore = (value - mean) / standard deviation

It gives the number of standard deviations a value is above or below the mean. An outlier is a value that has zscore above a specified upper limit or below a specified lower limit.

There are two options for dealing with outliers:

1. Winsorize (replace outliers with the values corresponding to the specified zscore limits).
2. Remove the records containing outliers from the data set.

## Nominal Attributes

In a nominal attribute, a value (label) that has a very low frequency of occurrence is considered to be an outlier. There are two options for dealing with outliers:

1. Replace outliers by the missing value symbol.
2. Remove the records containing outliers from the data set.

## Saving Outliers to a File

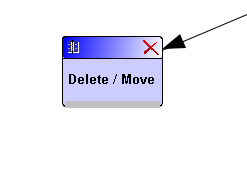
The operator also saves records containing outliers to a file. The outlier file is created when data are filtered through a path containing the Outliers node. The filtering process must be initiated from a descendant node. For example, a Statistics node below the Outliers node.

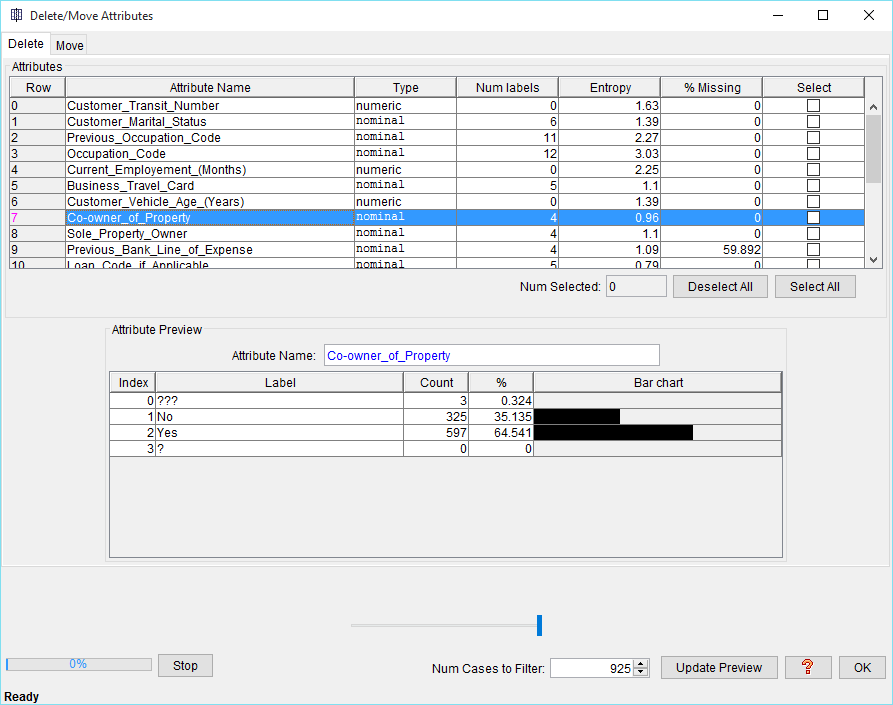
## Using the Operator

1. Select the attributes for which to handle outliers by checking check boxes in the Select column.
2. Select options for numeric attributes.
   * Enter the minimum and maximum z-scores.
   * Specify how to handle outliers in numeric attributes (winsorize or remove from the data set).
3. Select options for nominal attributes.
   * Set the minimum frequency for the values of nominal attributes in the Min count spinner.

**02) Delete/Move Attributes**

This option allows the user to remove attributes from the data or to move selected attributes either to the leftmost or to the rightmost position in the data set.



* 

**3). Discretize Numeric Attributes**

Discretization transforms numeric (continuous) attributes to nominal (categorical or discrete) attributes.

The range of a numeric attribute is divided into intervals and each interval is given a label. Attribute values are replaced by the labels of the intervals into which they fall.

The following discretization methods are currently available:

1. [Equal Width Discretization](about:blankeqWidthDiscr.html)
2. [Equal Frequency Discretization](about:blankeqFreqDiscr.html)
3. [Equal Frequency Discretization from Grouped Data](about:blankeqFreqDiscrGD.html)
4. [Defined Cut Points](about:blankdefCutPoints.html)

# Equal Width Discretization

Equal width discretization divides the range of a numeric attribute into a specified number of intervals of equal width.

Interval width is computed by dividing the attribute range by the number of interval

# Equal Frequency Discretization

Equal frequency discretization divides the range of a numeric attribute into a given number of intervals containing equal (or nearly equal) number of values.

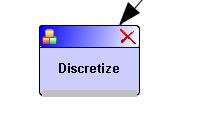
# Equal Frequency Discretization from Grouped Data

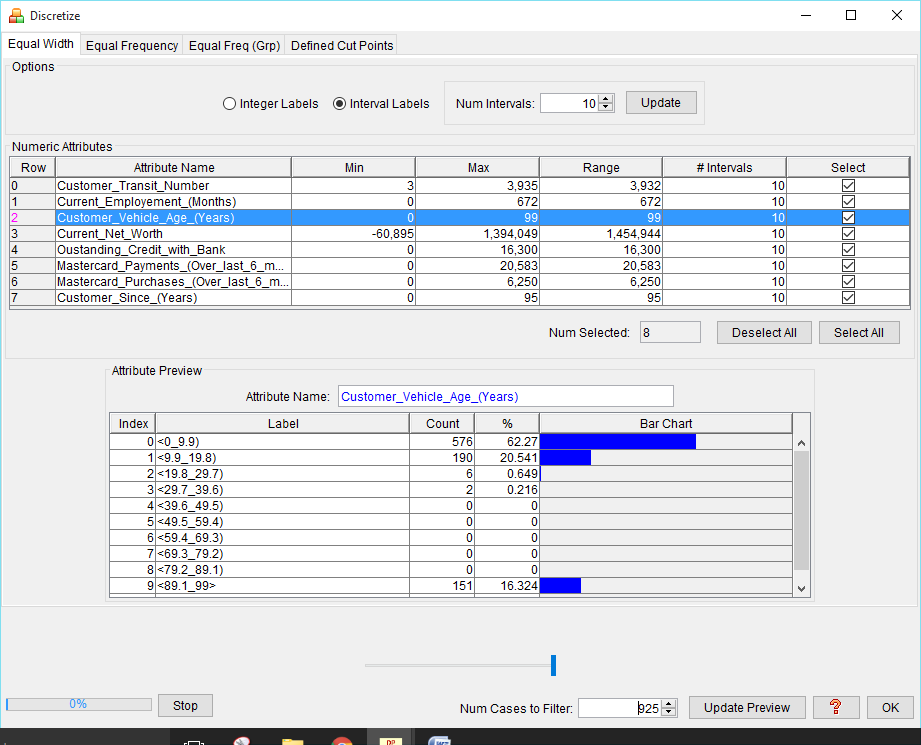
This operator creates a specified number of approximate equal-frequency intervals. It computes interval cut points by interpolating grouped data in frequency histograms. This is a standard statistical technique for computing quantiles from grouped data. Here the intervals are equivalent to quantiles.

The main advantage of this method is that it does not require sorting of attributes. The disadvantage is that the resulting intervals are only approximate.

# Defined Cut Points

This operator allows you to discretize numeric attributes manually by entering the desired cut points.





# 4) Handle Missing Values

This command provides operators for handling missing values. Currently the following methods are available:

1. [Delete Cases](about:blankdelRecords.html)
2. [Remove Attributes](about:blankremoveAttribs.html)
3. [Impute Values](about:blankimputeValues.html)
4. [Predict from Model](about:blankpredictFromModel.html)
5. [Create Missing Value Patterns](about:blankcreateMVPatterns.html)

# Delete Cases

This method removes cases containing missing values from the data set. This is a commonly used approach referred to as listwise deletion or casewise deletion.

1. **Remove Attributes**

This operator removes attributes containing missing values from the data set.

# Impute Values

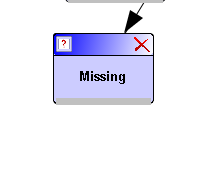
This operator replaces missing values with imputed values. It uses single-value imputations where all missing values in an attribute are filled with the same imputed value. The problem with this approach is that it can lead to bias. Commonly used imputations are the attribute mean or median for numeric attributes, and the mode for nominal attributes.

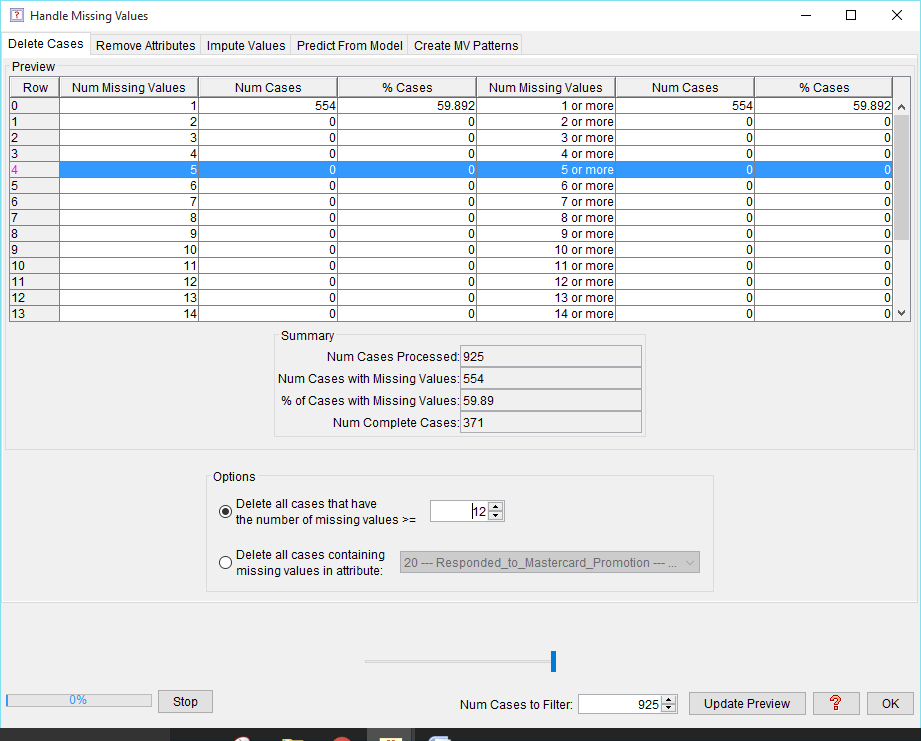
# Predict From Model

This operator replaces missing values with values predicted by a prediction model.

# Create Missing Value Patterns

This operator adds new attributes (missing value patterns) to the data set. It creates a new two-valued (dichotomous) variable for each selected attribute containing missing values. The values of the new variable represent two possible states: "value is present" and "value is missing".



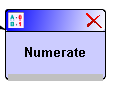


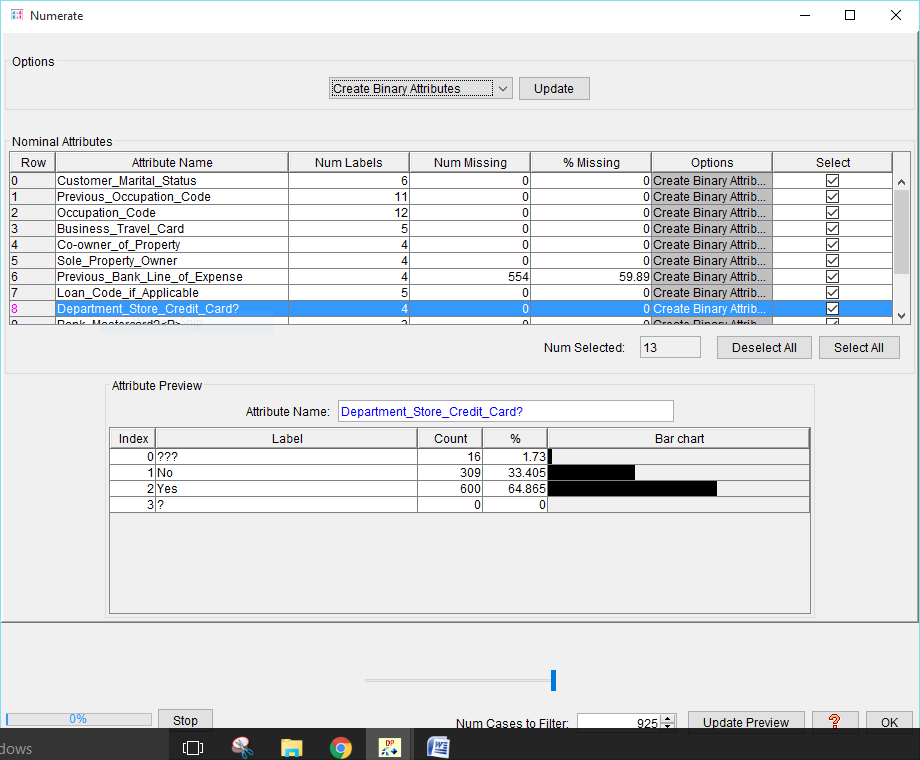
**5. Numerate Nominal Attributes**

This operator transforms nominal (categorical) attributes to numeric attributes.

Two methods are provided:

1. Create Binary Attributes
2. Replace Labels by Label Indices .



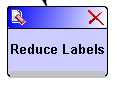


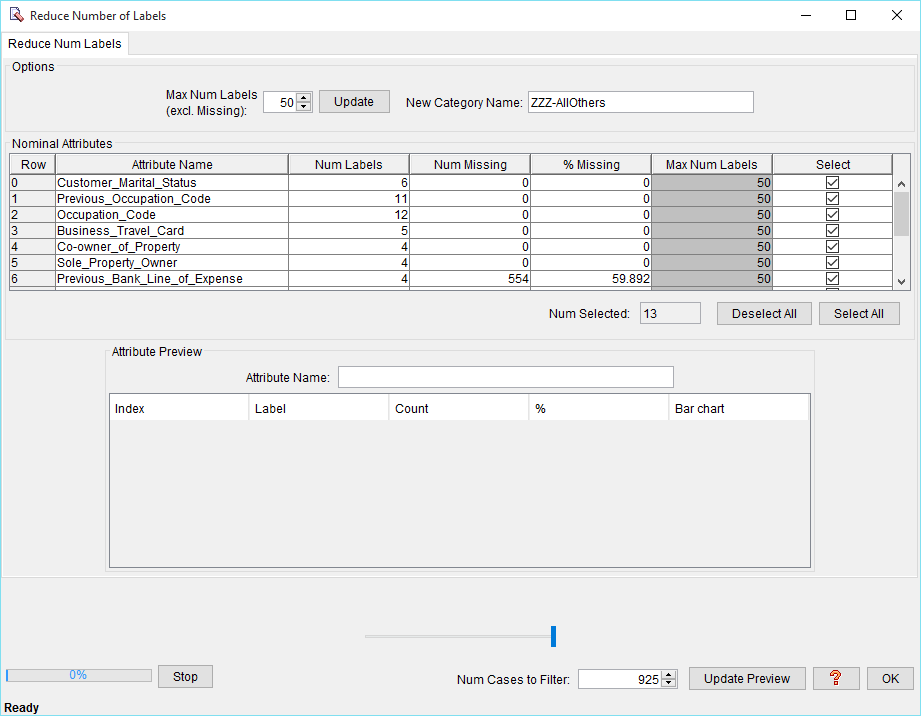
# 6. Reduce Number of Labels

Some data sets contain nominal (categorical) attributes with large number of distinct values (labels or categories). It may be necessary to reduce the number of labels for the reasons of computational efficiency.

This operator reduces the number of labels of nominal attributes by keeping up to a given number of most frequent labels and creating a new label from all the remaining labels.

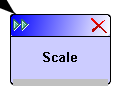
If a nominal attribute has more labels than the specified maximum number *m*, then the first *m-1* labels with the largest frequencies will be retained and one new label will be created out of all the remaining labels.

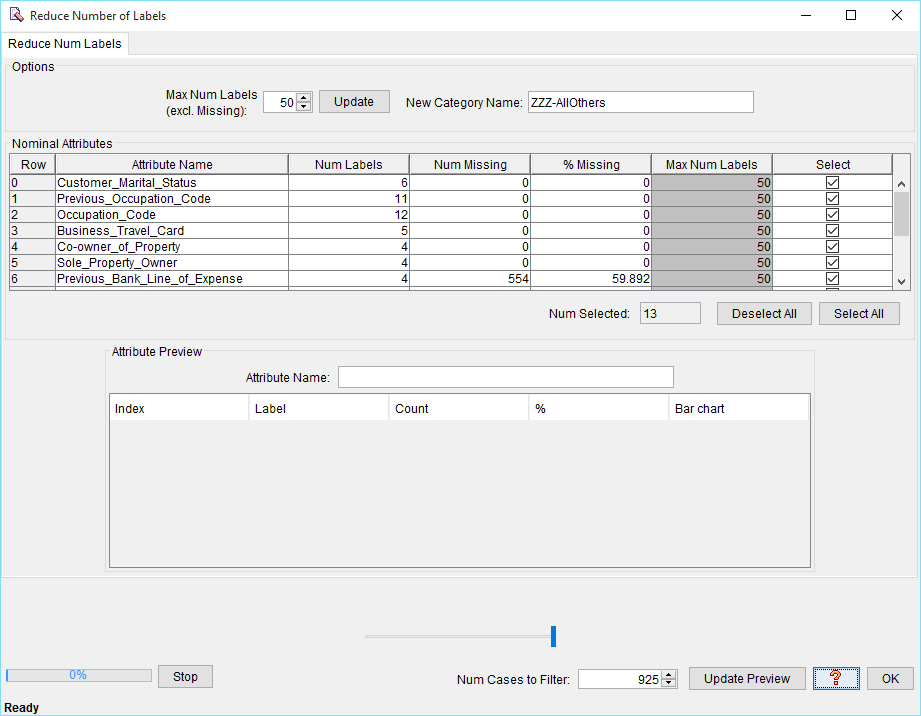




**7. Scale Numeric Attributes** This command provides operators for scaling (or normalizing) numeric attributes. Scaling is required for data mining algorithms that accept only attribute values within certain ranges. For example, neural networks, clustering, nearest neighbor among others. Scaling is also needed to prevent bias when attributes have very different ranges (e.g., age and salary). Currently the following scaling methods are provided:

1. [Linear](about:blanklinearScaling.html)
2. [Decimal](about:blankdecimalScaling.html)
3. [Hyperbolic Tangent](about:blankhtangentScaling.html)
4. [Soft-Max](about:blanksoftmaxScaling.html)
5. [Z-Score](about:blankzscoreScaling.html)



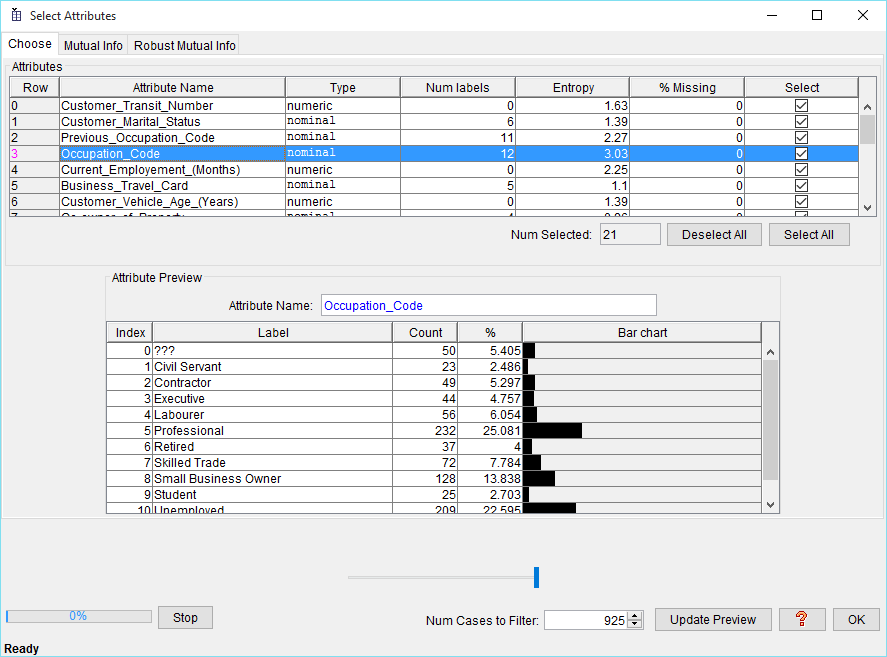


**Select Attributes**

This command selects a subset of attributes. The following methods are currently provided:

1. [Manual Selection](about:blankchooseAttributes.html)
2. [Mutual Information Selection](about:blankmutualInfoSelection.html)
3. [Robust Mutual Information Selection](about:blankrobustMutualInfo.html)





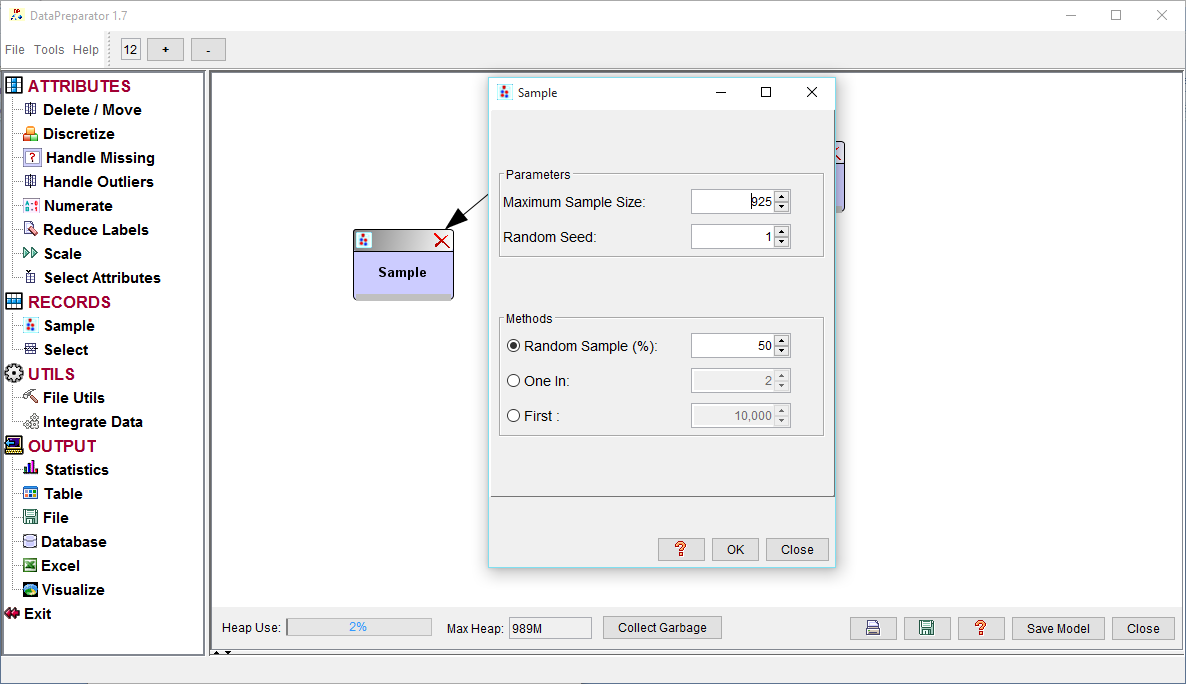
**2. Record operator**

**1. Sample**

This operator creates a sample from the data set. The following sampling methods are provided:

1. Random
2. One in K
3. First K

Random sampling selects cases at random according to a given percentage. One-in-K sampling selects every K-th case. First-K sampling selects the first K cases from the data set.



# 2. Select Records

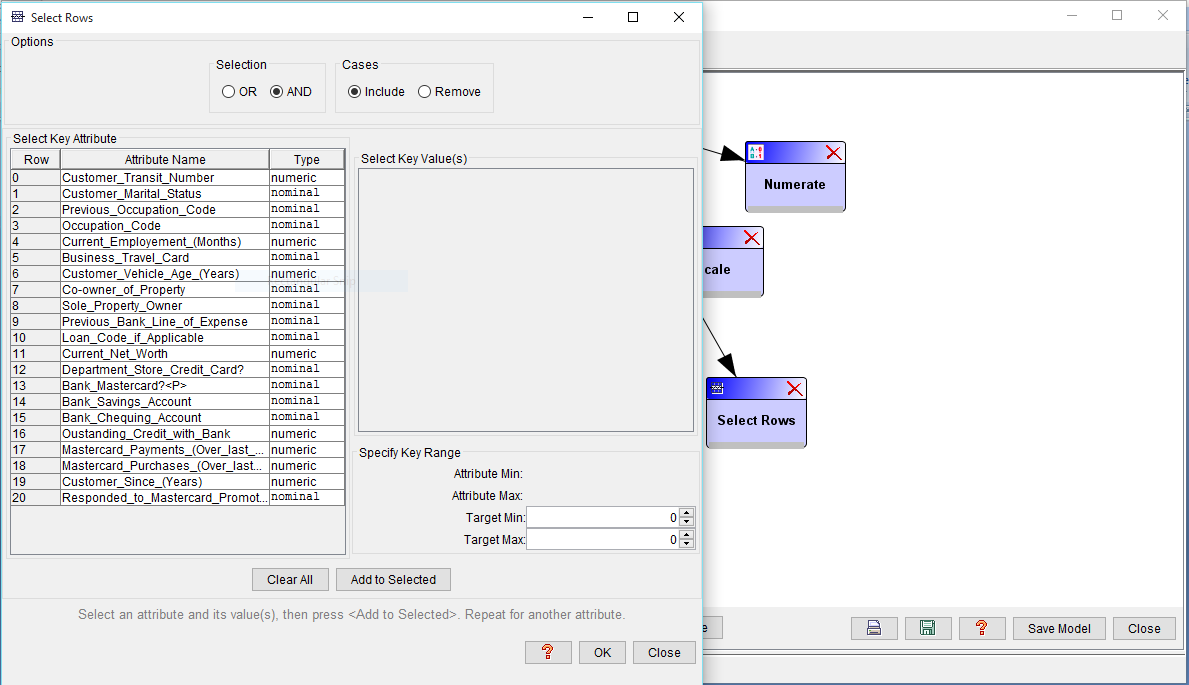
This operator selects records (cases) from the data set, based on a specified key and key values.

## Using the Operator

1. Select either AND or OR radio button.
   * If the AND radio button is selected then a case will be selected only if it contains the selected key values for all the selected keys.
   * If the OR radio button is selected then a case will be selected when it contains the selected key values of at least one selected key.
2. Specify whether the operation is Include or Remove.
   * If the Include radio button is selected then all the cases satisfying the selection criteria will be included in the resulting data set.
   * If the Remove radio button is selected then all the cases satisfying the selection criteria will be removed from the resulting data set.
3. Select a key (key attribute) from the list on the left.
4. Select key values.
   * For a nominal key attribute, select one or more key values from the list on the right. (Press ctrl key and click appropriate rows, then click Add to Selected).
   * For a numeric key attribute, specify the range of values to be selected by entering the minimum value in Target Min and the maximum value in Target Max spinners.
5. Press Add to Selected.

Repeat the steps 3, 4 and 5 above for other key attributes as needed.

1. Click OK.



**3.Utils**

**1. File utils**

**1.1Create Data Sets**

This operator partitions the data set into three files:

* Training File
* Test File
* Validation File

# 1.2 Create Missing Values

This operator creates a file containing missing values.

This may be useful when experimenting with algorithms for handling missing values.

**1.3 Append**

The *Append* operator can be used to append cases from a file, a database table or an Excel worksheet to the end of the current data set.

The appended rows and the current data set must have the same number of attributes and the corresponding attributes must be of the same type.

**1.4 Balance**

This operator creates a new file in which the labels of a selected nominal attribute (balancing attribute) have approximately equal frequencies.

Two techniques are provided:

1. Under sampling of Majority Classes
2. Oversampling of Minority Classes

**1.5 Merge**

The *Merge* operator merges a sorted data set with another sorted data set into a single file.

# 1.6 Sort

This operator sorts the data set in ascending or descending order of a specified key attribute and creates a sorted file.

# 1.7 Partition Cases

The Partition operator splits a file into multiple files.

There are three options:

1. [Split File into a Specified Number of Files](about:blanksplitRows-1.html)
2. [Split File by Attribute Values](about:blanksplitRows-2.html)
3. [Split File into 2 Files by Row Number](about:blanksplitRows-3.html)

# 1.8 Add Columns

The Add Columns operator adds (appends) columns from a file, a database table or an Excel worksheet to the right of the rightmost column of the current file. The number of rows in the resulting file is equal to the number of rows in the smaller file.

# 1.9 Change Names

This operator changes the names (identifiers) of selected attributes and/or attribute values (labels). It creates a new file containing the changed identifiers.

# 1.10 Encode

This operator encodes data using phonetic algorithms.

A phonetic algorithm encodes words on the basis of their pronunciation. Similarly sounding words will have similar code.

Two phonetic algorithms are included:

* [Soundex Algorithm](about:blanksoundexEncoding.html)
* [Metaphone Algorithm](about:blankmetaphoneEncoding.html)

**1.11 Join**

* The Join operator joins two files by a common key attribute.
* The rows of the resulting file are created from the rows of two files by joining the rows that have matching values of the key attribute. Both files must be sorted in ascending order of the key attribute.

**1.12 Smooth Columns**

The Smooth Columns operator reduces the number of distinct values of selected numeric attributes by replacing the original values with estimated values.

The following smoothing methods are provided:

1. Bin Average
2. Bin Boundaries
3. Bin Midpoint
4. Rounding

# 1.13 Split Columns

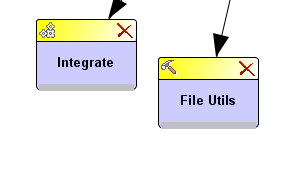
The Split Columns operator splits a file into two files by a specified attribute (column).

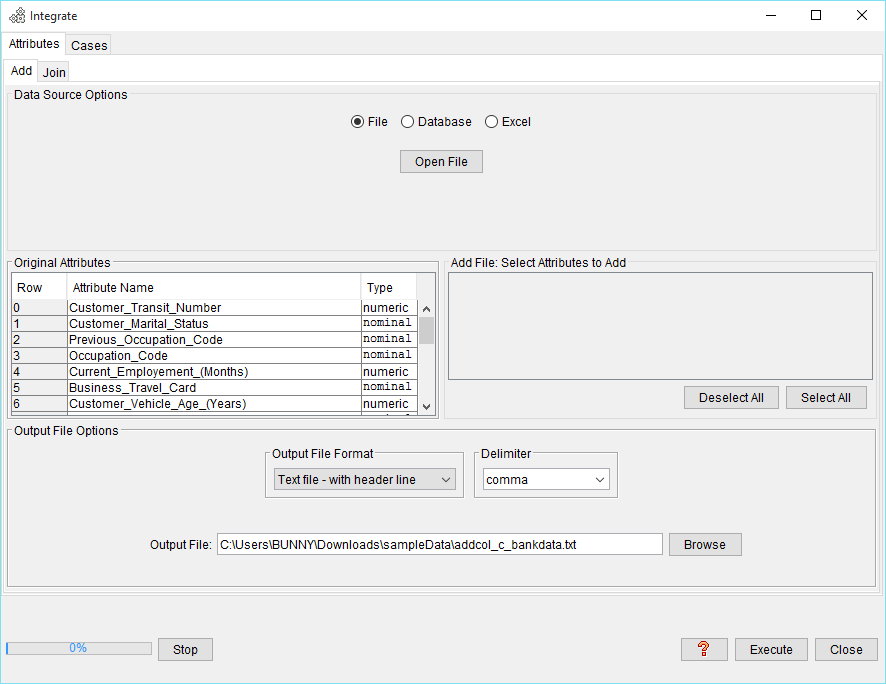
The columns with indices less than the specified column index are written to one file and the remaining columns to another file.

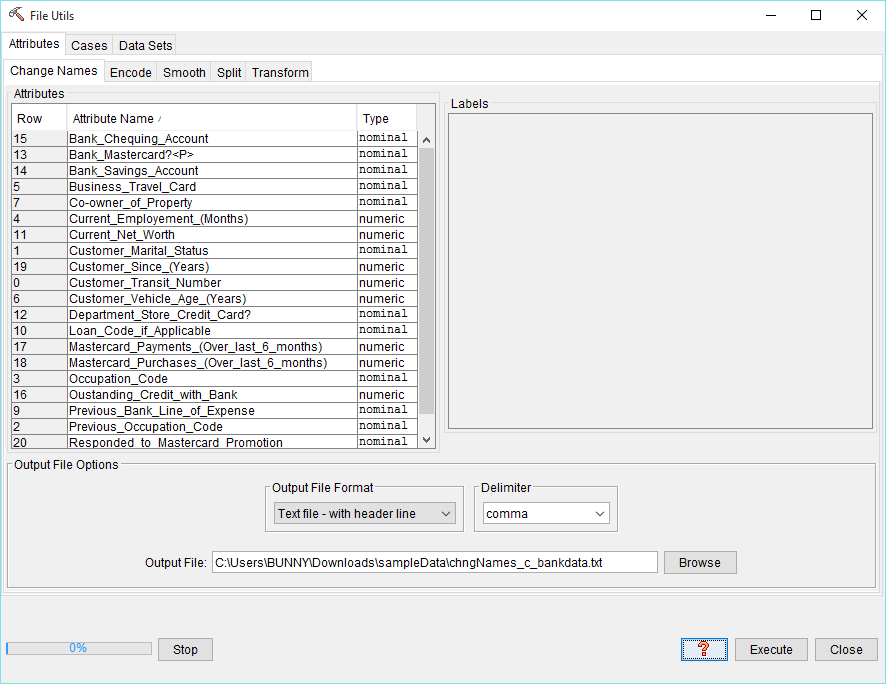
# 1.14 Transform Attribute Values

This operator transforms attribute values using several common mathematical functions:

ln(x), log2(x), log10(x), exp(x), sqrt(x), sin(x), cos(x), tan(x), 1/x, x2, x3







**4. Output**

**1. Statistics**

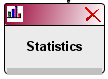
This operator displays statistical information for the attributes in the data set.

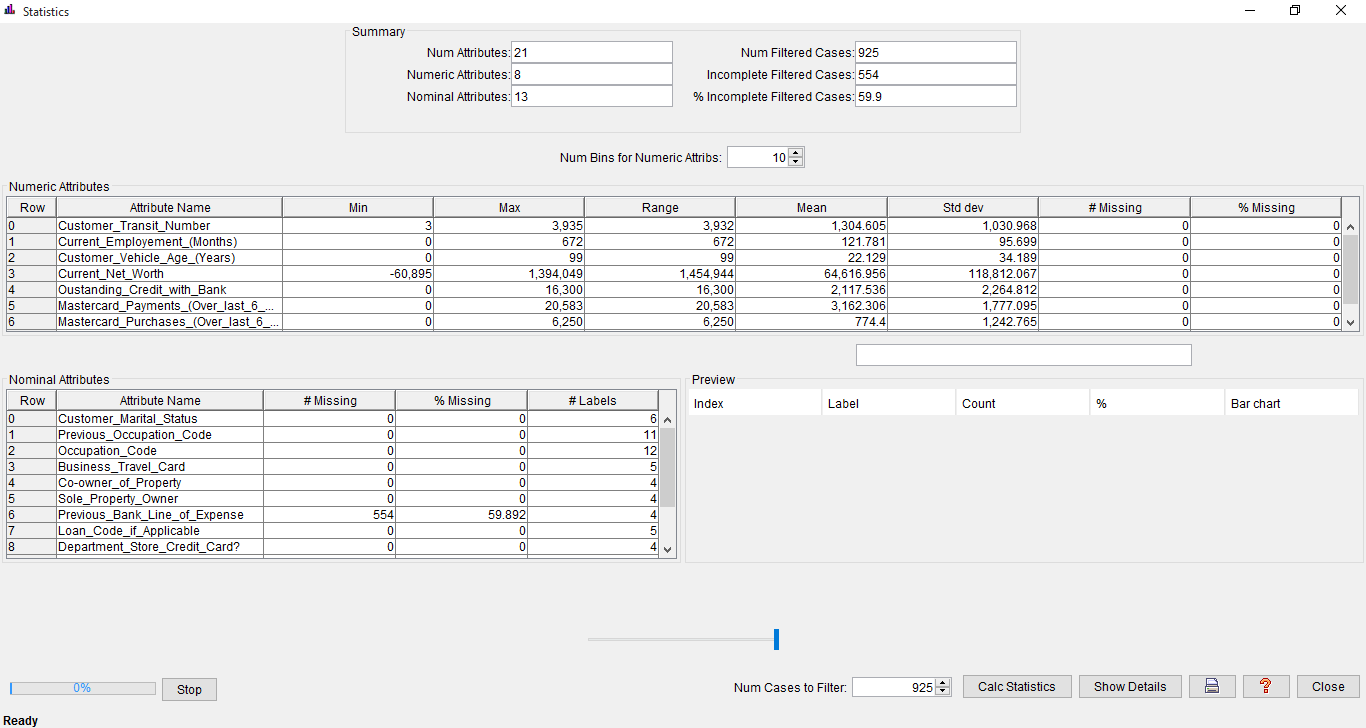
For numeric attributes it shows:

* Minimum
* Maximum
* Range
* Mean
* Standard deviation
* Number of missing values
* Percentage of missing values

For nominal attributes it shows:

* Number of missing values
* Percentage of missing values
* Number of labels

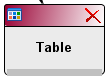


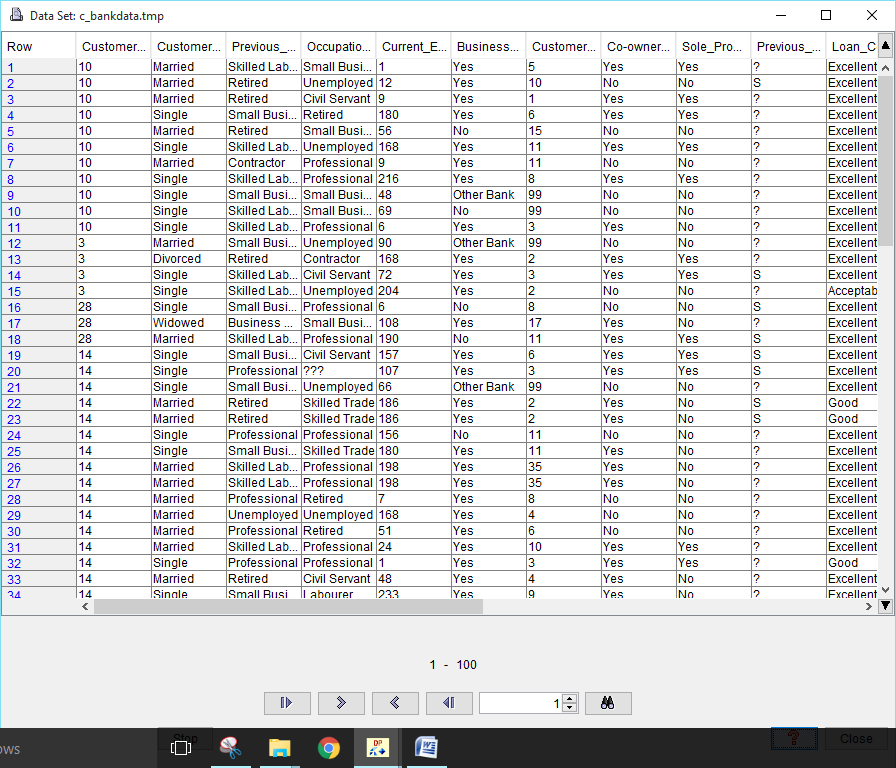


**2. Table**

This operator displays data in a two dimensional table.

It reads one page of data at a time. The page size is set to 100 lines. Initially the first page of *100* lines is loaded.

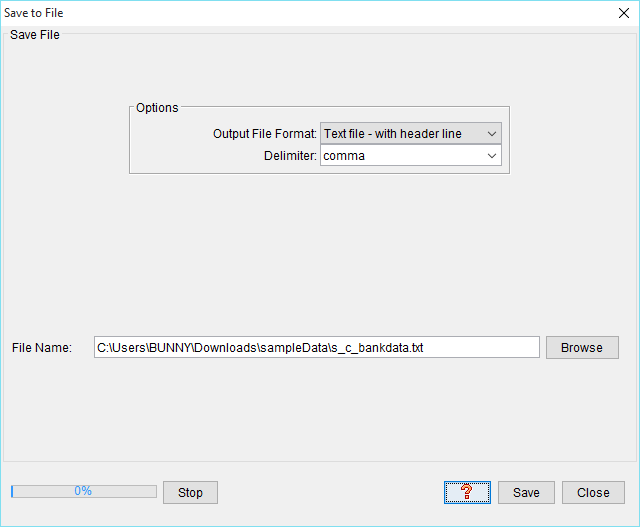




1. **File Output**

This operator saves output to a file.

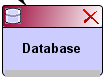


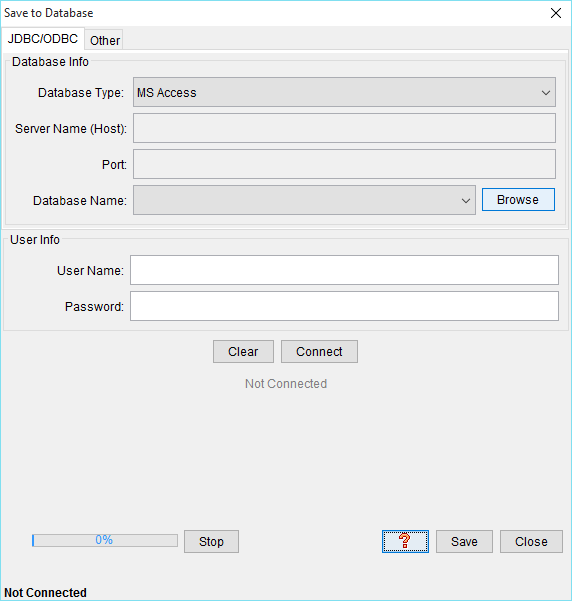


1. **Database Output**

This operator saves output to a database table. It contains two tabs:

1. JDBC/ODBC
2. Other

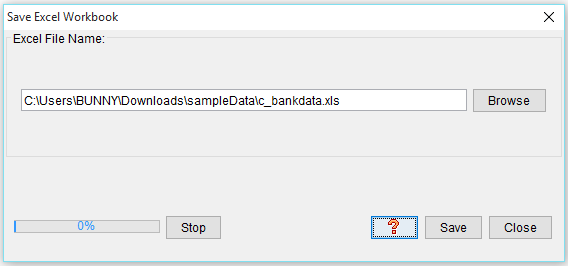




1. **Excel Output**

This operator saves output to an Excel spreadsheet (.xls file). Only up to 256 columns and 65,535 rows are allowed.





# 6. Visualize Data

## Numeric Attributes

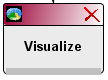
1. [Univariate Plots](about:blankvisualize/univariatePlots.html)
2. [Bivariate Plots](about:blankvisualize/bivariatePlots.html)
3. [Conditional Plots](about:blankvisualize/conditionalPlots.html)
4. [Matrix Plots](about:blankvisualize/matrixPlots.html)

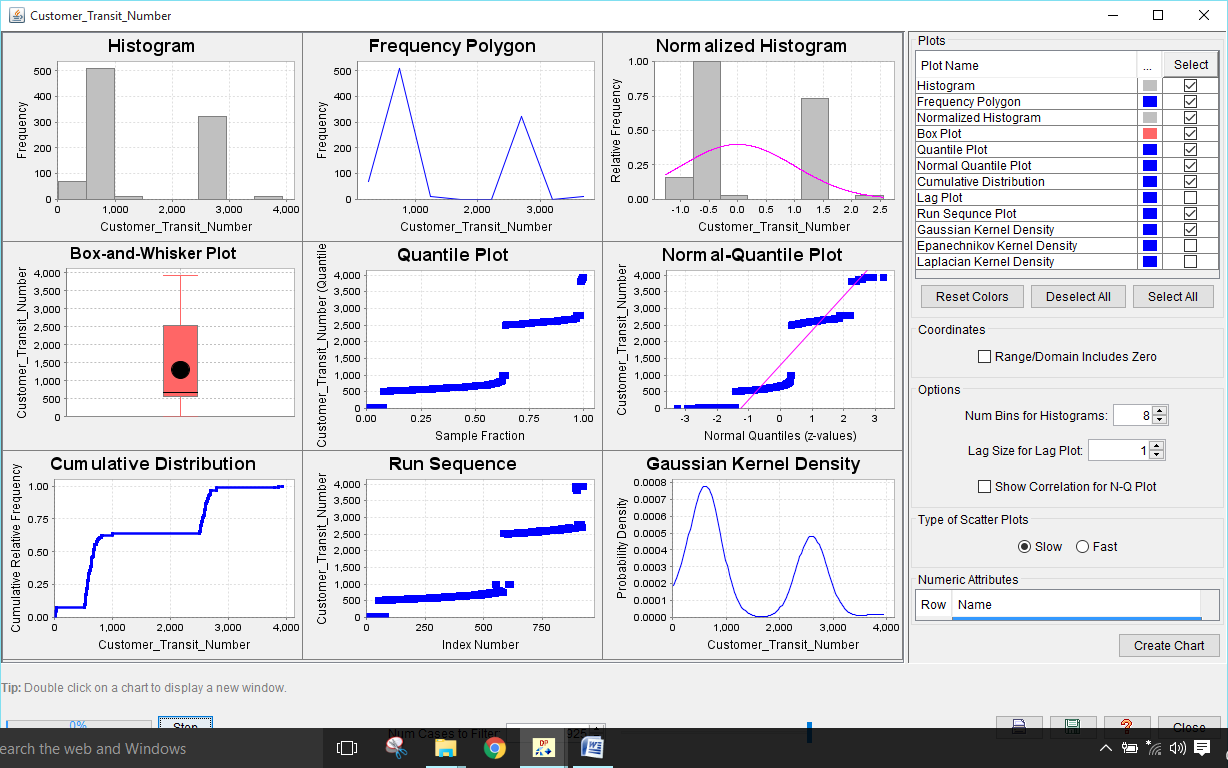
## Nominal Attributes

1. [Univariate Charts](about:blankvisualize/nominUnivariatePlots.html)
2. [Stacked Bar Charts](about:blankvisualize/nominBivariatePlots.html)

## Mixed Attributes

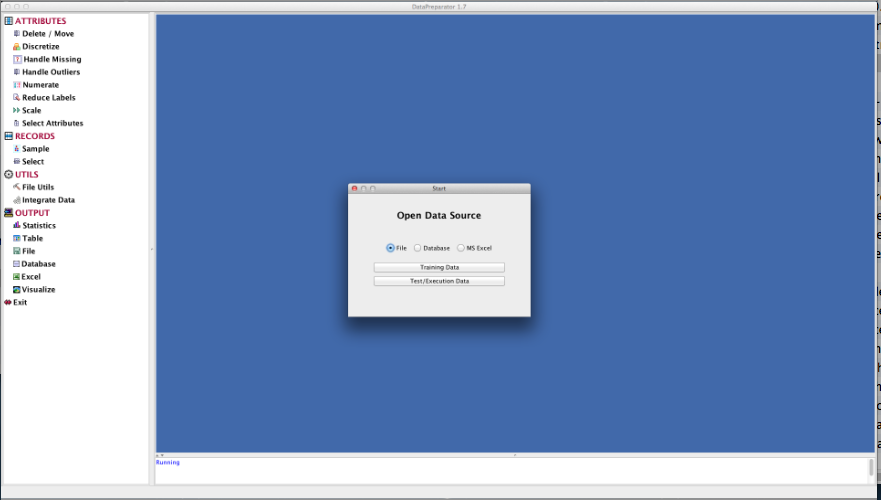
1. [Dependency Tree](about:blankvisualize/dependencyTreeChart.html)
2. [Parallel Coordinates](about:blankvisualize/parallelCoords.html)



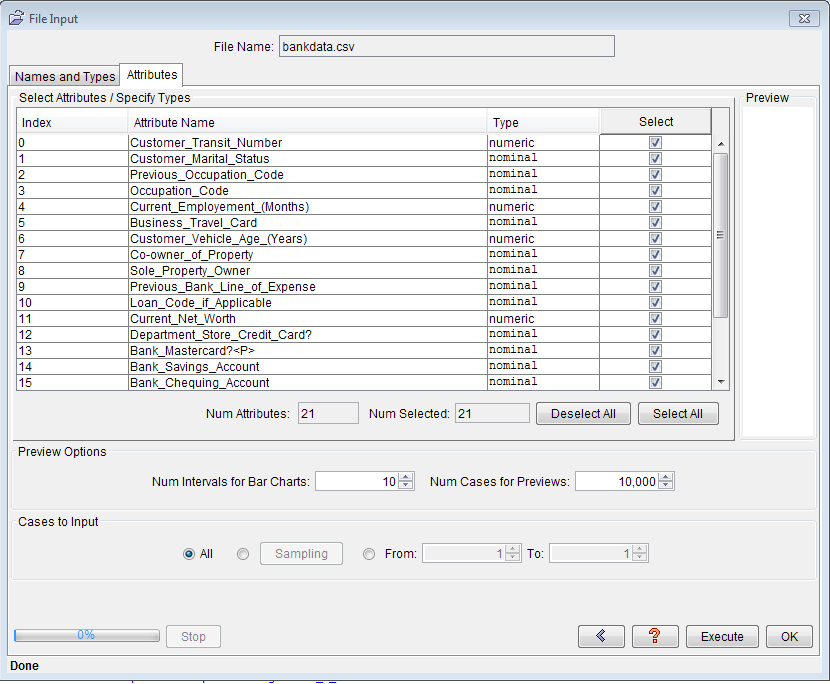


**Performed Steps:**

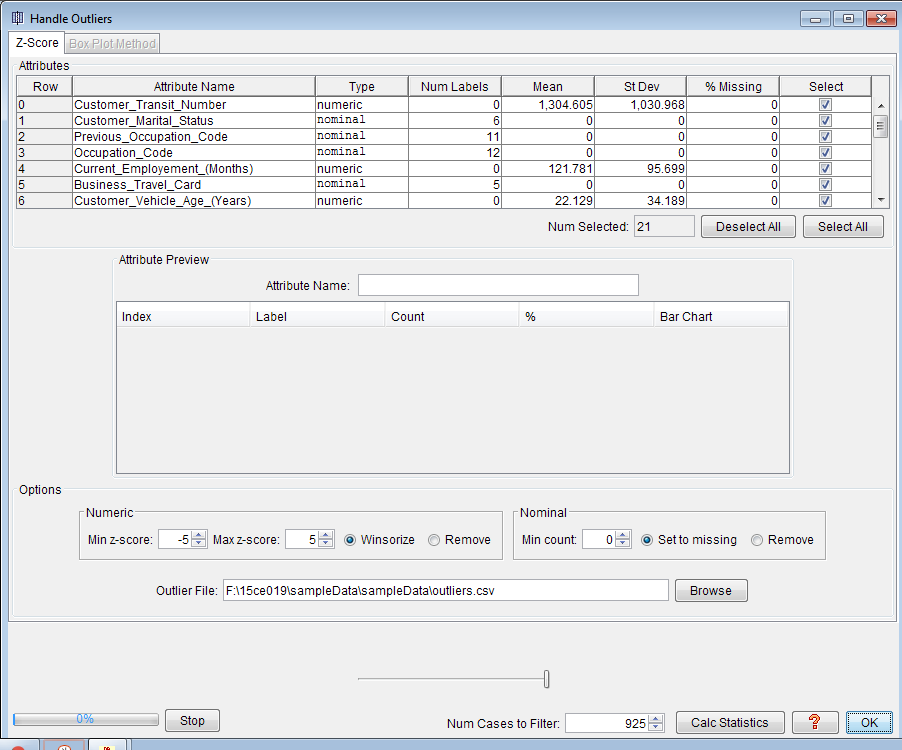
**1.) Starting screen will be look like this.**



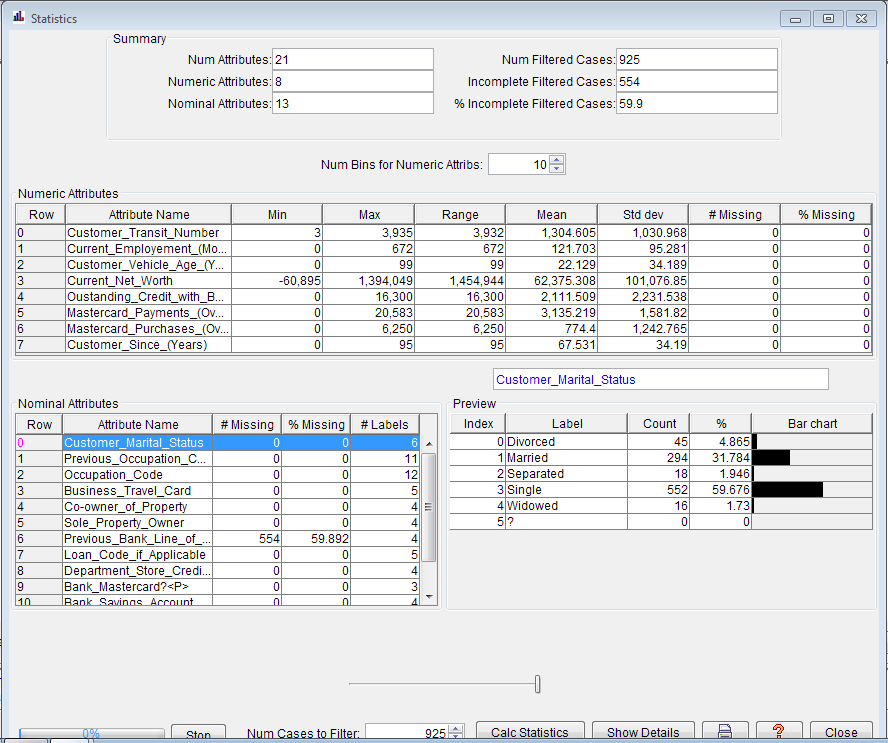
**2.) Select option ‘training data’ with file option.**

****

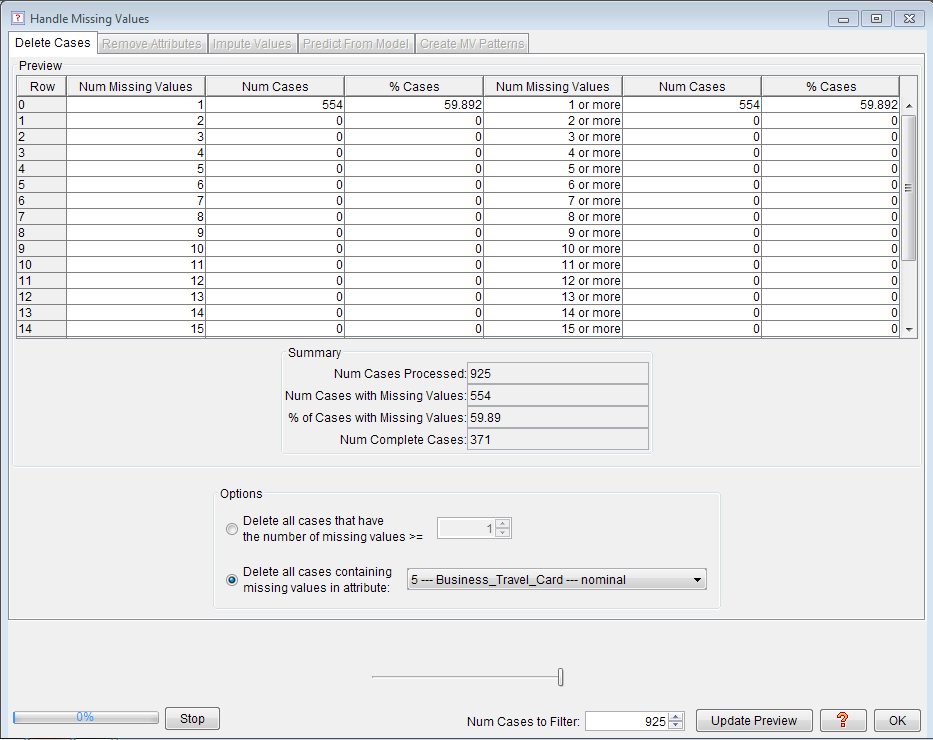
**3.) ‘Handle outliers’ option can be located at left side in the tool.**

****

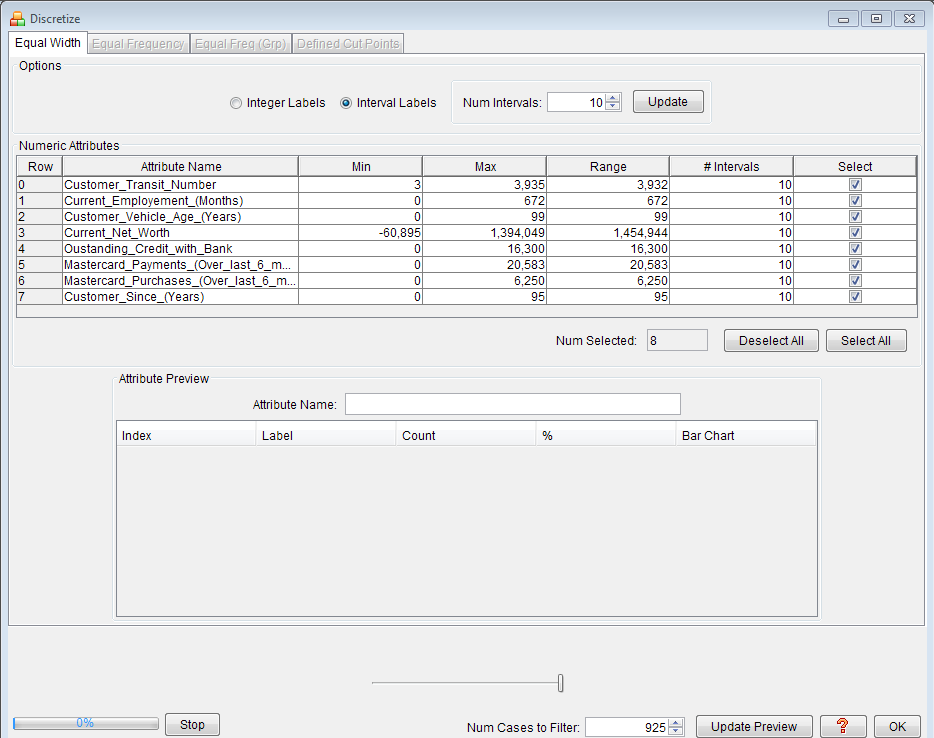
**4.) After adding statistics, this screen will be populated by clicking on the statistics.**

****

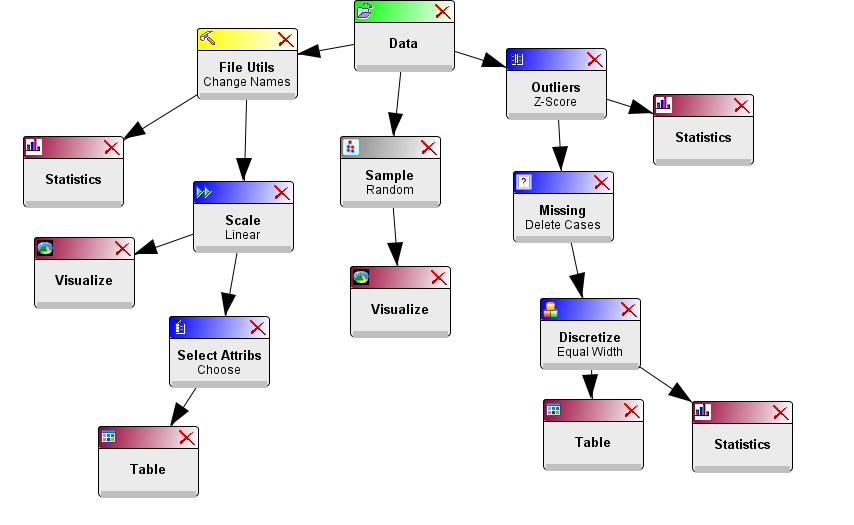
**5.) Handling missing values:**

****

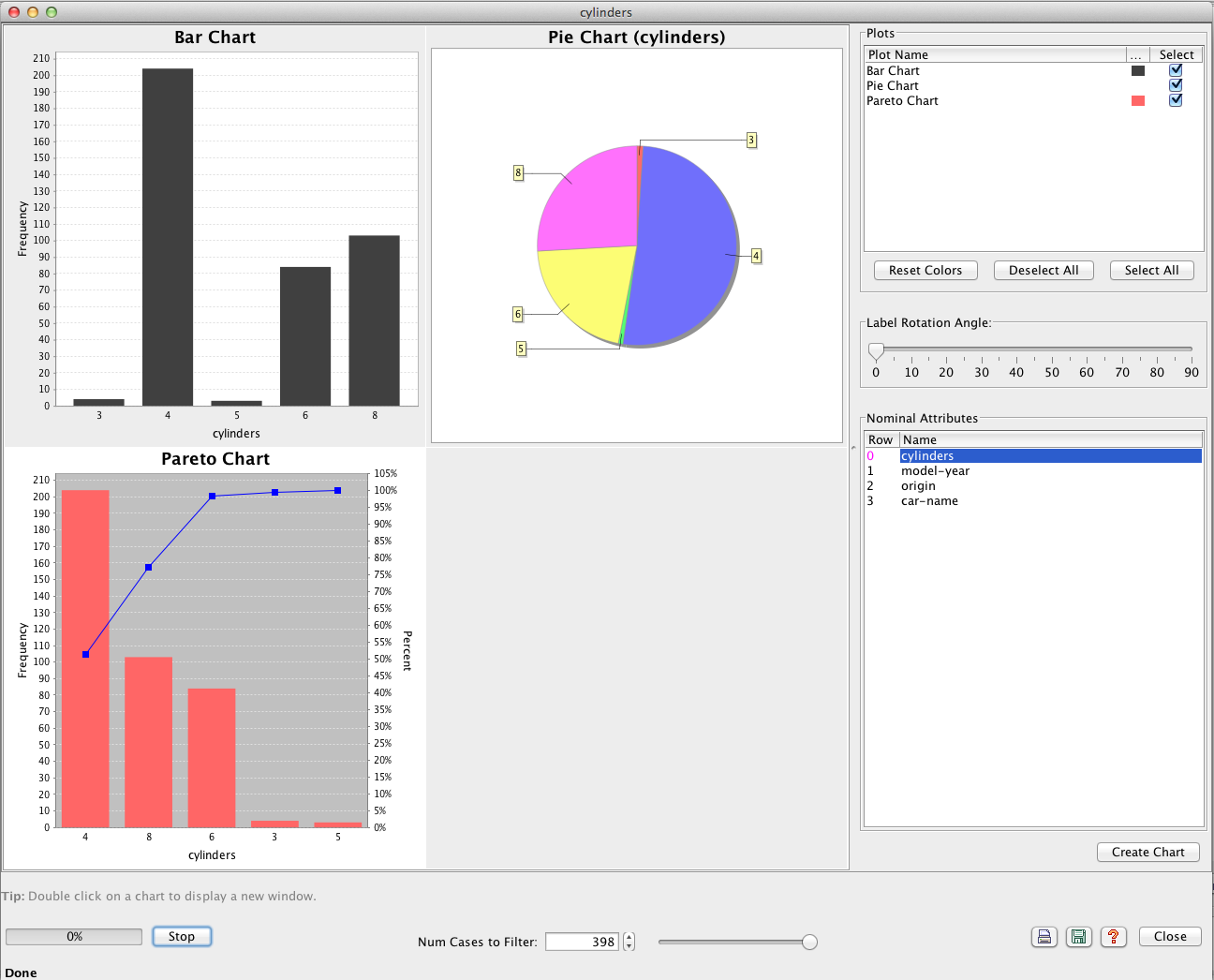
**6.) Discretize:**

****

**7.) Operator tree**

****

**8.) Various visualization options available here.**



**Conclusion:**

From this practical, we have learnt about the requirement of data preprocessing in data mining and how it can be done in “data preparator” tool.

**PRACTICAL 4**

**AIM: Describing data and its Statistical Analysis Graphically using R**

**Programming.**

**S/W: RStudio/RGUI**

**H/W: --**

**Theory:**

R Programming language has numerous libraries to create charts and graphs.

**Pie-Chart**

A pie-chart is a representation of values as slices of a circle with different colors. The slices are labeled and the numbers corresponding to each slice is also represented in the chart.

In R the pie chart is created using the **pie()** function which takes positive numbers as a vector input. The additional parameters are used to control labels, color, title etc.

Syntax: pie(x, labels, radius, main, col, clockwise)

Examples:

library("xlsx")

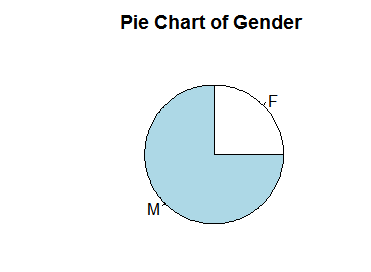
data <- read.xlsx("BatchC.xlsx" , sheetIndex = 1)

print(data)

labels <- c("F","M")

x <- table(data$Gender)

pie(x,labels)



library("xlsx")

library(plotrix)

data <- read.xlsx("BatchC.xlsx" , sheetIndex = 1)

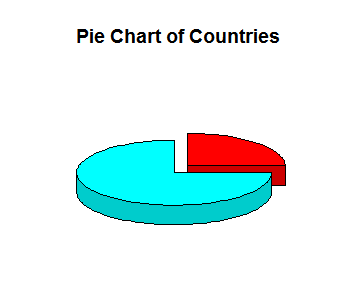
print(data)

labels <- c("F","M")

x <- table(data$Gender)

pie(x,labels)

pie3D(x,labels,explode = 0.1, main = "Pie Chart of Countries ")



**Barcharts**

A bar chart represents data in rectangular bars with length of the bar proportional to the value of the variable. R uses the function **barplot()** to create bar charts. R can draw both vertical and Horizontal bars in the bar chart. In bar chart each of the bars can be given different colors.

Syntax: barplot(H, xlab, ylab, main, names.arg, col)

Examples:

library("xlsx")

library(plotrix)

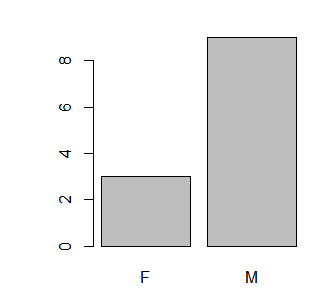
data <- read.xlsx("BatchC.xlsx" , sheetIndex = 1)

print(data)

labels <- c("F","M")

x <- table(data$Gender)

barplot(x)



**Line Graphs**

A line chart is a graph that connects a series of points by drawing line segments between them. These points are ordered in one of their coordinate (usually the x-coordinate) value. Line charts are usually used in identifying the trends in data.

The **plot()** function in R is used to create the line graph.

Syntax: plot(v, type, col, xlab, ylab)

Examples:

library("xlsx")

library(plotrix)

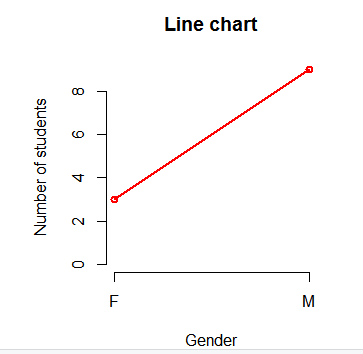
data <- read.xlsx("BatchC.xlsx" , sheetIndex = 1)

print(data)

labels <- c("F","M")

x <- table(data$Gender)

plot(x, col = “red”, xlab = “Number of students”, ylab = “Gender”, main = “Line Chart”)



**Boxplots**

Boxplots are a measure of how well distributed is the data in a data set. It divides the data set into three quartiles. This graph represents the minimum, maximum, median, first quartile and third quartile in the data set. It is also useful in comparing the distribution of data across data sets by drawing boxplots for each of them.

Boxplots are created in R by using the **boxplot()** function.

Syntax: boxplot(x, data, notch, varwidth, names, main)

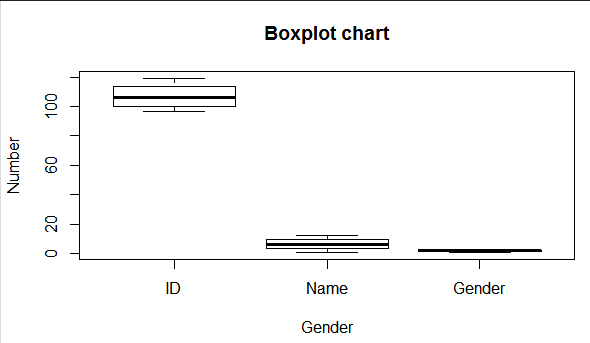
Examples:

library("xlsx")

library(plotrix)

data <- read.xlsx("BatchC.xlsx" , sheetIndex = 1)

boxplot(data, xlab = "Gender", ylab = "Number", main = "Boxplot chart")



**Histograms**

A histogram represents the frequencies of values of a variable bucketed into ranges. Histogram is similar to bar chat but the difference is it groups the values into continuous ranges. Each bar in histogram represents the height of the number of values present in that range.

R creates histogram using **hist()** function. This function takes a vector as an input and uses some more parameters to plot histograms.

Syntax: hist(v, main, xlab, xlim, ylim, breaks, col, border)

Examples:

library("xlsx")

library(plotrix)

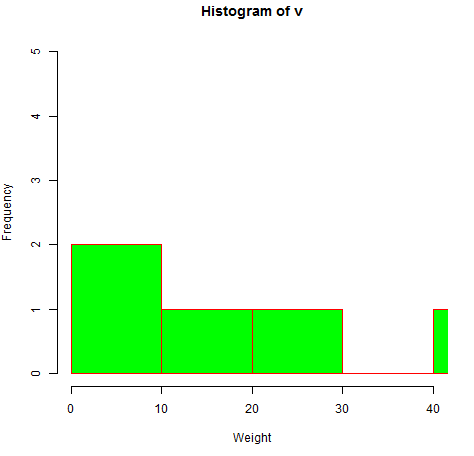
data <- read.xlsx("BatchC.xlsx" , sheetIndex = 1)

print(data)

labels <- c("F","M")

x <- table(data$Gender)

hist(x, main = “Histogram of v”, xlab = “Frequency”, ylab = “Weight”, col = “green”, border = “red”)



**Conclusion:**

From this practical we learnt that how different types are charts are formed using the data and one gather the knowledge by analyzing this charts.

**PRACTICAL 5**

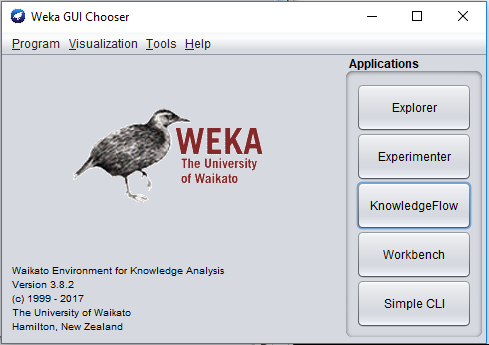
**AIM: Perform Different Data Mining Activities using Weka Explorer Tool (Open Source Data Mining Tool) &amp; Experimental Tool (Open Source Data Mining Tool).**

**S/W: Weka Tool**

**H/W: --**

**Theory:**

**WEKA TOOL**

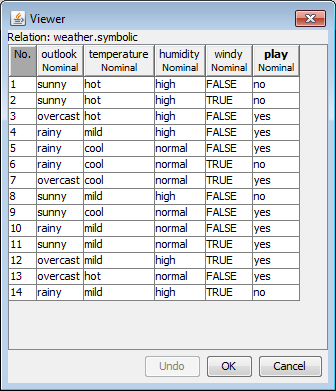


* WEKA is a state-of-the-art facility for developing machine learning (ML) techniques and their application to real-world data mining problems.
* It is a collection of machine learning algorithms for data mining tasks. The algorithms are applied directly to a dataset.
* WEKA implements algorithms for data preprocessing, classification, regression, clustering, association rules; it also includes a visualization tools.
* WEKA expects the data file to be in Attribute-Relation File Format (ARFF) file.

**Weka Options**

1. **Weka Explorer:**Weka Explorer is an environment for exploring data.
2. **Experimenter**: Experimenter is an environment for performing experiments and conducting statistical tests between learning schemes.
3. **KnowledgeFlow**: Knowledge Flow is a Java-Beans-based interface for setting up and running machine learning experiments.

**WeatherNominal.arff**



**Weka Explorer**

**Preprocess**

* Once the data is loaded, WEKA recognizes attributes that are shown in the ‘Attribute’ window. Left panel of ‘Preprocess’ window shows the list of recognized attributes:

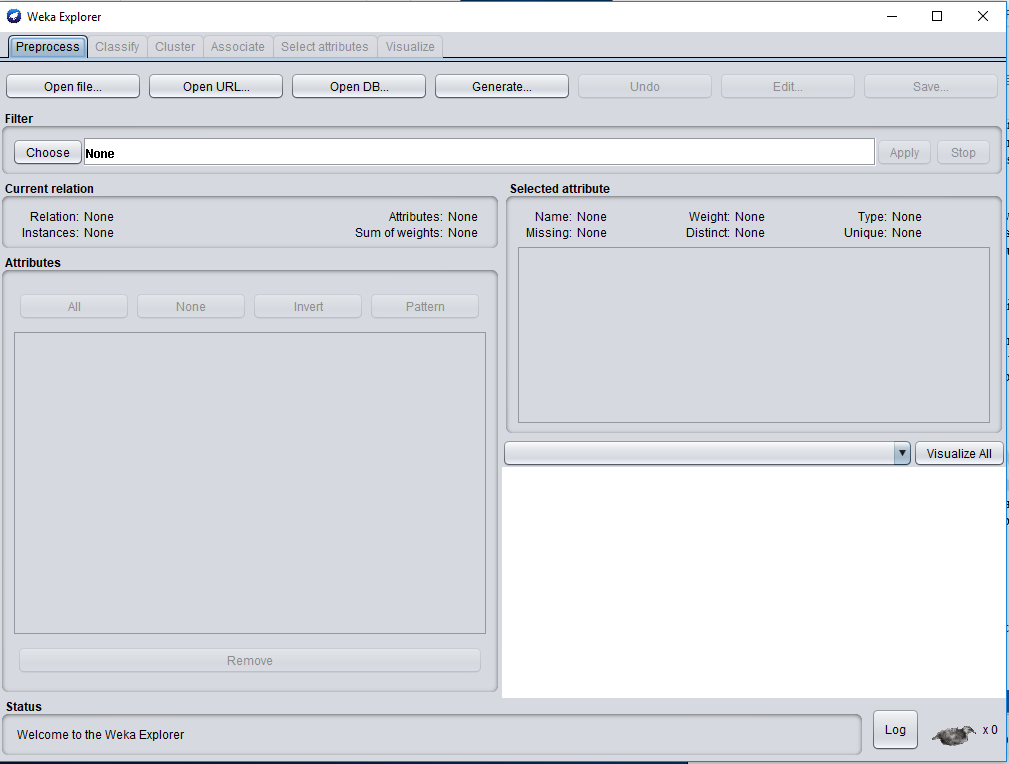
1. No. is a number that identifies the order of the attribute as they are in data file
2. Selection tick boxes allow you to select the attributes for working relation
3. Name is a name of an attribute as it was declared in the data file.

* The ‘Current relation’ box above ‘Attribute’ box displays

1. The base relation (table) name and the current working relation
2. The number of instances
3. The number of attributes

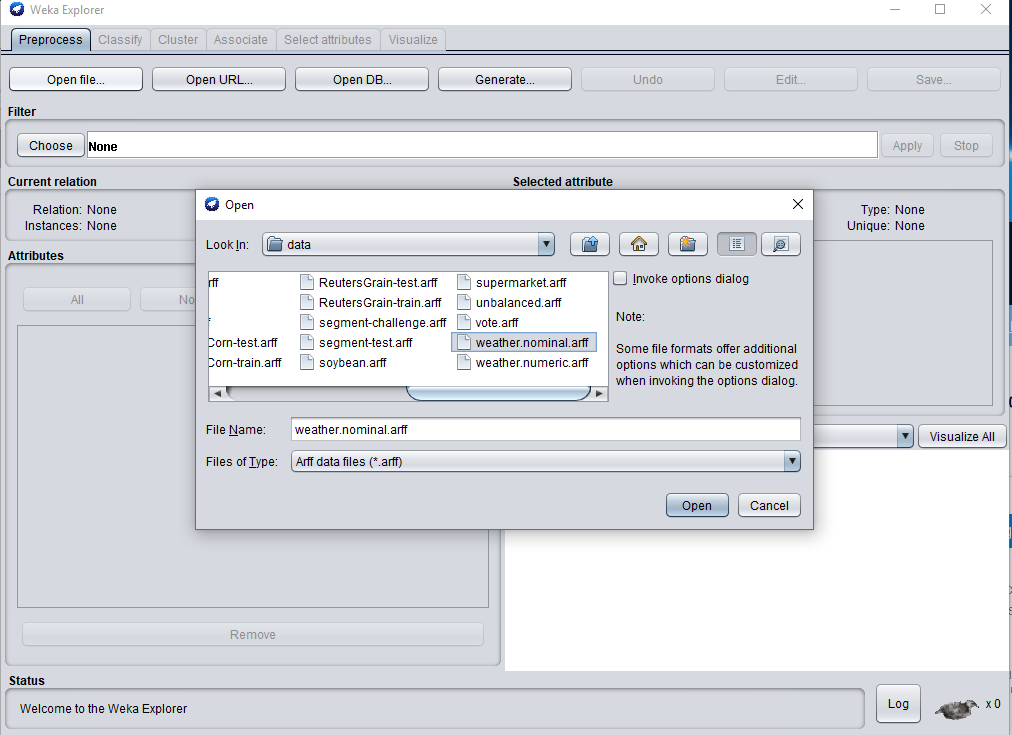
* During the scan of the data, WEKA computes some basic statistics on each attribute.
* The following statistics are shown in ‘Selected attribute’ box on the right panel of ‘Preprocess’ window:

1. Name is the name of an attribute
2. Type is most commonly Nominal or Numeric
3. Missing is the number (percentage) of instances in the data for which this attribute is unspecified
4. Distinct is the number of different values that the data contains for this attribute
5. Unique is the number (percentage) of instances in the data having a value for this attribute that no other instances have.

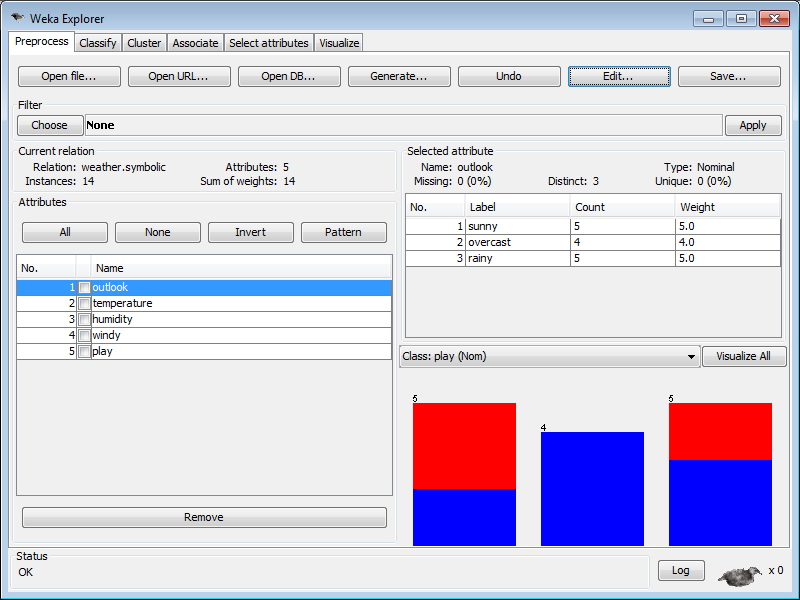


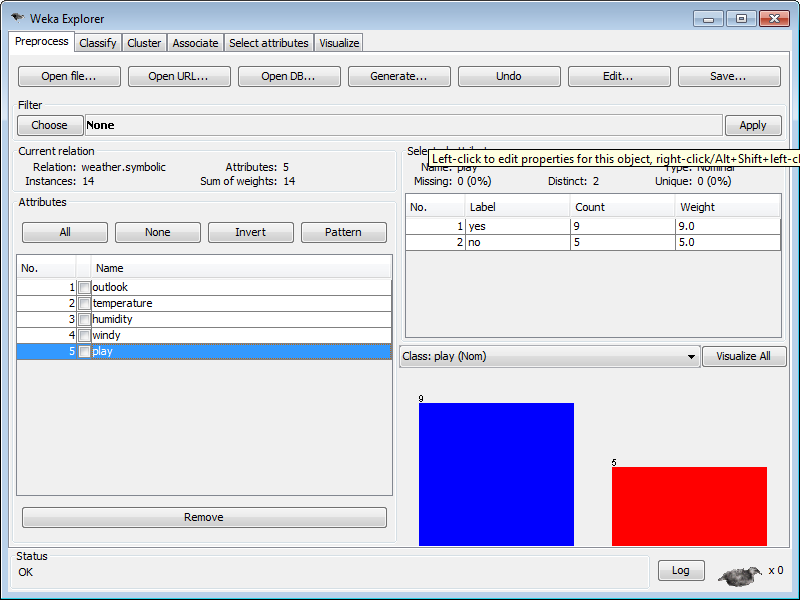
**Steps:**

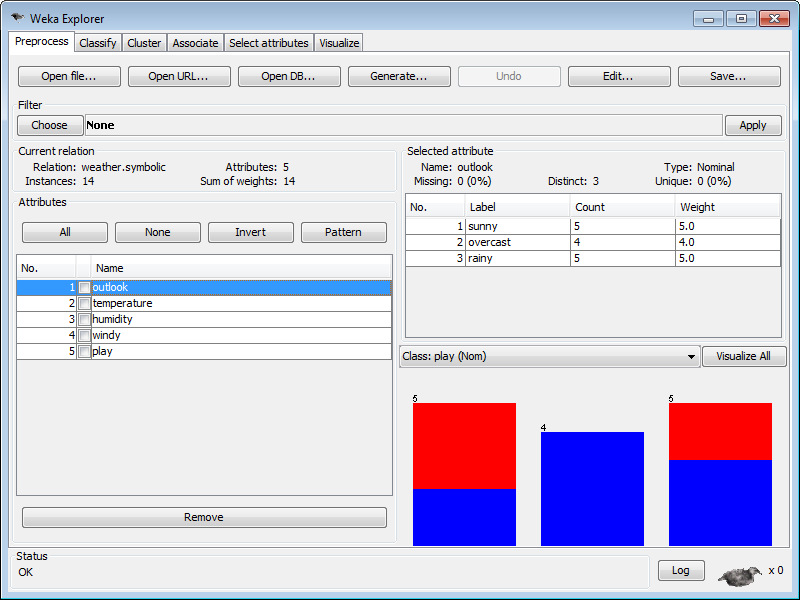
* + - 1. Click on preprocessing step. Open file .Open weathernominal.artf file



* + - 1. Choose filter (supervised/unsupervised)
      2. Choose tabe classify
      3. Tree –decision tree (j48)
      4. Use training set
      5. Click on start



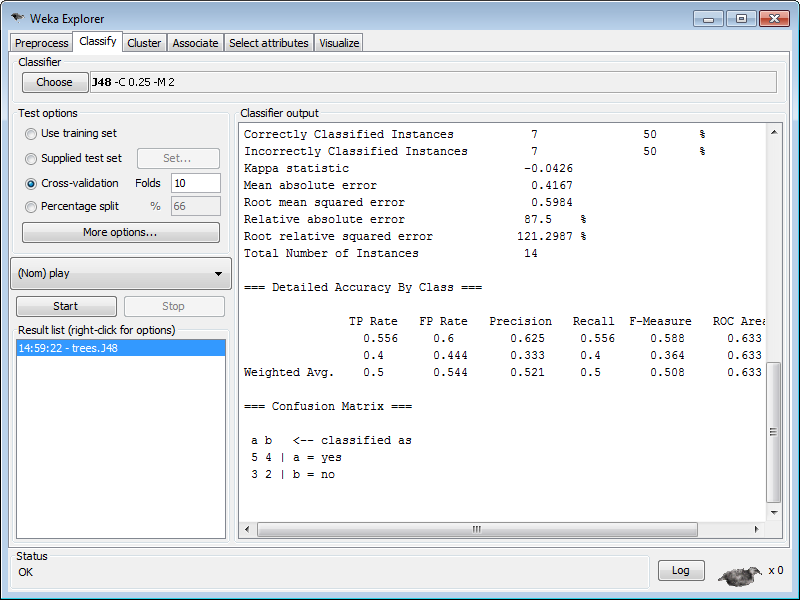


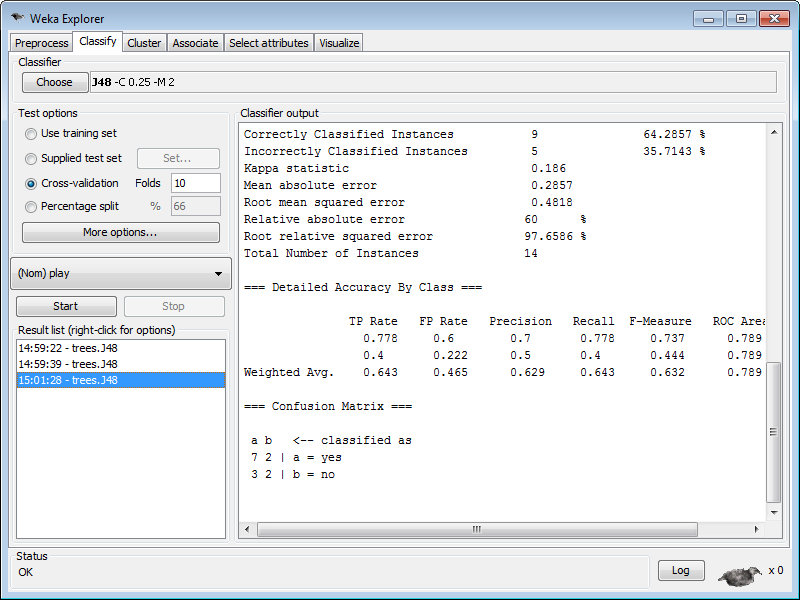


**Classify**

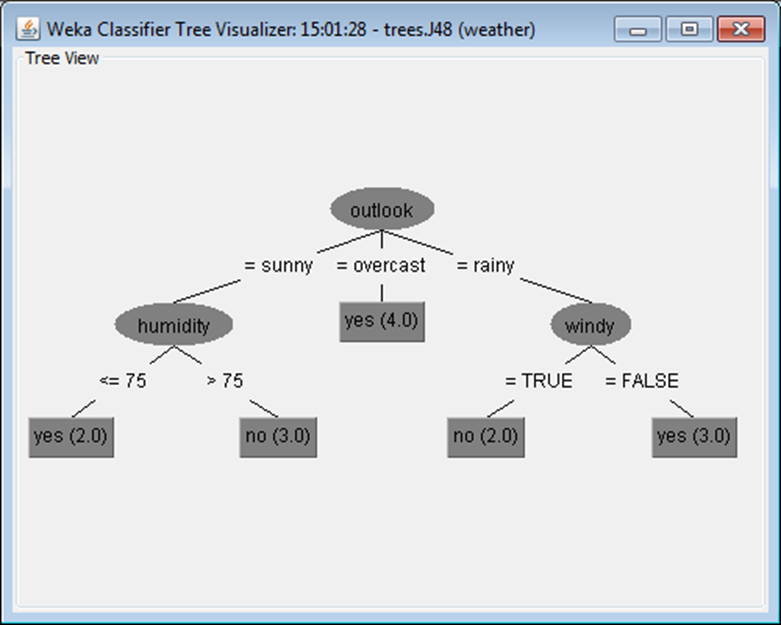
* Classifiers in WEKA are the models for predicting nominal or numeric quantities.
* Click on ‘Choose’ button in the ‘Classifier’ box just below the tabs and select C4.5 classifier WEKA 🡪 Classifiers 🡪 Trees 🡪J48.
* Setting Test Options

1. Use training set: Evaluates the classifier on haw well it predicts the class of the instances it was trained on.
2. Supplied test set: Evaluates the classifier on how well it predicts the class of a set of instances loaded from a file. Clicking on the ‘Set…’ button brings up a dialog allowing you to choose the file to test on.
3. Cross-validation: Evaluates the classifier by cross-validation, using the number of folds that are entered in the ‘Folds’ text field.
4. Percentage split: Evaluates the classifier on how well it predicts a certain percentage of the data, which is held out for testing. The amount of data held out depends on the value entered in the ‘%’ field.



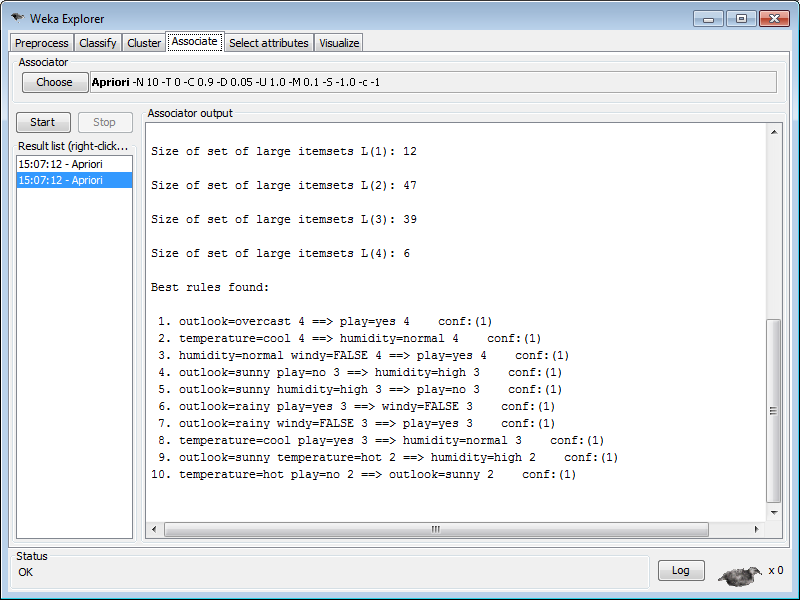


Visualization of Results: Right-click on the entry in ‘Result list’ for which you would like to visualize a tree 🡪Click visualize tree.



**Associate**

* WEKA contains an implementation of the Apriori algorithm for learning association rules.
* It works only with discrete data and will identify statistical dependencies between groups of attributes
* Apriori can compute all rules that have a given minimum support and exceed a given confidence.
* The association rule scheme cannot handle numeric values;



**Creating .arff File:**

@relation loan

@attribute Profession {Faculty,Clerk,Peon,Doctor,Software\_Engineer,Data\_Scientist}

@attribute Income numeric

@attribute Approved {Yes,No}

@data

Faculty 20000 Yes

Clerk 5000 No

Peon 7000 No

Doctor 30000 No

Software\_Engineer 25000       Yes

Data\_Scientist 30000 Yes

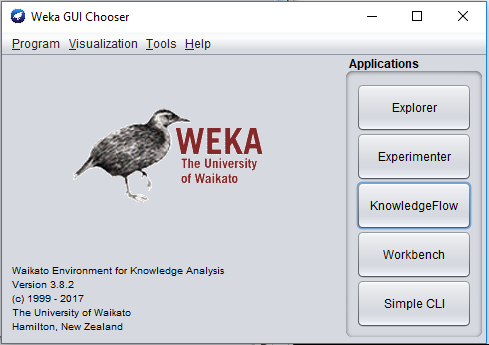
Clerk 5600 Yes

Faculty 10000 No

Doctor 25000 Yes

Software\_Engineer 20000 Yes

**WEKA EXPERIMENTAL**



**Weka Experimental**

* The Weka Experiment Environment enables the user to create, run, modify, and analyse experiments in a more convenient manner than is possible when processing the schemes individually. For example, the user can create an experiment that runs several schemes against a series of datasets and then analyse the results to determine if one of the schemes is (statistically) better than the other schemes.

1. **Defining an Experiment:** When the Experimenter is started, the Setup window (actually a pane) is displayed. Click New to initialize an experiment.
2. **Running an Experiment:** To run the current experiment, click the Run tab at the top of the Experiment Environment window.

If the experiment was defined correctly, the 3 messages shown above will be displayed in the Log panel.

1. Started
2. Finished
3. There were 0 errors

* **Experiment type**

The user can choose between the following three different types

• Cross-validation (default)

performs stratified cross-validation with the given number of folds

• Train/Test Percentage Split (data randomized)

splits a dataset according to the given percentage into a train and a test file (one cannot specify explicit training and test files in the Experimenter), after the order of the data has been randomized and stratified

• Train/Test Percentage Split (order preserved)

because it is impossible to specify an explicit train/test files pair, one can abuse this type to un-merge previously merged train and test file into the two original files (one only needs to find out the correct percentage)

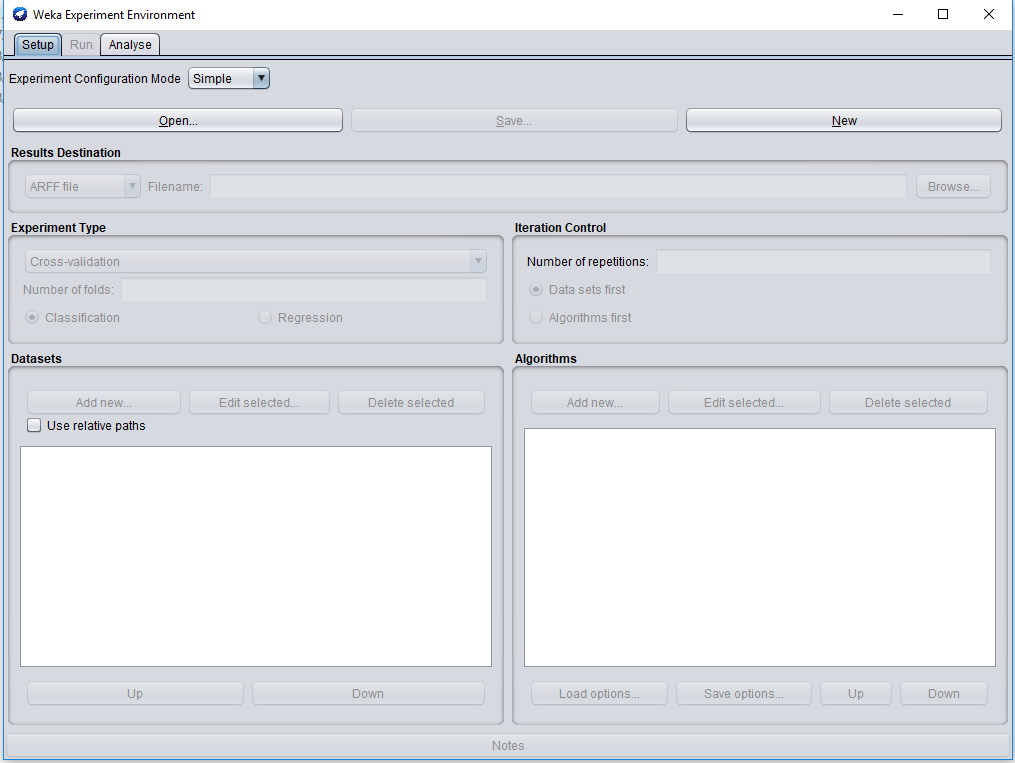
* **Iteration control**

• Number of repetitions

In order to get statistically meaningful results, the default number of iterations is 10. In case of 10-fold cross-validation this means 100 calls of one classifier with training data and tested against test data.

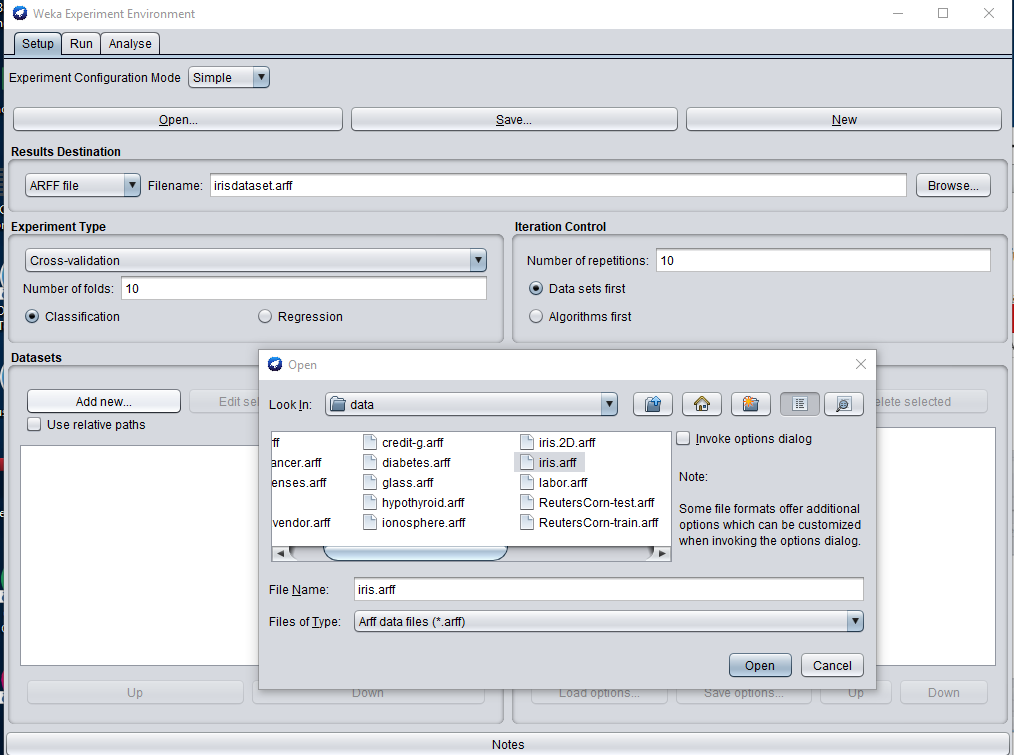
• Data sets first/Algorithms first

As soon as one has more than one dataset and algorithm, it can be useful to switch from datasets being iterated over first to algorithms. This is the case, if one stores the results in a database and wants to complete the results for all the datasets for one algorithm as early as possible.

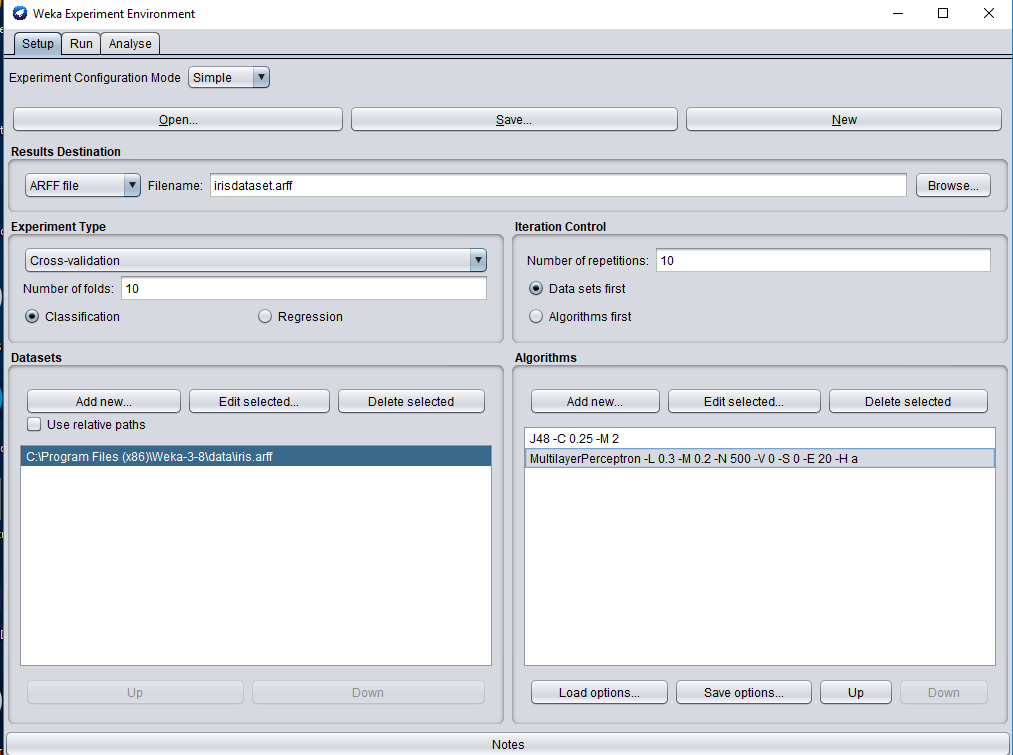


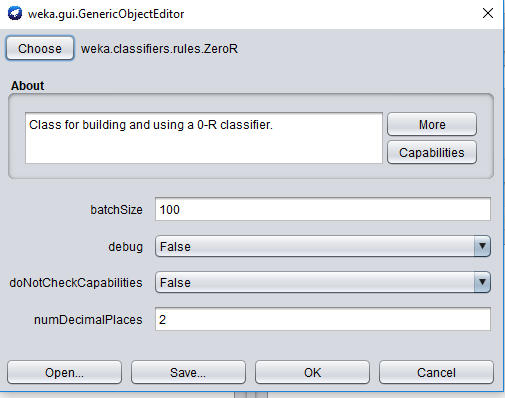
**Steps:**

* + - 1. Choose new and give filename.
      2. Choose arff file (Results in form)
      3. Choose either classifier/regression
      4. Select experiment type either training testing or cross validation.
      5. add new dataset iris.arff



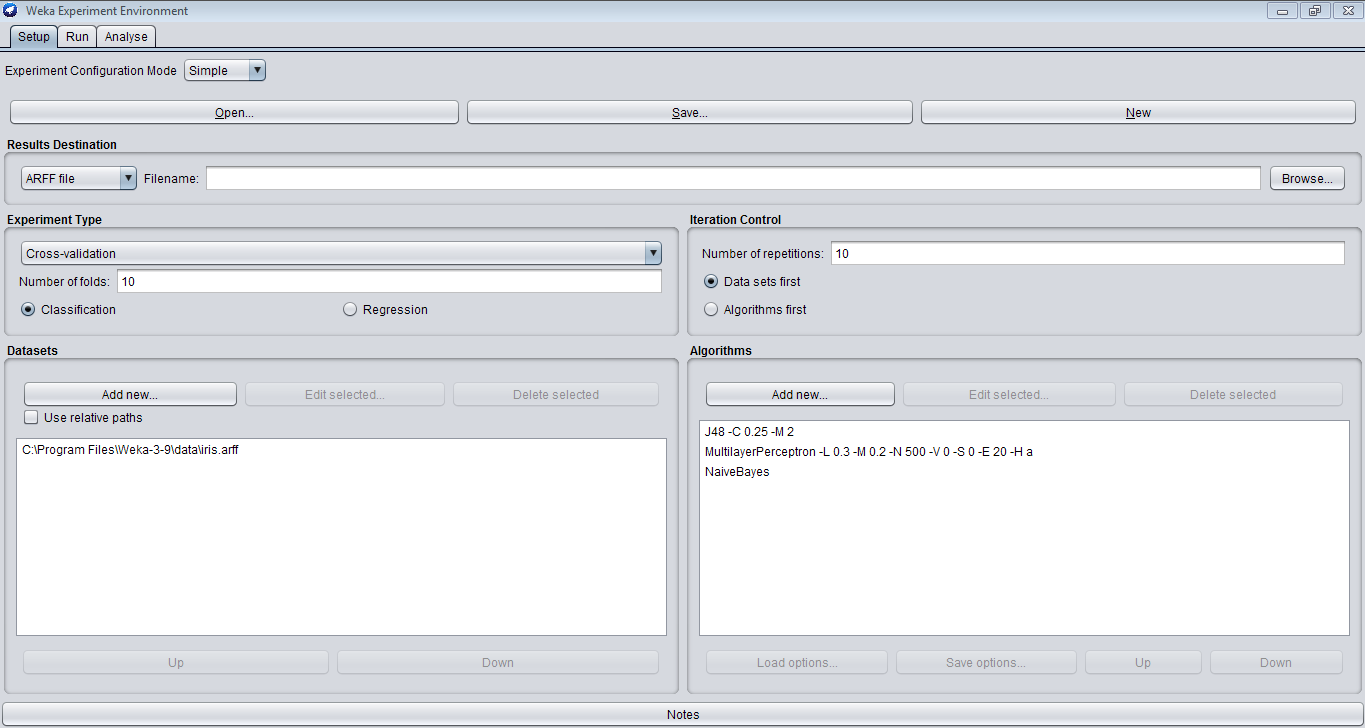
* + - 1. set number of iteration control
      2. Add two algorithm j48 and multilayer perceptron

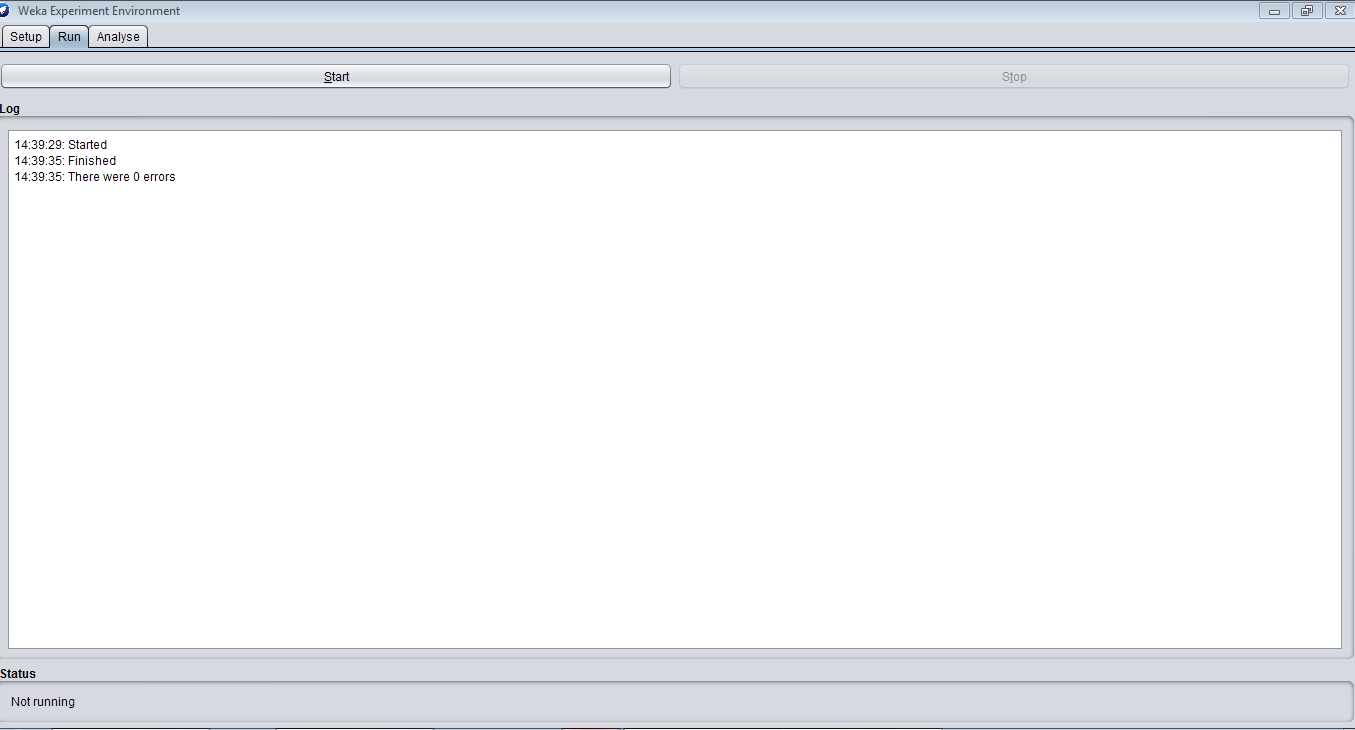


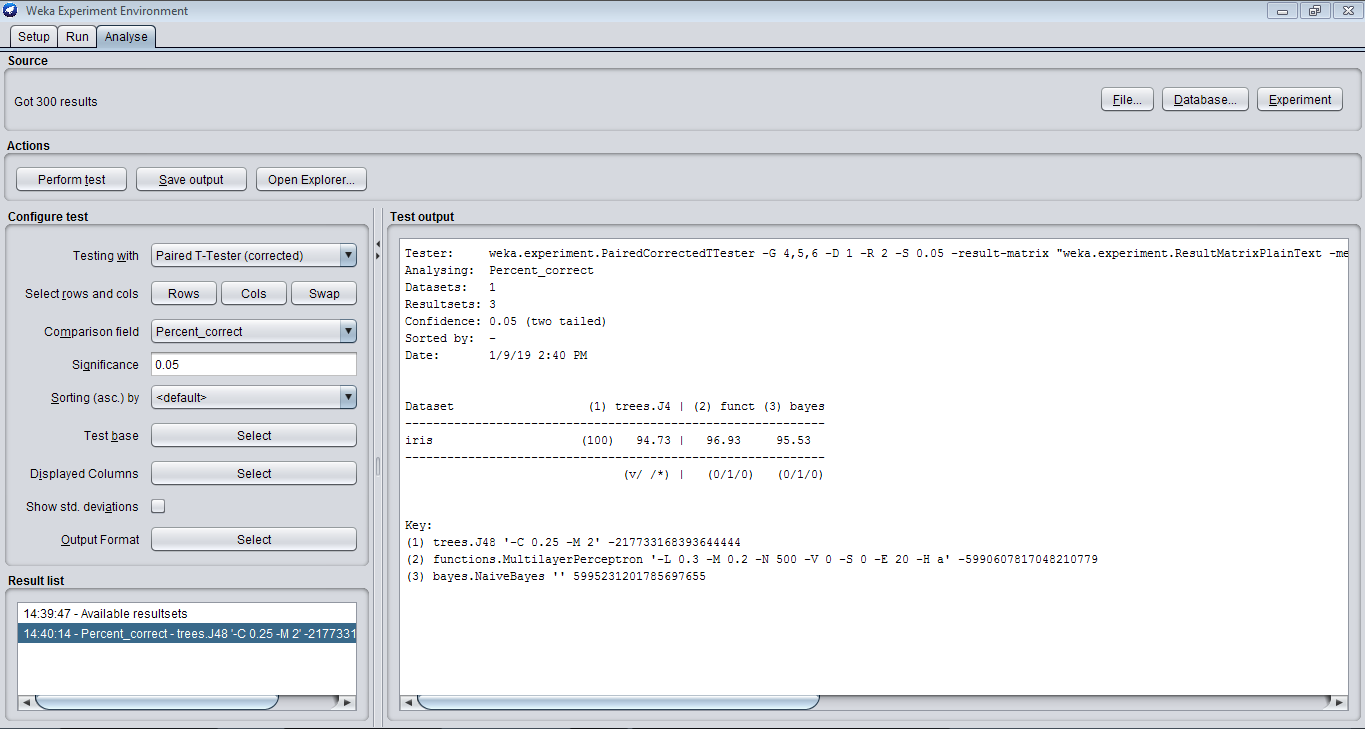


* + - 1. Choose run tab and click on start (0 errors message will come)
      2. Analysis experimenter
      3. Select rows and columns (as it is)
      4. Choose comparison field (as per your requirement)
      5. Click on perform test

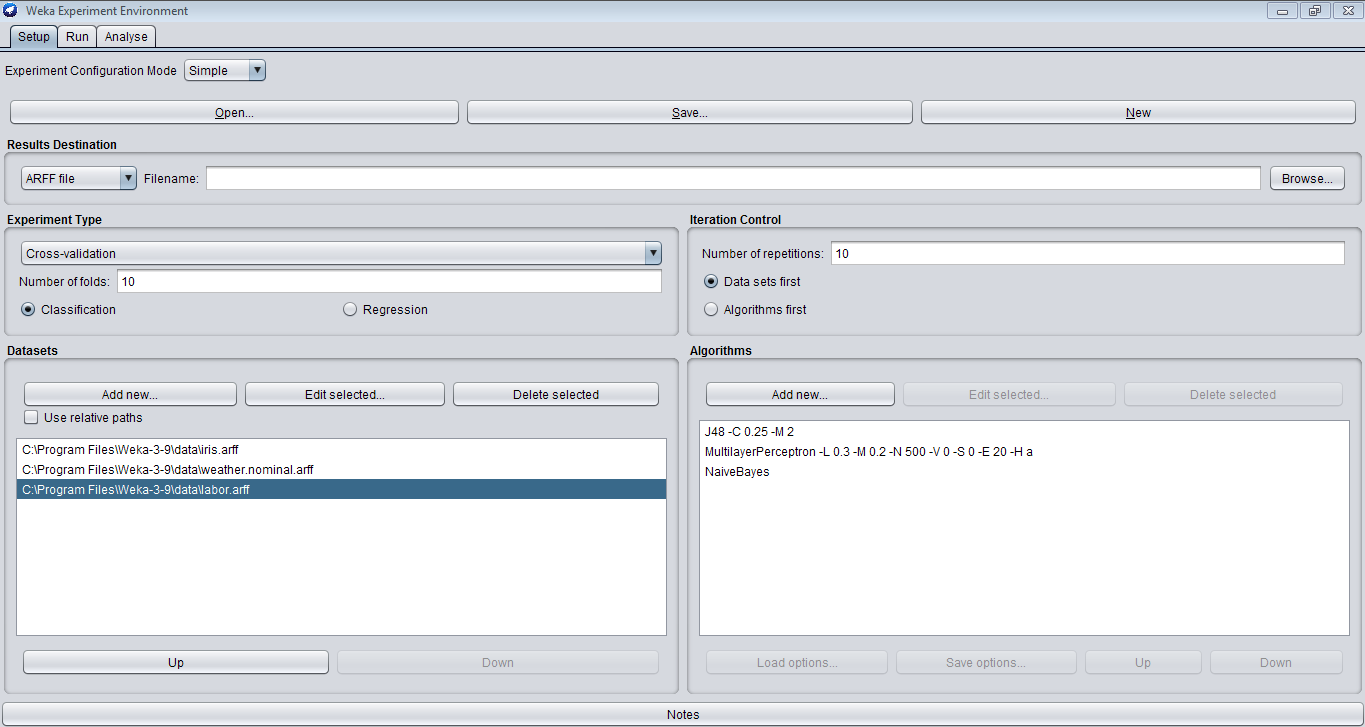
1 dataset,3 algorithms:

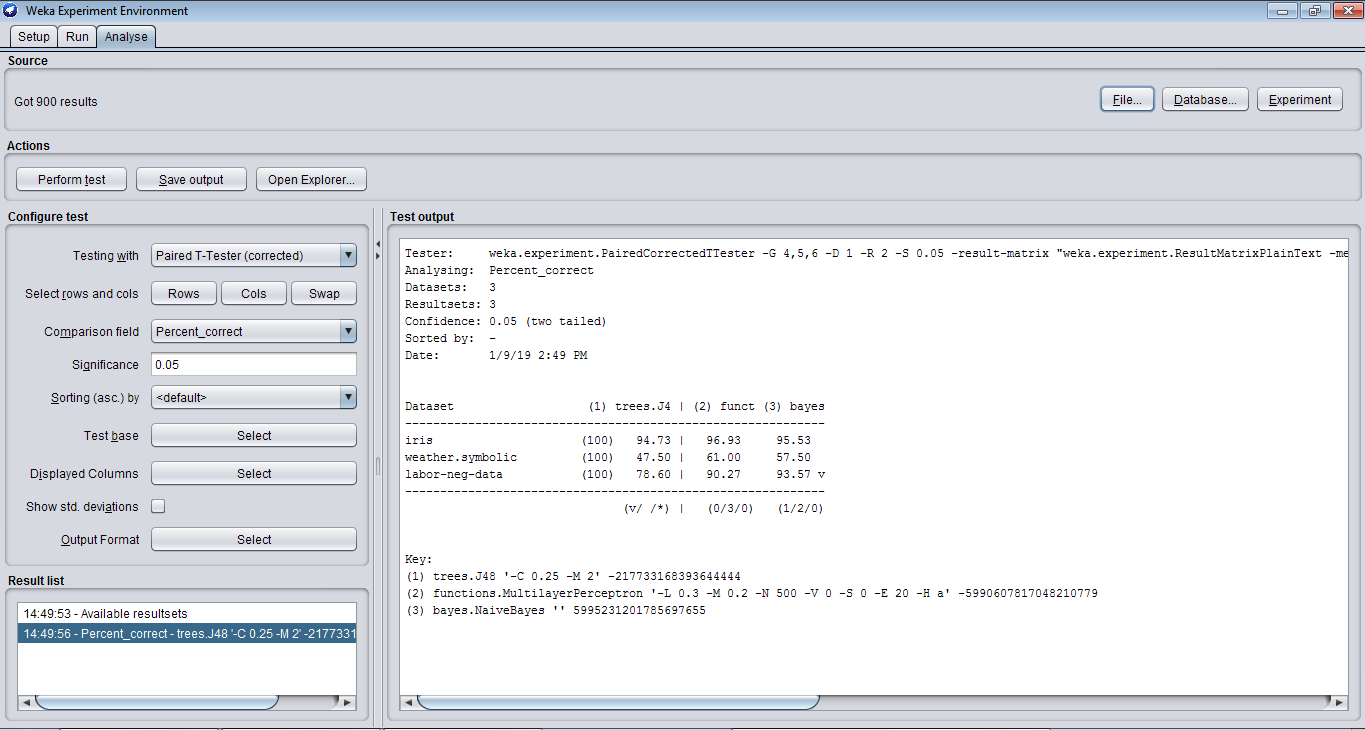




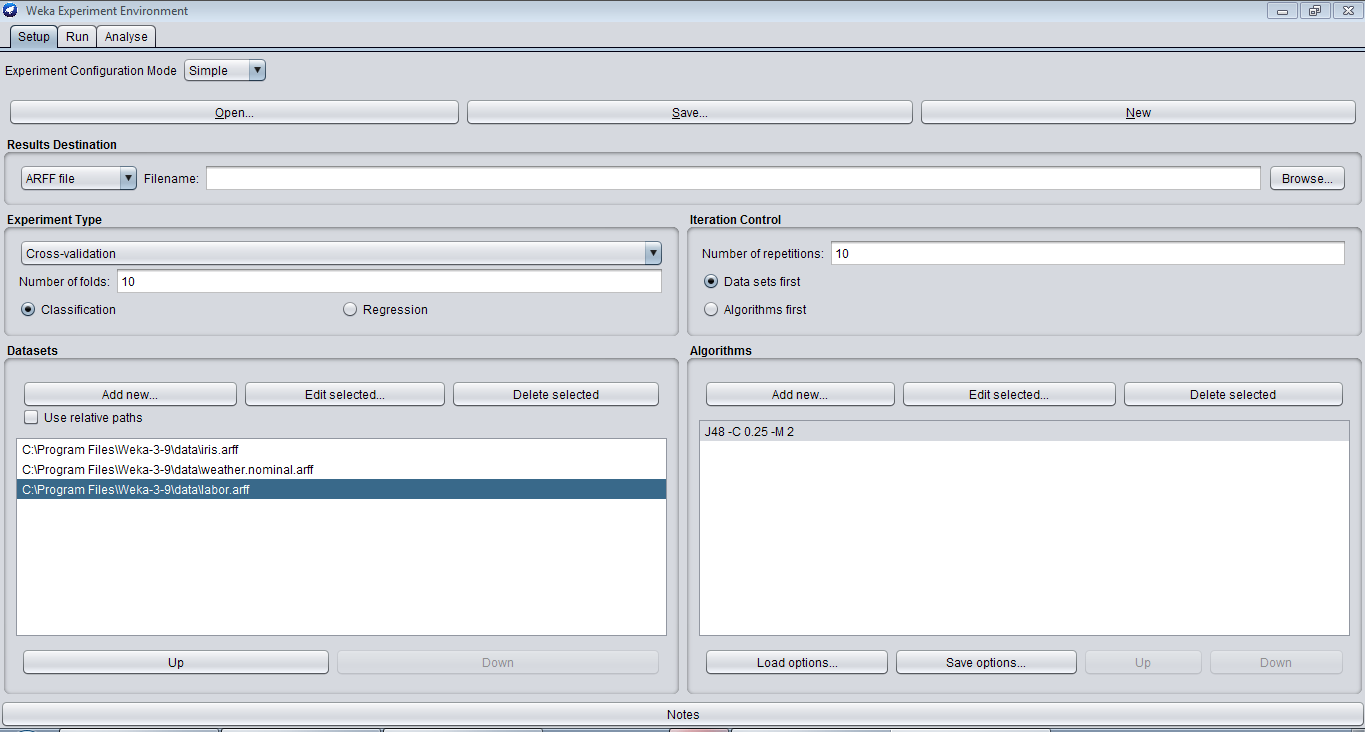


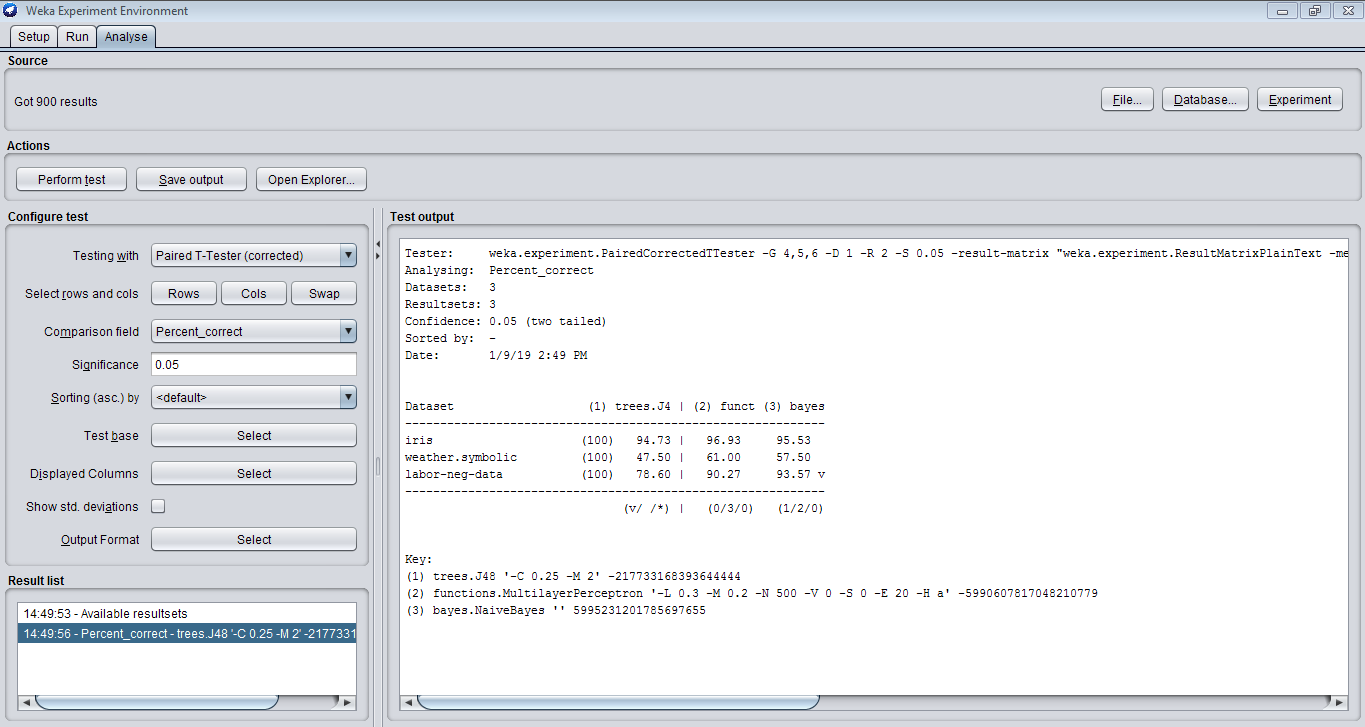
3 datasets,3 algorithms:





3 datasets,1 algorithm:





**Conclusion:**

From this practical we learnt about Different Data Mining Activities using Weka Experimental Tool

From this practical we learnt about Different Data Mining Activities using Weka Explorer Tool.

**PRACTICAL 7**

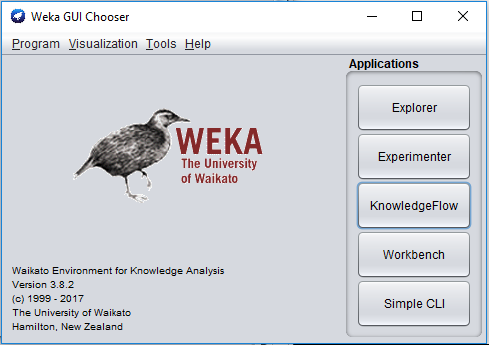
**AIM: Perform Different Data Mining Activities using Weka Knowledge**

**Flow Tool (Open Source Data Mining Tool).**

**S/W: Weka Tool**

**H/W: --**

**Theory:**

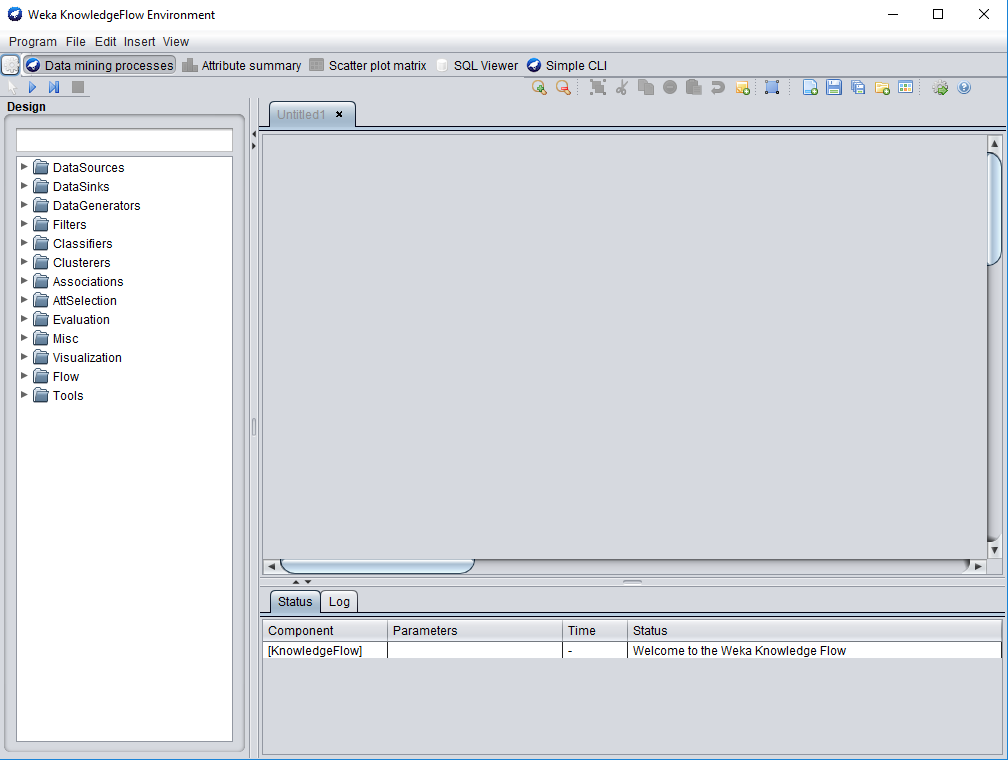


**Weka Knowledge Flow**

* Java-Beans-based interface for setting up and running machine learning experiments
* Data sources, classifiers, etc. are beans and can be connected graphically
* Data “flows” through components

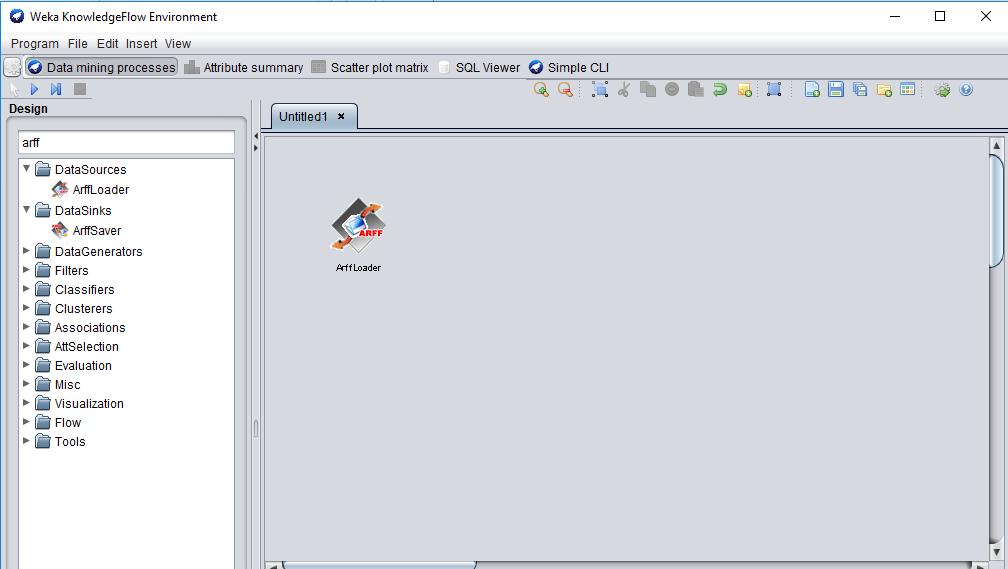
e.g.  data source → filter → classifier → evaluator

* Layouts can be saved and loaded again later
* Features:
* view models produced by classifiers for each fold in a cross validation
* intuitive data flow style layout
* process data in batches or incrementally
* process multiple batches or streams in parallel (each separate flow executes in its own thread)
* chain filters together
* visualize performance of incremental classifiers during processing (scrolling plots of classification accuracy, RMS error, predictions etc.)
* plugin facility for allowing easy addition of new components to the Knowledge Flow

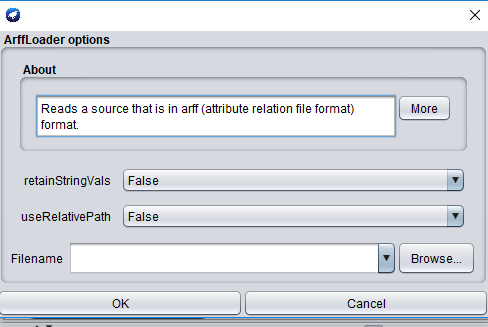


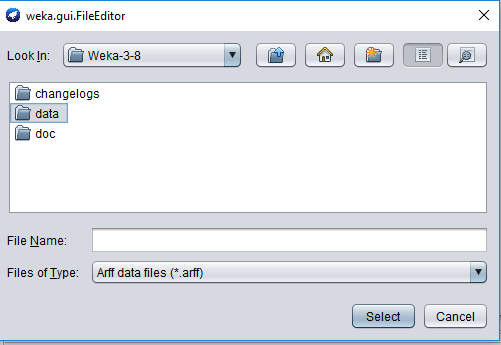
**Steps:**

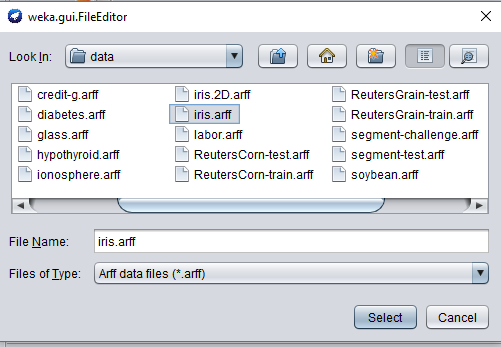
1. Select arff loader



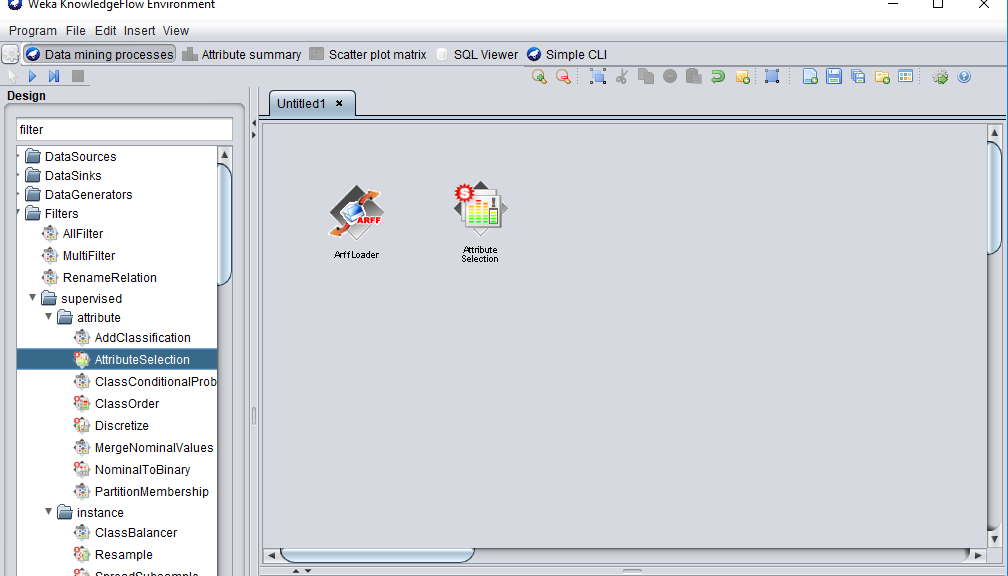
Right click -> configure and brows iris data



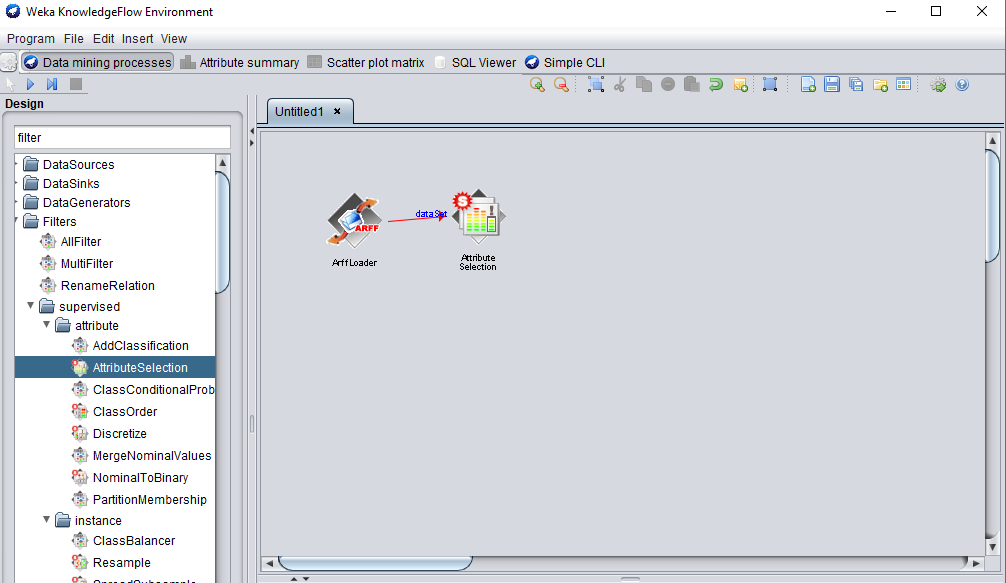




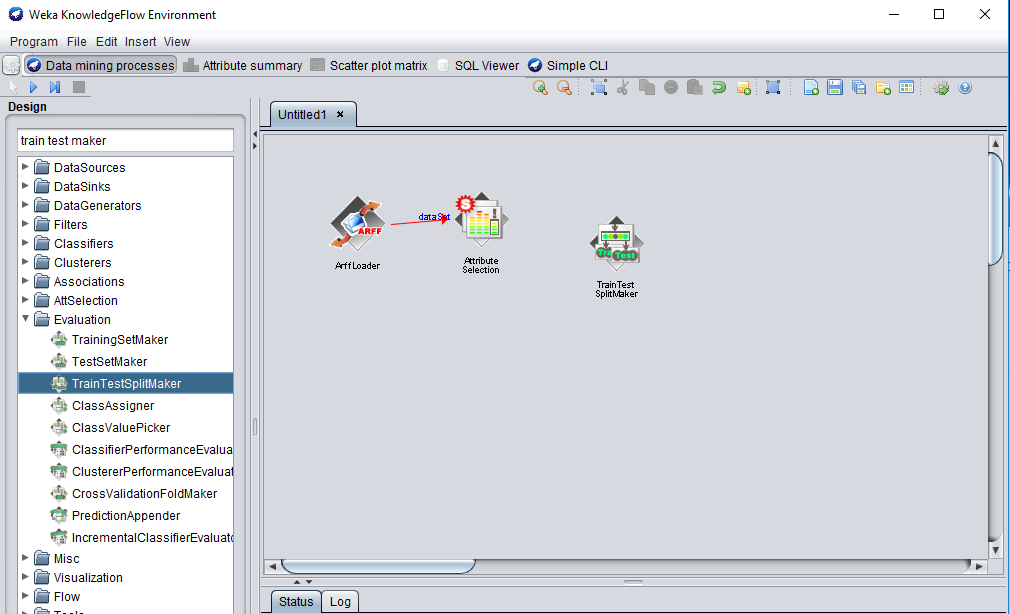
1. Go to filter -> attribute selection double click on it



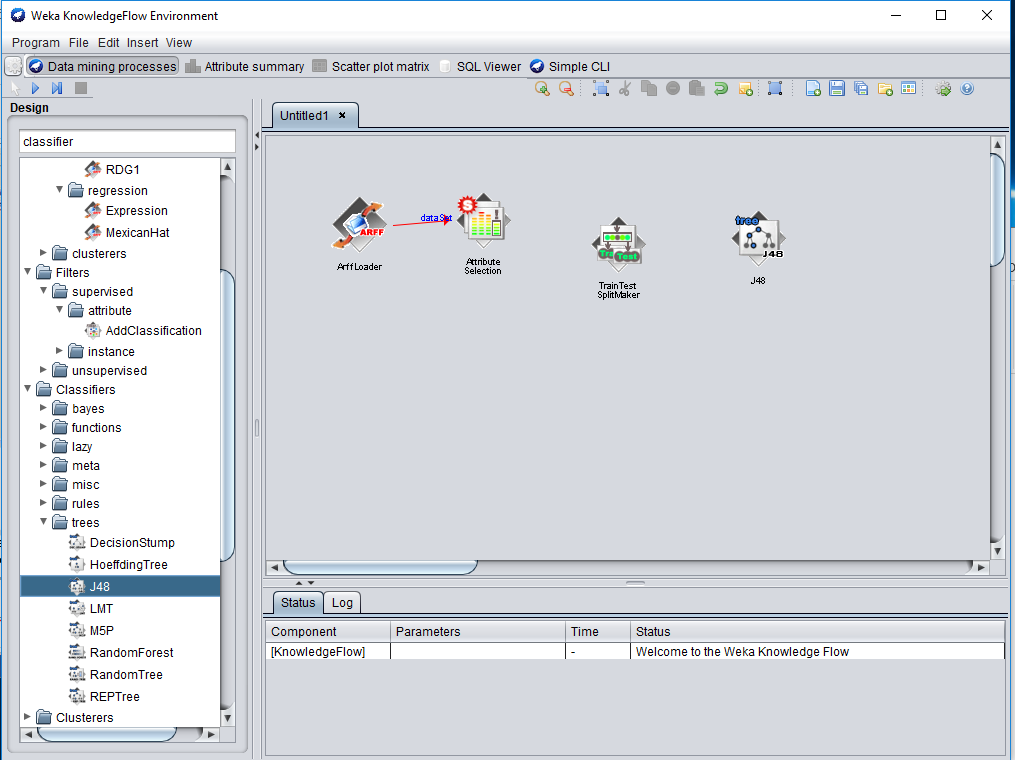
1. Again go to attribute right click dataset (stretch)

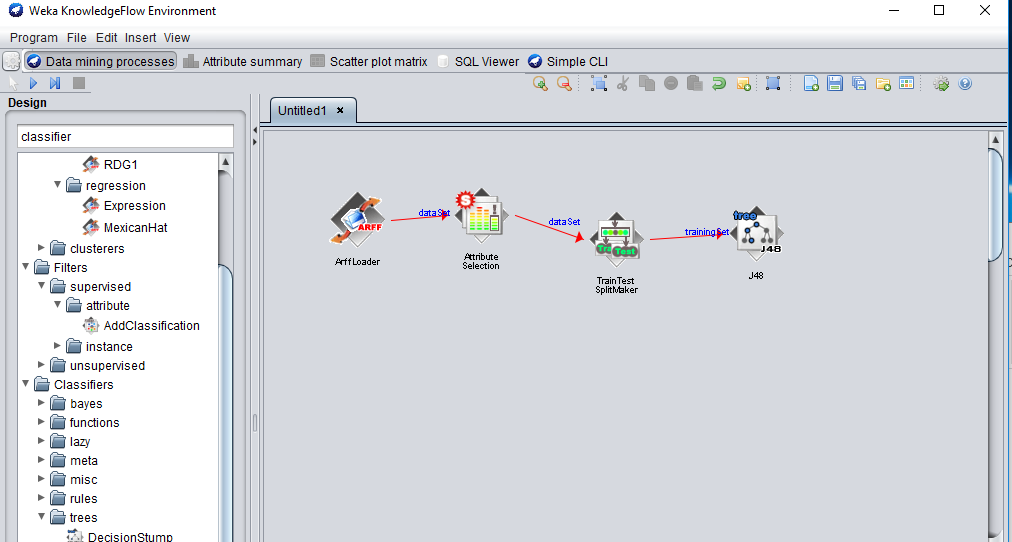


1. Evolution : choose train test maker

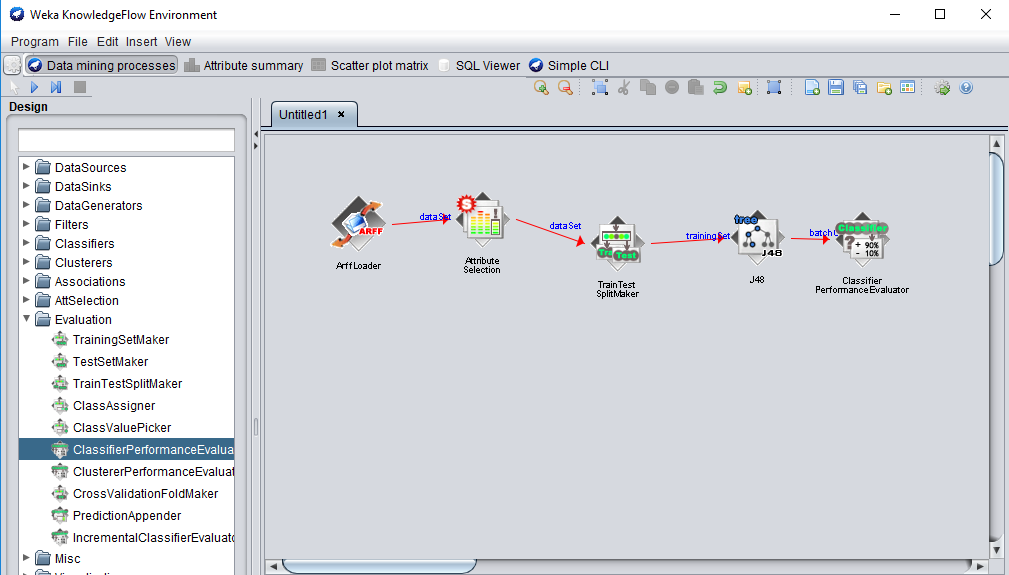


1. Goto classifier: trees and j48 (to evalution by right click and choose batch classifier)

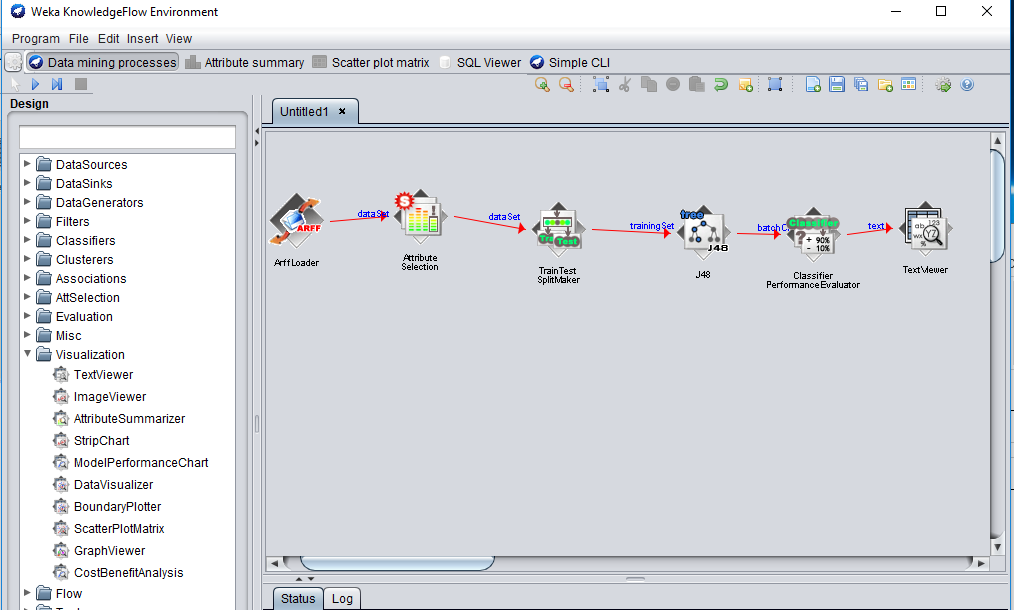


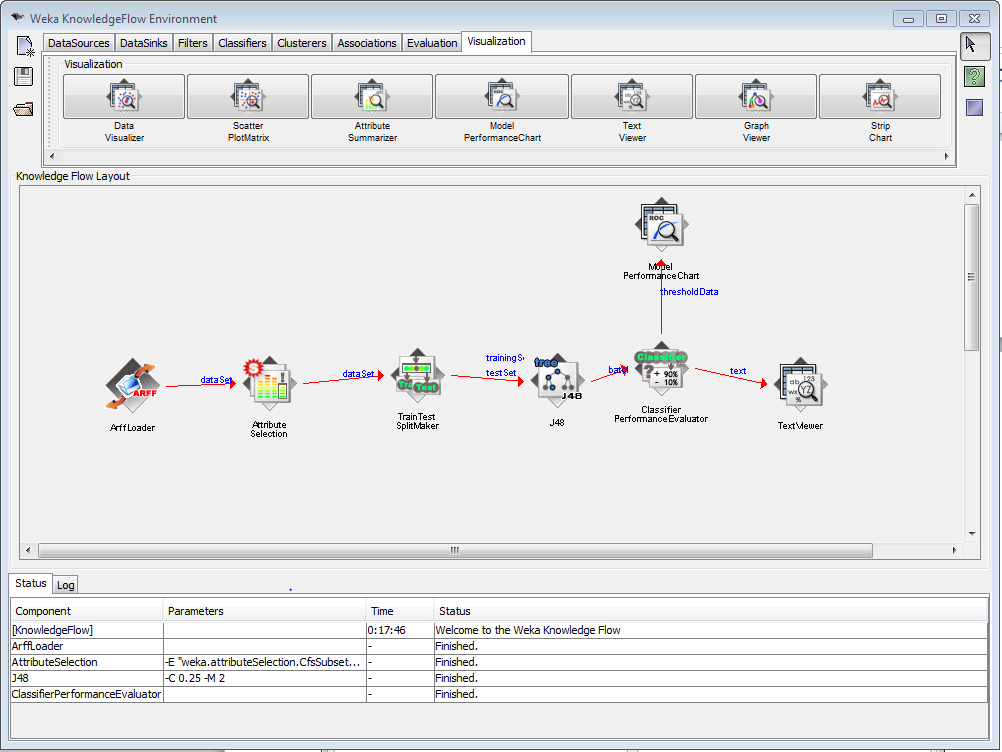


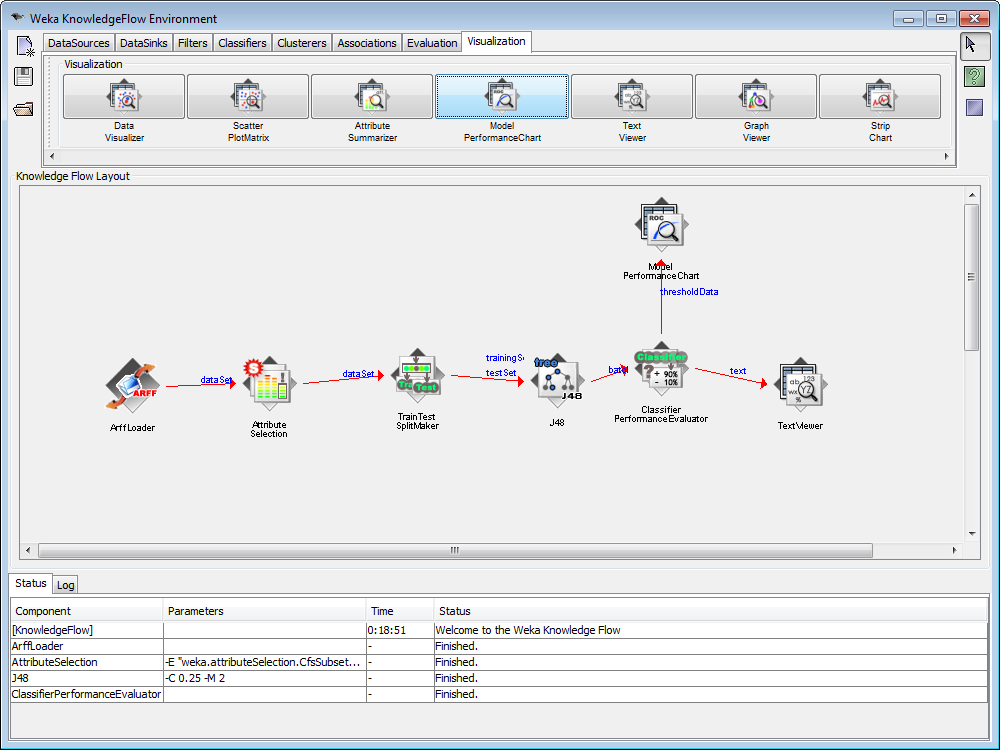
1. Net evaluation classifier performance evaluation (to visualization test)

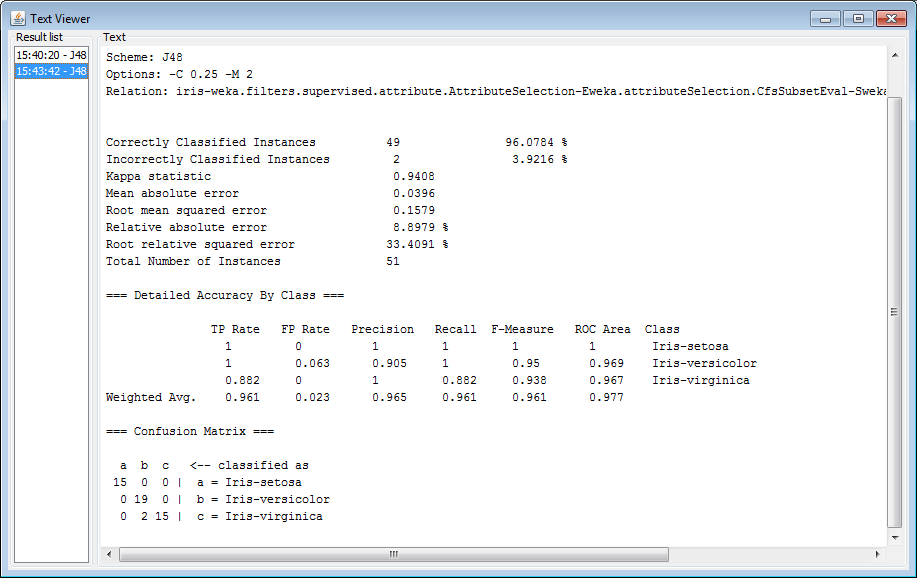


1. Visualization (text view)









**Conclusion:**

From this practical we learnt about Different Data Mining Activities using Weka Knowledge Flow Tool.