

# **ATDIDT User's Guide**

(version 2.x)

University of Liège  
Stochastic Methods Dpt.

Louis Wehenkel  
Christophe Druet

# Contents

<b>1</b>	<b>Overview of ATDIDT</b>	<b>2</b>
1.1	Introduction	3
1.2	Supervised learning problem	3
1.3	Unsupervised learning problem	3
1.4	Different subtasks of data mining	3
1.5	Available methods in ATDIDT	4
<b>2</b>	<b>Software installation of the Linux version</b>	<b>5</b>
<b>3</b>	<b>Common features of all methods</b>	<b>6</b>
3.1	Introduction	6
3.2	Main menu	6
3.2.1	Databases	6
3.2.2	Graphics	8
3.2.3	Automatic learning	8
<b>4</b>	<b>Visualisation tools</b>	<b>9</b>
4.1	Histograms	9
4.1.1	histogram	9
4.1.2	cond_histogram	9
4.1.3	db_stats	9
4.2	Cumulative distribution	9
4.3	Scatter plot	9
4.3.1	scatter_plot	9
4.3.2	cond_scatter_plot	10
4.3.3	colour_scatter_plot	10
4.3.4	scatter_plot_val	10
4.4	Dendograms	10
4.4.1	dendograms	10
<b>5</b>	<b>Automatic learning</b>	<b>11</b>
5.1	Introduction	11
5.2	Tree induction	11
5.2.1	Decision trees	11
5.2.2	Regression trees	12
5.2.3	Other features	12
5.3	Linear regression	13

5.3.1	<a href="#">learn_linear_regression_approximation</a>	13
5.3.2	<a href="#">tree_booster</a>	13
5.3.3	<a href="#">test_linear_regression_approximation</a>	13
5.3.4	<a href="#">hinges</a>	13
5.4	Non linear regression	13
5.4.1	<a href="#">train_mlp_regression/train_mlp_classification</a>	14
5.4.2	<a href="#">set_mlp_hidden_structure</a>	14
5.4.3	<a href="#">monitor_test_set</a>	14
5.4.4	<a href="#">mlp_stopping_pars</a>	14
5.4.5	<a href="#">output_layer_activation</a>	14
5.5	Similarity	14
5.5.1	<a href="#">K nearest neighbors</a>	14
5.5.2	<a href="#">K-means</a>	15

# About this document

## Purpose of this document

This document provides high level information and perspective for the purpose of understanding and using of the ATDIDTdata mining software.

This document provides information on how to use the software. It explains how to load databases and find the most appropriate data mining method for your problem.

## Who should use this document

This guide is intended for individuals who wants to discover the software and also those persons who wish to gain a larger perspective on how the software is organized from an implementation standpoint.

## Structure of this document

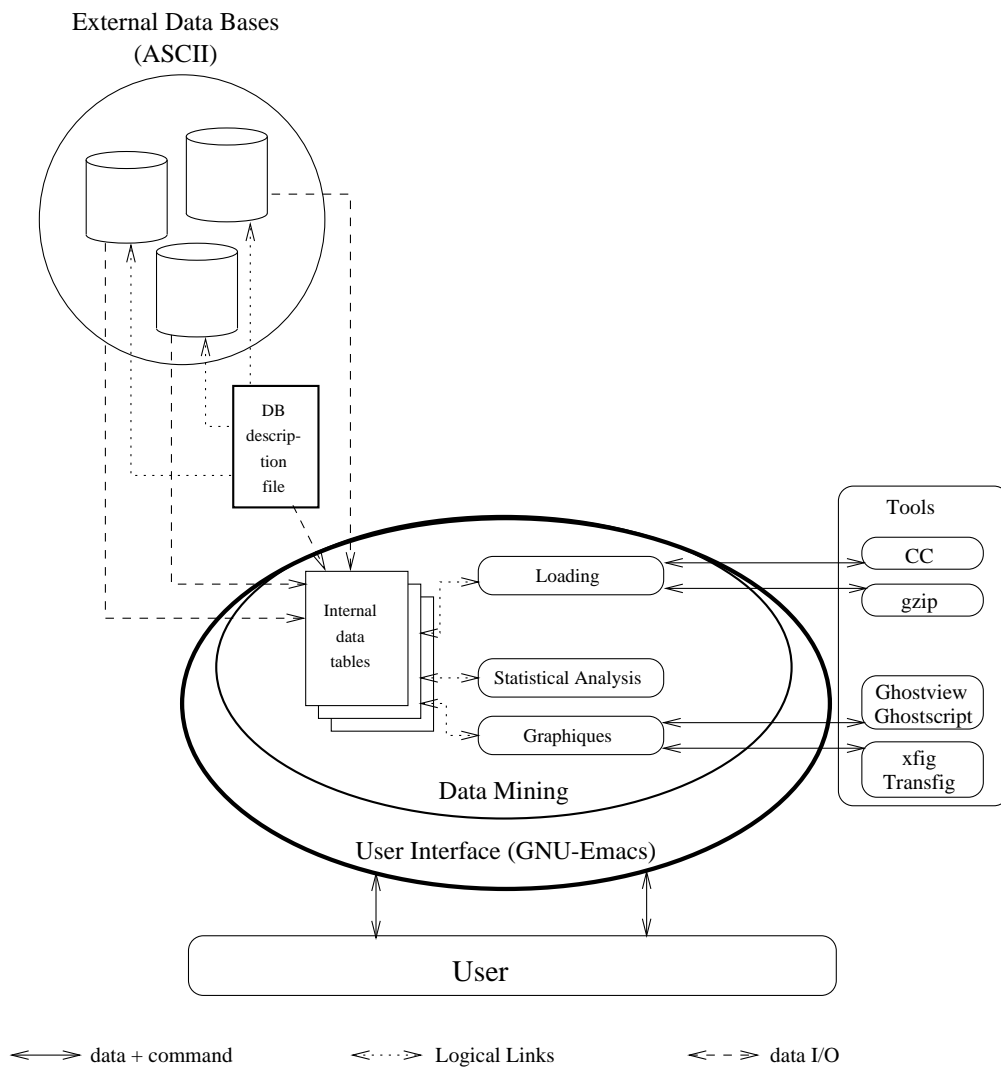
This document is structured in the following manner:

- Chapter 1 gives an overview of ATDIDT.
- Chapter 2 provides information and instructions on installing the software.
- Chapter 3 describes the common features to all the methods implemented in the software.
- Chapter 4 gives an overview of the visualisation tools of the software.
- Chapter 5 describes the different supervised methods available in the software, provides information on the user interface and explains how to use the methods.

## Conventions

# Chapter 1

## Overview of ATDIDT



## 1.1 Introduction

The ATDIDT software allows you to learn automatically. What is **Automatic Learning**<sup>1</sup> (AL)? Automatic learning is a highly multidisciplinary research field and set of methods (involving theoretical and applied methods from statistics, computer science, artificial intelligence, biology and psychology) to extract high level synthetic information (knowledge) from data bases containing large amounts of low level data. Its applications to engineering problems are extremely promising.

## 1.2 Supervised learning problem

One can formulate the generic problem of supervised learning as follows:

*Given a set of examples (the learning set (LS) of associated input/output pairs, derive a general rule representing the underlying input/output relationship, which may be used to explain the observed pairs and/or predict output values for any new unseen input.*

One uses the term *attribute* to denote the parameters (or variables) used to describe input and output information. An attribute can be symbolic or numerical depending on the methods.

The main unifying concept in automatic learning is to view it as a *search process* in a space of candidate *models* (or *hypotheses*). The search process aims at constructing a model of maximal quality and is guided by the information contained in the learning set and possibly some background knowledge about the practical problem domain.

The supervised problem is divided into two types: *classification* and *regression*. The classification methods concern problems where the output is a class (for instance, *stable* or *unstable*). The regression methods deal with continuous numerical attributes.

## 1.3 Unsupervised learning problem

In contrast to supervised learning, where the objective is clearly defined in terms of modeling input/output relationships, unsupervised learning methods are not oriented towards a particular prediction task. They try to find by themselves the existing relationships among states characterized by a set of attributes.

## 1.4 Different subtasks of data mining

1. **Representation** consists of (i) choosing appropriate input attributes to represent practical problem instances, (ii) defining the output information, and (iii) choosing a class of models suitable to represent input/output relationships.
2. **Attribute selection** aims at reducing the dimensionality of the input space by dismissing attributes which do not carry useful information to predict the considered output information.

---

<sup>1</sup>Some people call it *Data Mining* (DM), others *Knowledge Discovery in Data* (KDD). We will use those terms without distinction in this document.

3. **Model selection** will typically identify in the predefined class of models the one which best fits the learning states.
4. **Interpretation and validation** are very important in order to understand the physical meaning of the synthesized model and to determine its range of validity.
5. **Model use** consists of applying the model to predict outputs of new situations from the values assumed by the input parameters.

## 1.5 Available methods in ATDIDT

	Classification	Regression
SUPERVISED	Tree Induction (Recursive Partitioning)	
	MultiLayer Perceptron (Artificial Neural Network)	
	K-Nearest Neighbor methods	
		Generalized Linear Regression
UNSUPERVISED	Graphical Inspection Tools for Data Exploration	
	Clustering by K-means	
	Correlation Analysis using Hierarchical Agglomeration Clustering	
	Feature Extraction using MLPs	

## **Chapter 2**

# **Software installation of the Linux version**

*This information is not available in the present version of this document.*



# Chapter 3

## Common features of all methods

### 3.1 Introduction

The whole program has been implemented in LISP with some links to C for efficiency purposes. The interface is written in Emacs LISP. The user won't need to learn LISP in order to use the program. We will provide information to save the user being stuck.

### 3.2 Main menu

The two major options are DATA\_BASE and AUTOMATIC\_LEARNING. We will follow the standard steps the user will follow.

#### 3.2.1 Databases

##### Database loading

LOAD\_DB is the first obvious step in order to use the program. It will prompt the user for the name of a file containing the data base specification. There are two ways to load a database: the long way and the short way.

**Long way:** In addition to the data file the user defines a database declaration file where it defines the attributes, the type of data, the files. The attributes can be *attributs-explicites* (explicitly declared) or *attributs-fonctions* (constructed by manipulating or combining the previous ones). Several types of attributes can be declared: *ordonne* (in numerical order), *qualitatif-quinlan*, *qualitatif-binaire*(defining classes). There are three classes of attribute vlues: *real*, *integer* and *member*.

```
(DECLARE-BD
  example-db "Example of database"
  :OBJETS (prefixes "EX" 1 5000)
  :RE-INITIALISER t
  :ATTRIBUTS-EXPLICITES
  (
    (x "real in numerical order (range [-0.5,0.5] and default value 0.0)"
      :valeurs (real -0.5 0.5) :par-defaut 0.0 :type ordonne)
    (y "integer in numerical order (range [0,500] and default value 25)"
      :valeurs (integer 0 500) :par-defaut 25 :type ordonne)
    (z "RED or BLUE (default value RED)"
      :valeurs (member "RED" "BLUE") :par-defaut "RED" :type qualitatif-quinlan)
  )
  :ATTRIBUTS-FONCTIONS
  (
    (x*y "x multiplied by y"
      :valeurs (real -250. 250) :par-defaut 0.0 :type ordonne
      :fonction (* (x objet) (y objet)))
    (class_x "x bigger than 0.0"
      :valeurs (member "POS" "NEG") :par-defaut "POS" :type qualitatif-quinlan
      :fonction (if (> x 0.) "POS" "NEG"))
  )
  :chargement
  (
    ((x y)
      :dans (vms-file "example.dat") :format (object x y z)
;object allows to skip the corresponding element
    )
  )
)
(load-attributes
  example-ls ""
  :bd example-db
  :objets t
  :attributs (x y z)
)
```

The data file consists in a sequence of number corresponding to `:format (object x y z)`.

**Short way:** The user provides only a data file. The lines starting with a semicolon are comments, and may appear anywhere in the file, except for the first line which is mandatory and must be exactly as in the example below. The first 'non-comment' line must contain the name of the database. The second 'non-comment' line contains the attribute names immediately followed by their types: numerical or symbolic. A numerical attribute can be real or integer. A symbolic attribute denotes any discrete attribute. Its value can be symbols, numbers or both of them. Then each line corresponds to one example and contains the values of the attributes. If the first attribute name is object and its type is name, then the first column is the object name (otherwise the name will be compute on the fly based on the line number). Comment lines start with a semi-column.

```
!--+ This is javadb type file
; Data base example
omib
object name x numerical y numerical z numerical
OP1 5.34 6.123 3.584
OP2 6.37 2.123 4.1243
...
OP10000 2.92 9.176 0.14
```

### Subset selection

The major points of the subsets selection is to define a *learning set* (LS<sup>1</sup>) and a *test set* (TS<sup>2</sup>): SUBSETS\_SELECTION.

LEARNING\_SET and TEST\_SET allow the user to select objects. To define the sets one can use the options in the following table to select objects.

(member $x_1 x_2 \dots$ )	objects $x_1, x_2 \dots$
(first $n$ )	the $n$ first objects loaded
(last $n$ )	the $n$ last objects loaded
(from $n_1 n_2$ )	the objects from $n_1$ to $n_2$
(random $n set$ )	$n$ objects selected randomly in the set $set$
(not-in $set$ )	all the objects which are not in $set$
(suchthat $att cdt set$ )	the objects such that $att$ respects the condition $cdt$ in the set $set$

### Attribute selection

The next step is of course to select the candidate attributes (inputs) to use and for supervised learning the goal (output): ATTRIBUTES\_SELECTION.

CANDIDATE\_ATTRIBUTES is used to define the candidate attributes (\*candidate-attributes\*). The candidate attributes are in the upper window and the non-candidate attributes in the lower window.

GOAL\_CLASSIFICATION is the output in case of a classification problem (decision trees, MLPs).

GOAL\_REGRESSION is the output in case of a regression problem (linear regression, neural networks, regression trees).

The user can also define new attribute functions with DEFINE\_ATTRIBUTE.

## 3.2.2 Graphics

We consider the visualisation of the data as the next **very** important step in the process. Even if the immediate use of automatic learning methods looks attractive at this point of the procedure, this is really usefull to examine the data in details to see whether the samples of attribute values are consistent. The chapter 4 will provide information about the visualisation tools in ATDIDT.

## 3.2.3 Automatic learning

The next step AUTOMATIC\_LEARNING is to choose the method that fits best to the problem. We will consider each method separately and provide the user with information about the parameters to set in order to correctly use the methods. The chapter 5 will describe all the available methods in ATDIDT.

<sup>1</sup>Variable \*learning-set\* in LISP.

<sup>2</sup>Variable \*test-set\* in LISP

# Chapter 4

## Visualisation tools

Those methods are very important as a starting point when you are confronted with new data. Each method described here uses the objects contained in the [LS](#). They are part of the menu GRAPHICS.

### 4.1 Histograms

The number of bars can be set in `NUMBER_OF_BARS`.

#### 4.1.1 histogram

Draws the absolute frequencies versus the value of the chosen attribute.

#### 4.1.2 cond\_histogram

Draws the absolute frequencies versus the value of the chosen attribute and colors the bars conditionally to `*goal-classification*`.

#### 4.1.3 db\_stats

Draws the conditional histograms for all the attributes in `*candidate-attributes*`.

### 4.2 Cumulative distribution

`CUMULATIVE_DIST` draws the cumulative distribution of the chosen attribute, i.e. the integral of the histogram with percentiles.

### 4.3 Scatter plot

#### 4.3.1 scatter\_plot

Draws one numerical attribute versus an other numerical attribute.

### 4.3.2 `cond_scatter_plot`

Draws one numerical attribute versus an other numerical attribute colored conditionally to the `*goal-classification*`.

### 4.3.3 `colour_scatter_plot`

Draws one numerical attribute versus an other numerical attribute colored conditionally to a third attribute.

### 4.3.4 `scatter_plot_val`

Draws one numerical attribute versus an other numerical attribute. The points are flagged with the value of a third attribute.

## 4.4 Dendograms

### 4.4.1 `dendograms`

Draws the correlation of the numerical attributes in `*candidate-attributes*`. The principle is (i) to calculate the correlation factor of each pair of attributes, (ii) to group to two most correlated and (iii) to continue the grouping process using the smallest correlation factor for the groups:

$$\text{corr}([A_1, A_2], A_3) = \min(\text{corr}(A_1, A_3), \text{corr}(A_2, A_3))$$

# Chapter 5

## Automatic learning

### 5.1 Introduction

The menu `AUTOMATIC_LEARNING` allow the user to choose the methods. This chapter will describe the different methods implemented in `ATDIDT`. We will separate those methods in groups:

1. `TREE_INDUCTION`,
2. `LINEAR_REGRESSION`,
3. `NON_LINEAR_REGRESSION`,
4. `SIMILARITY`.

Most of the methods will be described in terms of algorithm, parameters and problem type. Several methods are obvious and don't need a detailed description.

### 5.2 Tree induction

The menu `TREE_INDUCTION` gives the user access to the various options for decision and regression tree induction. All the regression and decision trees are stored internally in the variable `*DECISION-TREES*`. The commands using suffix "TREE" work on both decision and regression trees.

`CHOOSE_TREE` allows to change the value of `*CURRENT-TREE*` by selecting a tree from the list `*DECISION-TREES*`.

#### 5.2.1 Decision trees

`BUILD_DT` starts building a decision tree with presently selected `*candidate-attributes*`, parameters `*alfa*` and `*h-min*` and `*goal-classification*`. The global variable `*current-dt*` is set to the result, which is also pushed on the global variable `*decision-trees*`. The program asks the user to provide the name of the tree, and generates a new function attribute which represents the value (class) computed by the tree for an object. Note that the list of `*candidate-attributes*` is expanded by replacing temporal attribute specifications by a list of scalar ones, and filtered to

remove attributes which type is not handled by decision tree building. Admissible types are: *ordered*, *qualitatif-quinlan*, *qualitatif-binaire* and *linear-combination*.

SET\_\*ALFA\* allows to change de value of parameter *\*alfa\** for detecting "DEADENDS" in decision trees.

SET\_\*H-MIN\* allows to change the value of parameter *\*h-min\** for detecting "LEAVES" in decision trees.

## 5.2.2 Regression trees

BUILD\_RT starts building a regression tree with presently selected *\*candidate-attributes\**, parameters *\*total-variance-min\** and *\*v-min\**, and *\*goal-regression\**. The global variable *\*current-dt\** is set to the result, which is also pushed on the global variable *\*decision-trees\**. The program asks the user to provide the name of the tree, and generates a new function attribute which represents the value (regression) computed by the tree for an object. Note that the list of *\*candidate-attributes\** is expanded by replacing temporal attribute specifications by a list of scalar ones, and filtered to remove attributes which type is not handled by regression tree building. Admissible types are: "ordered".

SET\_\*ALFA-RT\* allows to change de value of parameter *\*alfa-rt\** for detecting "DEADENDS" in regression trees.

SET\_\*TTVM\* allows to change de value of parameter *\*total-variance-min\** for detecting "DEADENDS" in regression trees.

SET\_\*V-MIN\* allows to change de value of parameter *\*v-min\** for detecting "LEAVES" in regression trees.

## 5.2.3 Other features

BEST\_FIRST changes order of node development in tree growing (either depth-first or best first).

SET\_\*C-MAX-GROW\* Fixes an upper bound on tree complexity which is active only in the best first strategy.

TEST\_TREE This command tests the *\*current-dt\** (either a decision or a regression tree), using the presently selected test set and the appropriate method.

PRUNE\_TREE prunes the *\*current-dt\** (either a decision or a regression tree), using the backward pruning method. Before using this option you must test the tree.

SET\_\*C-MAX-PRUNE\* fixes an upper bound on tree complexity which is active only in the tree pruning option.

DRW\_PR\_SEQ provides a graphic of pruning sequences curves for both forward and backward pruning.

DESCRIBE\_TREE outputs a description of the *\*current-dt\**.

DISPLAY\_TREE generates a postscript file and displays it using the postscript previewer (normally "ghostview"). The tree is put on a single page.

MP\_DISPLAY\_TREE generates a postscript file and displays it using the postscript previewer "ghostview". The tree is put on several pages if necessary. This is useful for very large trees.

DRAW\_TEST\_SET allows to toggle the drawing of the test set (lower part of node boxes).

GET\_DT\_ERRORS allows to select as *\*learning-set\** the test objects mis-classified by the tree in the last test. If the tree has not yet been tested, the result is nil. Note that for regression trees, these are the states for which the output as approximated by the tree is different from the *\*goal-regression\** used to build the tree, which is often a very large set. Thus the option is not very useful for regression trees.

## 5.3 Linear regression

The menu LINEAR\_REGRESSION gives you access to the standard and generalized linear regression learning.

### 5.3.1 learn\_linear\_regression\_approximation

Builds a linear combination regression of the *\*candidate-attributes\** attributes with respect to *\*goal-regression\**. The method builds a new function attribute representing the linear model.

Note that the list of *\*candidate-attributes\** is expanded by replacing temporal attribute specifications by a list of scalar ones, and filtered to remove attributes which type is not handled by the method. Admissible types are: "ordered numerical".

If the covariance matrix is singular, the method provides some hints on which attributes should be removed in order retry.

SET\_WEIGHT\_DECAY allows to change the weight-decay term used for linear regression models. A small non-zero value prevents the correlation matrix from being singular.

### 5.3.2 tree\_booster

Builds a generalized linear model using small regression trees as interaction terms. The method starts by building a linear regression on the candidate attributes, then it iteratively fits the residual using small regression trees. At the end the whole model is refitted using linear regression on the attributes together with the interaction terms. SET\_BOOSTING\_TREE\_COMPLEXITY modifies the parameter *\*size-of-trees-for-boosting\** which define the trees size.

### 5.3.3 test\_linear\_regression\_approximation

This command prompts for the name of a numerical attribute to compare with *\*goal-regression\**. It generates automatically a two-dimensional scatter-plot representing the *\*test-set\** in terms of these two attributes.

### 5.3.4 hinges

Is not documented in this version.

## 5.4 Non linear regression

The menu NON\_LINEAR\_REGRESSION gives access to the standard non linear regression learning (artificial neural network, multilayer perceptron).



### 5.4.1 `train_mlp_regression/train_mlp_classification`

This command builds a non-linear regression using MLPs and a batch conjugate gradient method. The software was developed by Antonio Munoz of IIT (Universidad Pontifica de Comillas, Madrid, Spain). If the "use\_test\_set" toggle is on, it returns the MLP approximation found during training which obtained the least error on the test set, otherwise it returns the last MLP obtained during training. The criterion used for training is minimum square error without weight-decay term. It produces a new attribute.

### 5.4.2 `set_mlp_hidden_structure`

This command allows to change the structure of the next MLP that will be learned.

### 5.4.3 `monitor_test_set`

This allows to toggle the test-set-error monitoring function during training. This function will be active only if the toggle is on and if the test set is not empty.

### 5.4.4 `mlp_stopping_pars`

This command prompts for the value of the parameters used to decide when to stop the iterative gradient descent for MLPs.

### 5.4.5 `output_layer_activation`

This command prompts for the activation function of the output layer of mlps (hidden layers are always TANH). Output layer may be either linear, TANH or HEAVISIDE.

## 5.5 Similarity

The menu SIMILARITY gives you access to the automatic learning methods based on *similarity* computations between objects: "K-Nearest-Neighbors" and "K-Means". Both methods use the following status variables:

- `SET_KNN_REFERENCE_SET` defines the set of objects to be used as a learning set using the variable `*knn-reference-set*`.
- `*knn-attributes*`: the set of attributes (and normalizations) used to define the euclidean distance.

### 5.5.1 K nearest neighbors

`SELECT_KNN_REFERENCE_SET` selects the `*knn-reference-set*` (same method than to select a `*learning-set*`)

`NORMALIZE_KNN_ATTRIBUTES` starting with the current value of `*candidate-attributes*` builds a list of attributes `*knn-attributes*` suitable for distance computations (on numerical attributes).

Again temporal attributes are replaced by lists of numerical ones. Then they are normalized by computing their standard-deviation in the `*knn-reference-set*`.

`SET_KNN_OUTPUT` allows to choose an attribute (scalar symbolic or numerical), which will be used as output-variable for the "KNN" method. For a symbolic output the method uses majority voting among the `*knn-k*` nearest neighbors; for a numerical output interpolation by the inverse of the distance squared.

`SET_KNN_K` sets the number of neighbors `*knn-k*` (default 1) which are effectively used. The maximum value is set by the parameter "knn-k-max" (default value 15) which can be increased at the expense of higher computing effort (KNN is a slow method).

`HYBRID_DT_KNN` allows you to inherit in a single step all the parameters `*knn-reference-set*`, `*knn-attributes*` and `*knn-output*` from a previously built decision or regression tree. Note that the `*knn-attributes*` are normalized by the quotient the information quantity they provide in the tree and their standard-deviation computed in the `*knn-reference-set*`. In particular, only the attributes selected by the tree are used.

`KNN_TEST_SET_TEST` compares the approximated output of KNN with the actual one on the `*test-set*`.

`KNN_CROSS-VALIDATION_TEST` applies the "leave-one-out" method to `*knn-reference-set*`, for `*knn-k*` increasing from 1 to "knn-k-max" and automatically sets `*knn-k*` to the value which yielded the best accuracy. Note that the algorithm is quadratic (computationally) in the size of the `*knn-reference-set*`.

`KNN_STATISTICS` uses `*knn-reference-set*` to compute the statistics of distances between objects and their `knn-k-max` nearest neighbors and generates conditional histograms (if `*knn-output*` is symbolic) scatter plots otherwise. This option may be useful to design distance rejection criteria for the method.

`FIND_NEAREST_NEIGHBORS` selects in the variable `*learning-set*` the `knn-k-max` nearest neighbors of an object which name is prompted for. These objects can then be viewed for inspection.

## 5.5.2 K-means

`SET_KMEANS_K` sets the value of `*k-means-k*`, the number of clusters built by the K-MEANS algorithm.

`RUN_KMEANS` runs the K-MEANS algorithm on the presently selected `*knn-reference-set*` and with the presently defined metric from `*knn-attributes*`

`DRAW_CLUSTERS` generates automatically all  $\frac{n(n-1)}{2}$  two-dimensional scatter plots of the clusters taking the `*candidate-attributes*` two-by-two<sup>1</sup>.

---

<sup>1</sup>Don't try this if the number of attributes is high, unless you have a lot of time.