4.900

Case-based Reasoning

by Rebecca Shalfield

Case-Based Reasoning

21 July, 2004

Logic Programming Associates Ltd.
Studio 30
The Royal Victoria Patriotic Building
Trinity Road
London SW18 3SX
England

phone: +44 (0) 20 8871 2016
fax: +44 (0) 20 8874 0449
web site: http://www.lpa.co.uk

CONTENTS

CONTENTS

CONTENTS

Chapter 1 – Introduction

Introduction

We have a possibly large number of records in a database. The task is to specify an initial input query and then attempt to retrieve a 'manageable' set of records that are 'close' (i.e. inexact or partial matches). We can then, having cached those records, sort and resort them according to weights.

We want to utilise the power and efficiency of the database engine to reduce the scale of the problem. This is achieved by the intelligent generation of hundreds of SQL queries that differ from each other only very slightly.

This example consists of three phases:

- Setting up the input query.
- Retrieving the records (cases) by incrementally relaxing the search criteria for each column.
- Zoom in on the retrieved records to see how close they really are. Present the cases to the user.

Objectives

The objectives are to:

- develop a simple (stand-alone) case-based reasoning tool for educational and teaching purposes.
- provide a case-based reasoning API plus source code to the example for development purposes and integrate it with other tools and languages (such as Visual Basic, Java, etc.).
- showcase the power of the combination of Prolog plus ProData and the generation of SQL queries.

What's Included?

The Case-Based Reasoning toolkit includes:

- A collection of routines, an API, for building CBR solutions

- A source-code example which uses these routines to build a CBR-based application demo
- A way to run that demo as if it were a stand-alone application

Whilst the former is aimed at application developers and researchers interested in building application which have some CBR component, the latter is primarily aimed at educators who are engaged in teaching the basics of CBR.

The LPA CBR toolkit assumes a database of previous records, stored in a relational database. This is typically Access, SQL Server or Oracle. Given a desired record to look for, the CBR toolkit provides routines which will find records deemed close to the desired record. These can then be viewed in order of closenesss with the ability to affect the notion of how closeness is computed.

This user guide starts with the worked example and then documents the API.

Chapter 2 – CBR Demo

This chapter describes a case-based reasoning demonstration program. This demo uses the CBR API, the ProData toolkit and dialogs designed using the Dialog Editor utility.

The StatLog ODBC Data Source

This example assumes you have set up the Access database, StatLog.MDB, as an ODBC data source via the ODBC Administrator.

Loading And Running The Program

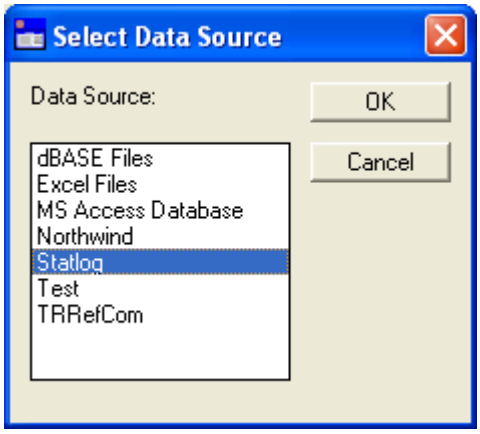
Launch *WIN-PROLOG* and load (via the File\Load menu option) the file CBR_EG.PL from the EXAMPLES\CBR directory; execute the following goal from the Prolog prompt:

```
?- run. <enter>
```

Phase 1 – Setting Up The Input Query

Selecting The Data Source

This dialog lists all the available ODBC data sources and allows you to select one. The underlying code to populate the listbox in this dialog uses ProData.



Select the StatLog data source.

Selecting The Table

This dialog allows you to select the table. The underlying code to populate the listbox in this dialog uses ProData.

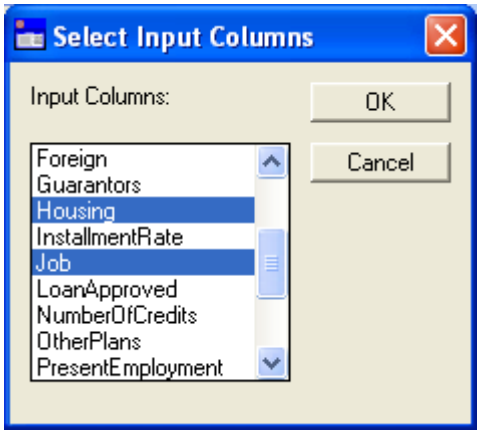
Select the GermanCredit table. GermanCredit is a database of loan application information together with a flag stating whether an individual loan application was approved or not.

Input Columns

An input column is a column that you wish to set an expression for. An input expression is a constraint that any retrieved cases is required to try to satisfy.

Selecting The Input Columns

This dialog allows you to select one or more input columns.



Choose the following columns: CreditAmount, Housing, Job and PurposeOfLoan.

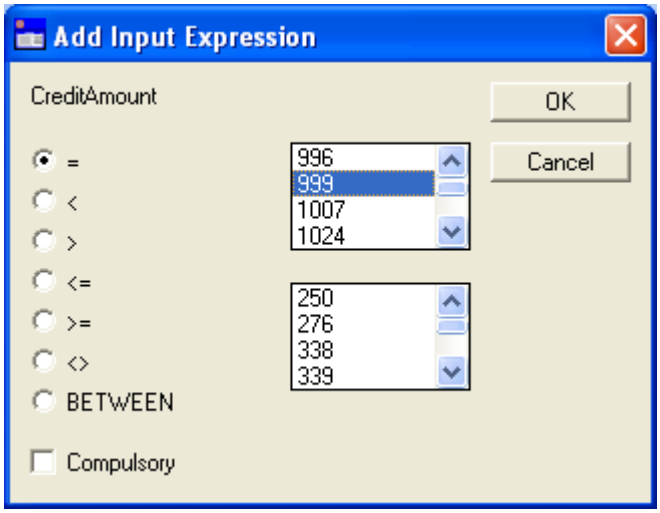
Adding The Input Expressions

There are three types of input columns – continuous, discrete and time. A continuous column contains numerical values. A discrete column contains distinct non-numeric values.

For each of the input columns selected, you will be presented with a dialog allowing you to add its input expression.

For a continuous column, a single numerical value is normally selected.

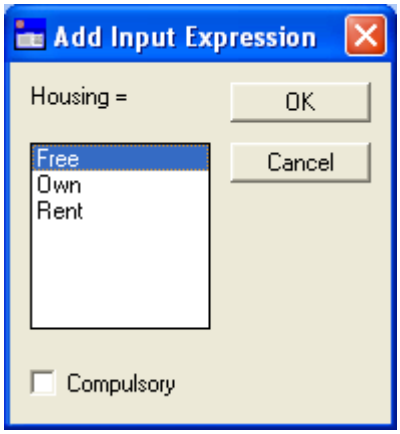
Set the input expression, "CreditAmount = 999".



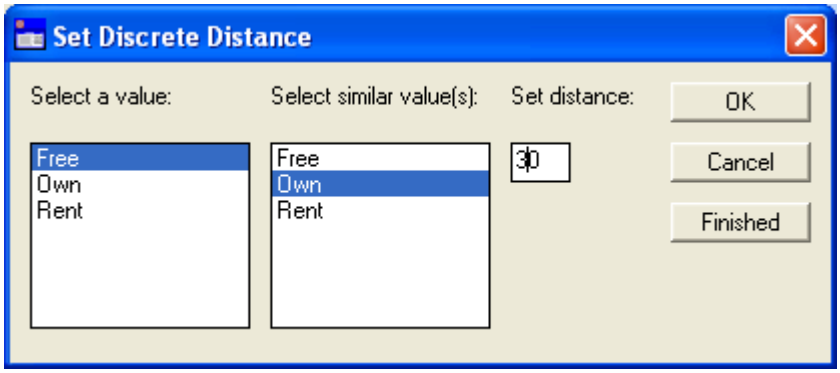
For a discrete column, one or more values may be selected.

You can force a particular input expression to be a constraint that any retrieved cases must satisfy by setting its 'compulsory' checkbox.

Set the input expression, "Housing = 'Free'".



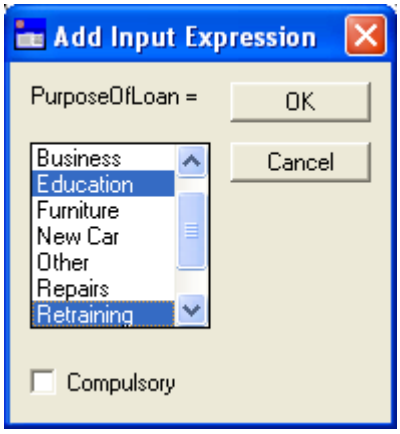
For each discrete column, you are given the option to set the distance between any two discrete values.



Select one discrete value from the first list and one or more from the second list. The default distance is 100, which means they are the maximum distance apart. Enter an integer between 0 (values are identical) and 100 (values are completely different). When you click [OK] the settings will take effect and you will be returned to the Set Distances dialog to set other distances. When you are finished, click the [Finished] button.

This establishes that the discrete values, 'Free' and 'Own', have a distance of 30.

Set the input expression, "PurposeOfLoan IN ('Education','Retraining')" by selecting both 'Education' and 'Retraining' from the listbox.



Set the expression "Job = 'Management'".

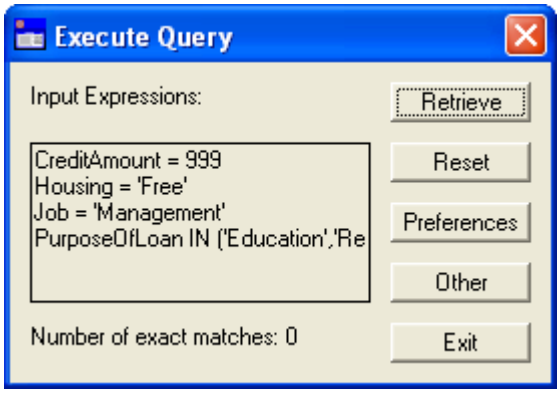
This program also handles date fields where the time part is not being used.

This completes phase 1; you will now be taken to the Execute Query dialog.

Phase 2 – Retrieving The Cases

Executing The Input Query

You have now established the 'input query'. The Execute Query dialog gives you the opportunity to review the input expressions making up the input query and then allows you to execute the input query.



The Execute Query dialog will inform us, given the input expressions, how many (if any) exact matches have been found.

From the Execute Query dialog, you can go in a number of directions:

- The [Retrieve] button performs a search and retrieves the cases (described next). If you're happy with the input query now, click on the [Retrieve] button.
- The [Reset] button takes you to the Reset dialog (described later).
- The [Preferences] button takes you to the Preferences dialog (described later).
- The [Other] button takes you to the Other dialog (described later).
- The [Exit] button will, following confirmation, quit the CBR demo.

Searching

Records are 'matched' by relaxing and/or eliminating parts of the input query until the 'threshold' is met. The threshold is the minimum number of records as set on the Preferences dialog.

For instance, the input `CreditAmount = 999' may be relaxed to 'CreditAmount BETWEEN 817.26 AND 1180.74'.

The input query might have one or more of its input columns eliminated altogether.

In a partly eliminated input query, relaxing can also be applied.

Performing An Exact Or Relaxed Search

Clicking the [Retrieve] button on the Execute Query Dialog will initially perform an exact search. Your SQL query looks as follows:

```
SELECT COUNT(*) FROM GermanCredit WHERE PurposeOfLoan IN
('Education','Retraining') AND CreditAmount = 999 AND
Housing = 'Free' AND Job = 'Management'
```

In case-based reasoning, it is normal for an exact match SQL query to fail. Following the failure of an exact match SQL query, a global default -/+ percentage threshold of 1% through to 100% (this can be lowered on the Preferences dialog) is automatically applied to all columns set as inputs.

An initial relaxed percentage is calculated for each column up front and applied in order to speed up the search.

With a 1% threshold set, our SQL query looks as follows:

```
SELECT COUNT(*) FROM GermanCredit WHERE Housing = 'Free'
AND Job = 'Management' AND PurposeOfLoan =
('Education','Retraining')  AND CreditAmount BETWEEN
817.26 AND 1180.74
```

A -/+ percentage of 1% is calculated on the entire range of the column. The CreditAmount column ranges from 250 to 18424, a span of 18174, so 1% is 181.74; adding 181.74 to 999 gives 1180.74.

The -/+ percentage threshold is also applied to discrete columns where a distance between the given value and any other value has been set. Once that -/+ percentage has been reached during a search, the SQL query will include all values having a distance from the given value <= to the -/+ percentage.

With a 30% threshold set, our SQL query looks as follows:

```
SELECT COUNT(*) FROM GermanCredit WHERE Housing IN
('Free', 'Own') AND Job = 'Management' AND PurposeOfLoan
= ('Education','Retraining')  AND CreditAmount BETWEEN
250 AND 6451.2
```

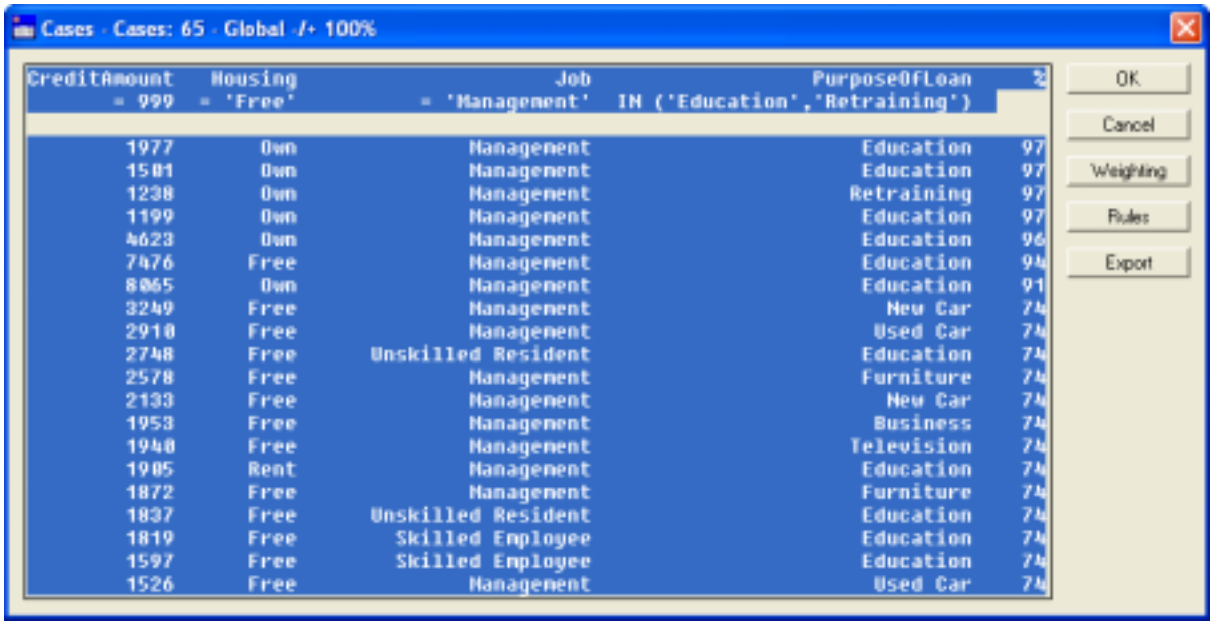
Phase 3 – Viewing The Cases

Sorting The Cases

The retrieved cases are given a score and then sorted into order.

The cases can then be viewed in order, with the 'best' cases at the top.

Viewing The Cases



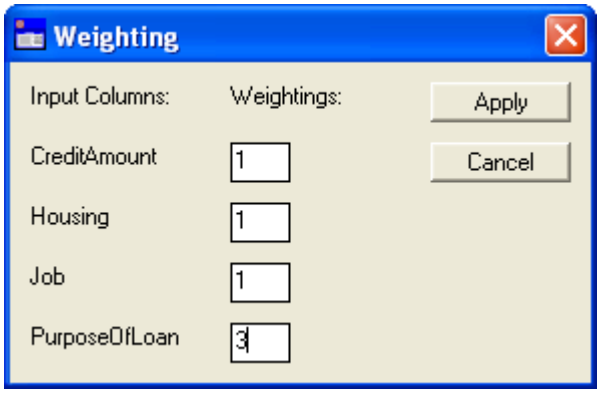
The four columns to which input expressions were set have their expression in the column heading.

If there are duplicate cases, only one will appear. The original records from which the cases were derived may not be duplicates, but the data in the output columns (a subset of the columns in the records) may be. You can sometimes force duplicate cases to appear by increasing the number of output columns.

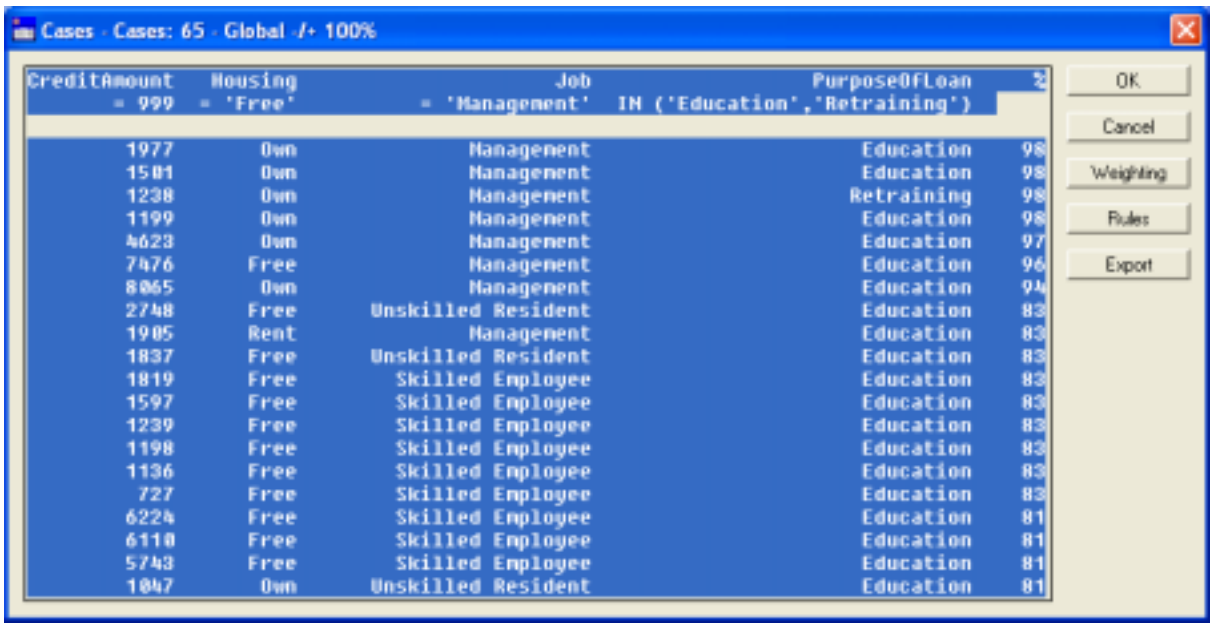
To the right of each case is its percentage as to how closely it matched the 'ideal' (i.e. a case that matched the input expressions exactly).

Weighting The Cases

All input expressions have a default weighting of 1. Such weightings can be changed via the Add Weightings dialog. Clicking the [Add Weights] button on the Cases dialog takes you to the Add Weightings dialog.



Change the weighting of 'PurposeOfLoan' from 1 to 3 and click on the [Apply] button. The cases will be sorted and displayed again in the Cases dialog.

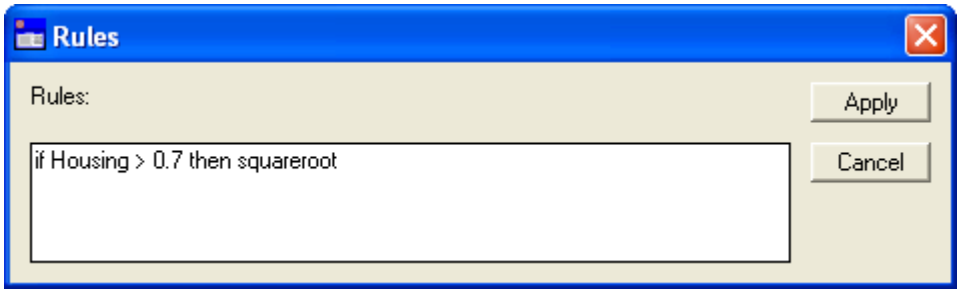


You can see that the cases with a PurposeOfLoan equal to Education or Retraining have come to the top. If you were to scroll the text in the textbox to the left, you would see that the percentages are slightly different than before.

Applying A Rule

Clicking on the [Rules] button will take you to the Rules dialog. Type the following rule into the textbox:

if Housing > 0.7 then squareroot



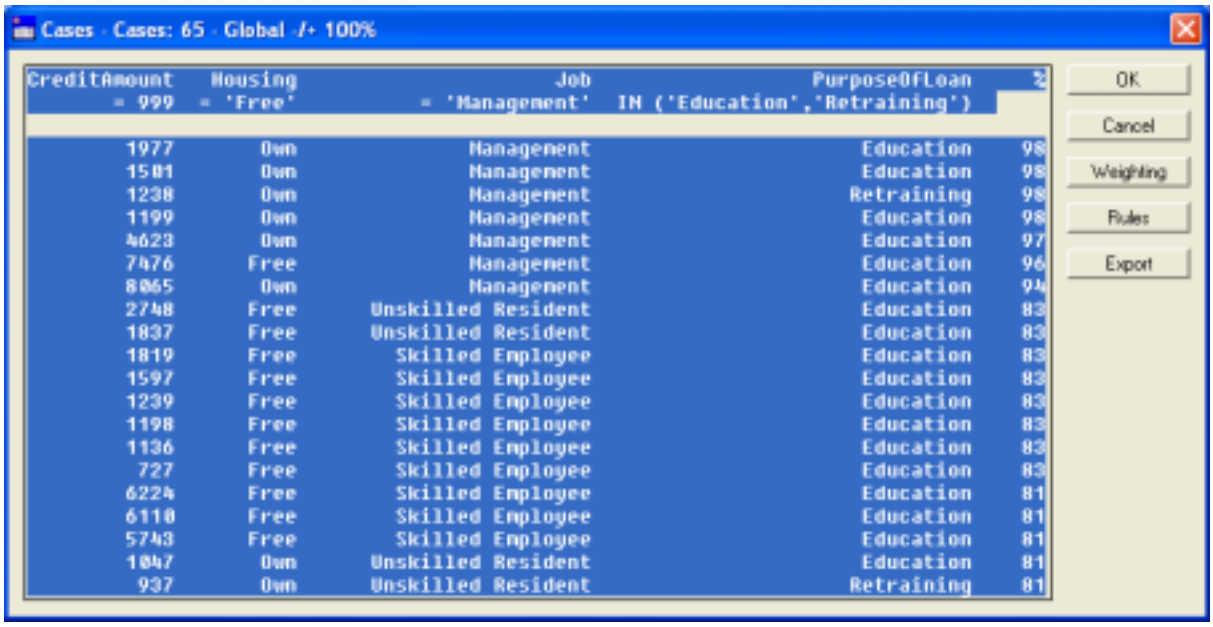
More than one rule can be entered, each on a separate line; for example:

```
IF Housing > 0.7 THEN squareroot
IF CreditAmount < 0.3 THEN square
```

A rule can have one, two or more conditions separated by 'AND'; for example:

```
IF Housing > 0.7 AND CreditAmount > 0.7 THEN cuberoot
```

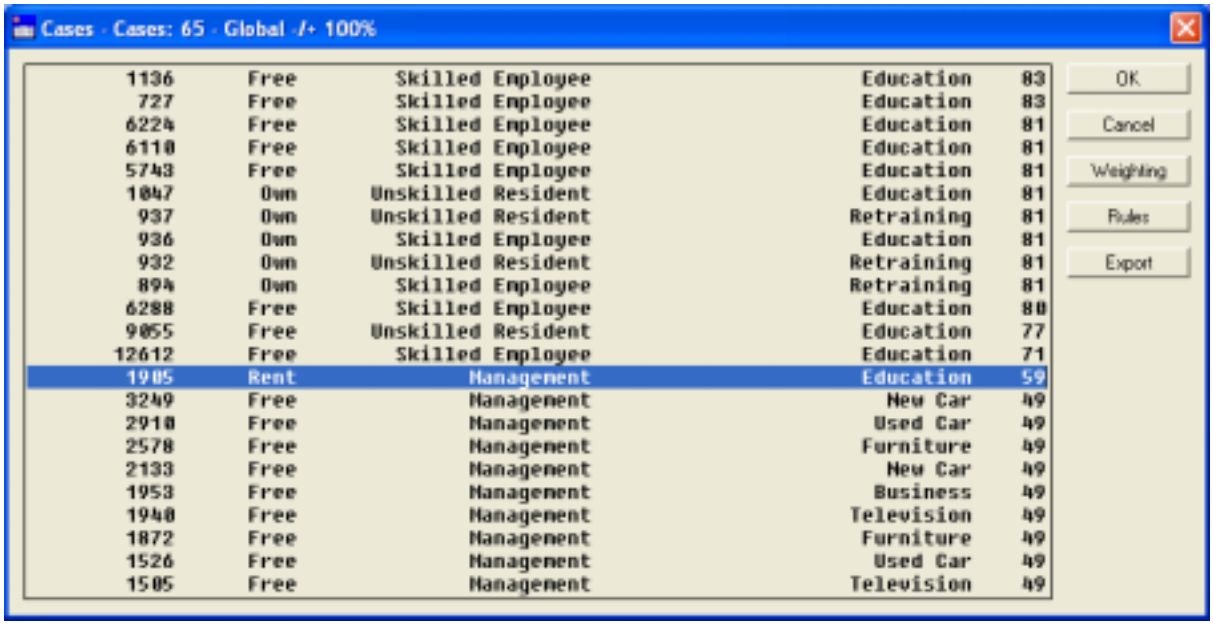
Click on the [Apply] button when ready. The rule will now be applied to the retrieved cases whereupon the Cases dialog will reappear.



You can see that the rule:

1905 Rent Management Education 83

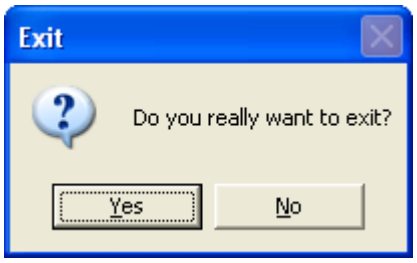
is no longer in the top 20; in fact, it now has a percentage of 59.



1136	Free	Skilled Employee	Education	83
727	Free	Skilled Employee	Education	83
6224	Free	Skilled Employee	Education	81
6110	Free	Skilled Employee	Education	81
5743	Free	Skilled Employee	Education	81
1047	Own	Unskilled Resident	Education	81
937	Own	Unskilled Resident	Retraining	81
936	Own	Skilled Employee	Education	81
932	Own	Unskilled Resident	Retraining	81
894	Own	Skilled Employee	Retraining	81
6288	Free	Skilled Employee	Education	80
9055	Free	Unskilled Resident	Education	77
12612	Free	Skilled Employee	Education	71
1905	Rent	Management	Education	59
3249	Free	Management	New Car	49
2910	Free	Management	Used Car	49
2578	Free	Management	Furniture	49
2133	Free	Management	New Car	49
1953	Free	Management	Business	49
1940	Free	Management	Television	49
1872	Free	Management	Furniture	49
1526	Free	Management	Used Car	49
1505	Free	Management	Television	49

Exiting The Program

Clicking the [Exit] button on the Execute Query dialog will take you to the following dialog:

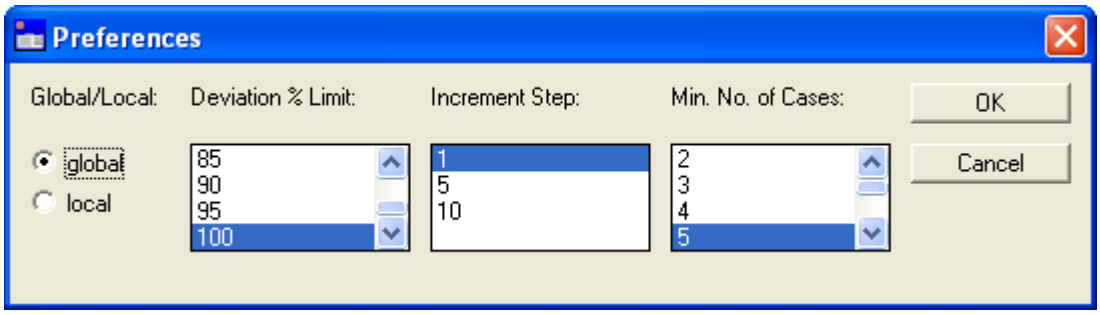


Clicking [Yes] will exit the program; clicking [No] will return you to the Execute Query dialog.

Chapter 3 – Additional Features of the CBR Demo

Increasing the Minimum Number of Cases Threshold

The Preferences dialog allows you to increase the minimum number of cases so that you can get back the minimum you require. Click on the [OK] button on the Cases dialog, then click on the [Preferences] button on the Execute Query dialog.



Once the minimum number of cases settings has been altered, you will need to click on the [Retrieve] button on the Execute Query dialog again to redo the search and retrieve a revised set of cases.

Input Expressions

When setting a BETWEEN X AND Y input expression, both the lower and upper value may be selected, one from each listbox.

Exporting The Cases

The cases can be exported as a CSV file named CBRCASES.PL.

The Elimination Search

If, through relaxing, still no or insufficient cases can be matched, you will proceed into performing an elimination search.

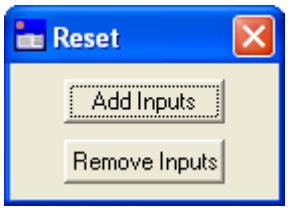
Performing An Elimination Search

An elimination search is required when the exact and/or relaxed search is unable to find any or insufficient records. An elimination search tries to match on a subset of the input expressions. If five input expressions have been set, the elimination search will first try to match on each subset of four out of the five input expressions. Once again, the global default -/+ percentage threshold of 1% through to 100% is automatically applied to any input column within each chosen subset.

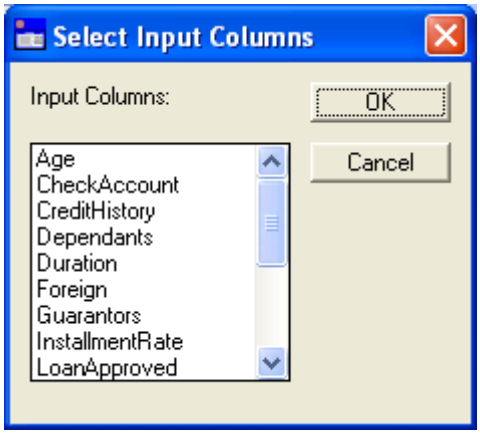
Given five original input expressions, it will first try to match using each combination of four input expressions. Once sufficient records have been found, they will be displayed.

The Reset Dialog

The Reset dialog is reached from the Execute Query dialog.

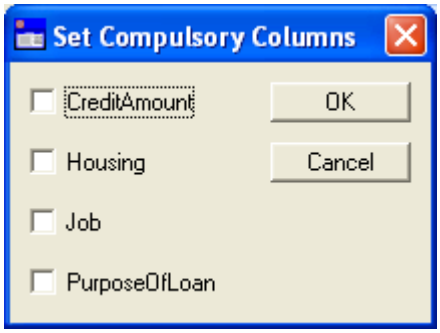


The [Add Inputs] button allows you to add additional input expressions via the Select Input Columns dialog seen earlier.

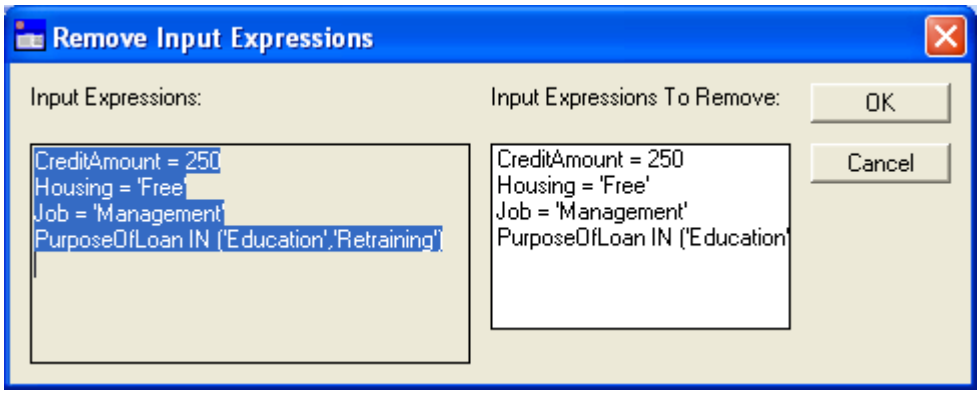


This allows you to set additional input expressions. Any columns already set will not appear in the menu.

The [Compulsory] button leads to the Set Compulsory Columns dialog which allows you to set those input columns that you want to be compulsory. This prevents a column being eliminated during a search.



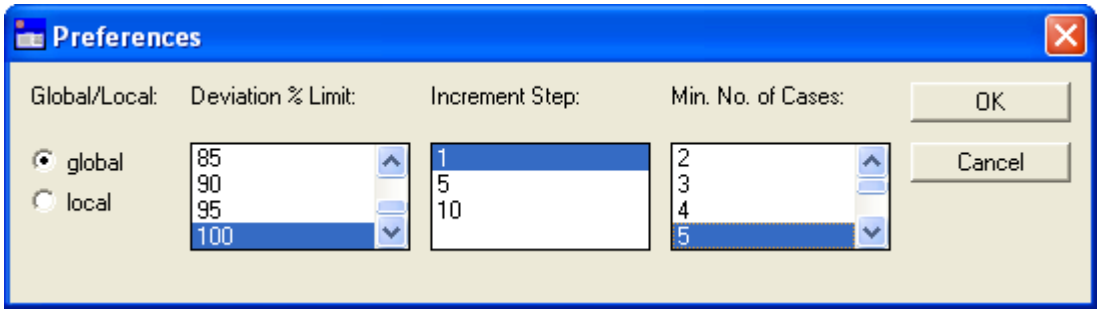
The [Remove Inputs] button takes you to the Remove Input Expressions dialog.



This dialog allows you to remove one or more existing input expressions. This dialog also allows you to view the existing input expressions without removing any of them; just select nothing from the menu and click [OK] when you're ready.

The Preferences Dialog

Clicking the [Preferences] button on the Execute Query dialog takes you to the Preferences dialog.

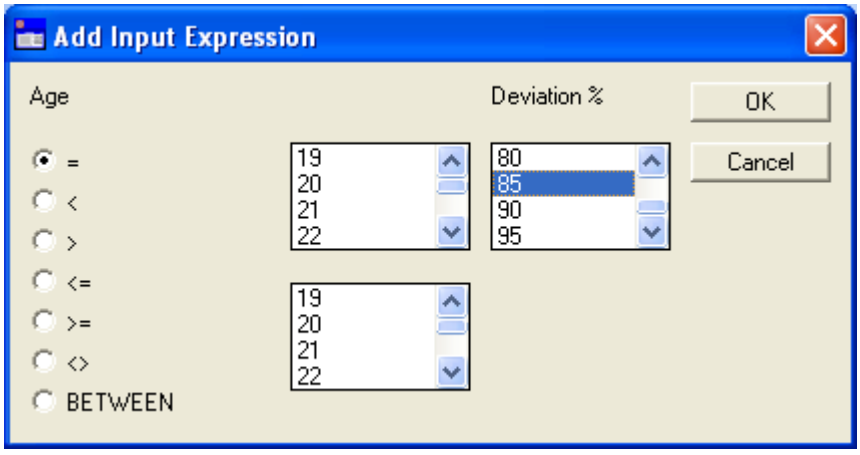


This dialog allows you to increase/decrease the global default -/+ percentage threshold for a column.

It also allows you to increase/decrease the minimum number of cases required to be matched; a search will continue until that number of cases or above have been found or the default -/+ percentage threshold has been reached.

Setting A Local Input Expression

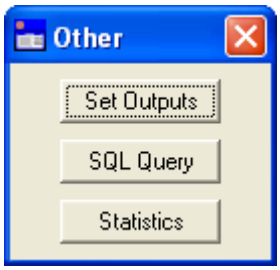
The Preferences dialog allows you to switch from a global default -/+ percentage threshold to a local one. Whenever you set a new input expression, the dialog will be slightly different:



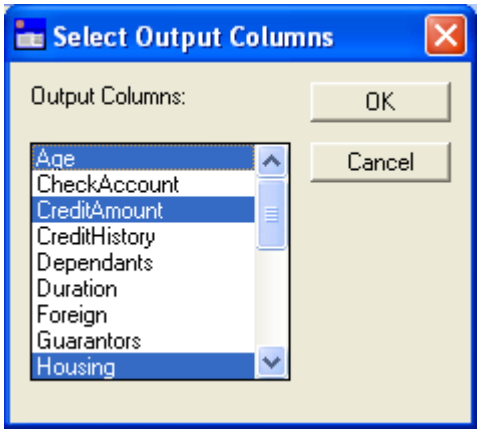
The -/+ % applies to this input expression only.

The Other Dialog

The Other dialog is reached by clicking on the [Other] button on the Execute Query dialog.

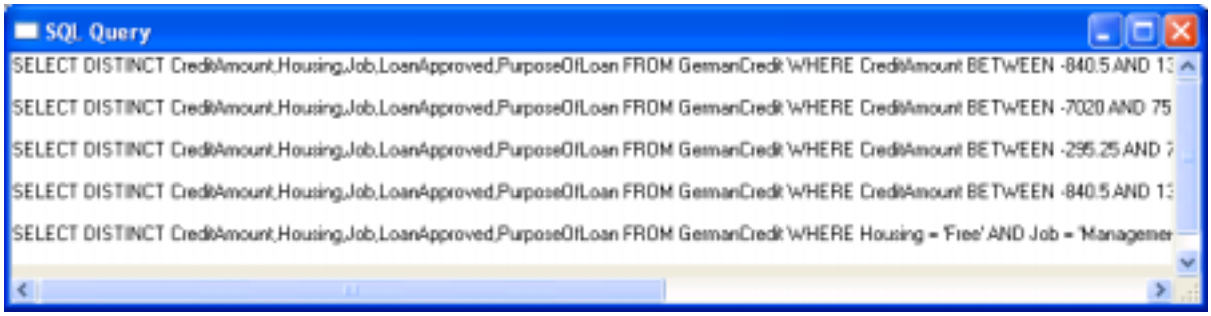


The [Set Outputs] button takes you to the Select Output Columns dialog which allows you to select the output columns. An output column is a column that will be displayed for each retrieved case on output. Although the input columns are selected by default, you can also select zero or more additional columns for output; these columns will be displayed on output even though the data in such columns was not used in the input query.



Viewing The SQL Query

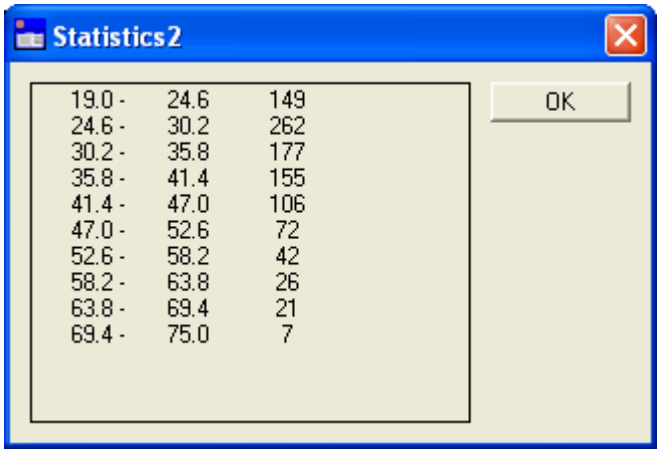
Clicking the [SQL Query] button on the Other dialog will allow you to save the most recently executed SQL queries into a text window.



You will be returned to the Execute Query dialog.

Statistics

The statistics facility, reached via the [Statistics] button on the Other dialog, gives you the counts for each 10% of a continuous range or each discrete value.



The Distancing Algorithm

When an input expression is of the type BETWEEN LOW AND HIGH, the distance is calculated as follows:

- when case's value is >= to LOW and =< to HIGH, a distance of 0 is recorded
- when case's value is less than LOW, the distance is the difference between the case's lower value and LOW
- when case's value is greater than HIGH, the distance is the difference between HIGH and the case's greater value

Chapter 4 – Case-Based Reasoning API

Starting/Stopping

These predicates start and stop the CBR API:

cbr_startup/0
cbr_shutdown/0

ODBC Connection

These predicates connect to and disconnect from an ODBC data source:

cbr_connect/1
cbr_connected/1
cbr_disconnect/0

Tables

These predicates allow you to connect to a table:

cbr_table/1

Columns

These predicates return columns:

cbr_non_input_columns/1

Input Expressions

These predicates allow you to add, view and remove input expressions:

cbr_add_input_expression/1
cbr_add_input_expression/2
cbr_add_input_expression/3
cbr_add_input_expression/4
cbr_input_columns/1
cbr_input_expression/8
cbr_input_expressions/1
cbr_remove_input_expression/1

Column Type

These predicates get or set the data type of a column:

cbr_get_column_type/2
cbr_set_column_type/2

Column Statistics

This predicate gets statistics on a given column:

cbr_column_statistics/2

Column Distincts

These predicates return the distinct values in a column and allow you to set the distance between them:

cbr_column_distincts/3
cbr_add_discrete_distance/4
cbr_discrete_distance/4
cbr_remove_discrete_distance/3

Compulsory Columns

These predicates add and remove the compulsory status for a column:

cbr_add_compulsory/1
cbr_compulsory_columns/1
cbr_remove_compulsory/1

Weighting

This predicate sets the weighting on a column:

cbr_weighting/2

Output Columns

These predicates allow you to add, remove and view the output columns:

cbr_add_output_column/1
cbr_output_columns/1
cbr_remove_output_column/1
cbr_removeall_output_columns/0

Rules

These predicates allow you to add, view and remove rules

cbr_add_rule/1
cbr_add_rule/3
cbr_rules/1
cbr_removeall_rules/0
cbr_remove_rule/1

Thresholds

These predicates allow you to set various thresholds:

cbr_minimum_cases/1
cbr_increment_step/1
cbr_get_deviation_percentage/2
cbr_set_deviation_percentage/2

Cases

These predicates allow you to retrieve and view cases:

cbr_ideal_case/1
cbr_exact_retrieve/1
cbr_retrieve/1
cbr_sql_query/1
cbr_cases/2
cbr_report/2

cbr_add_compulsory/1

add compulsory column status

cbr_add_compulsory(Column)

+Column <string> or <list>

Comments — Add compulsory column status.

Examples — ?- **cbr_add_compulsory(`PurposeOfLoan`).** *<enter>*

Notes — The compulsory status of a column can also be set when it is added via the *cbr_add_input_expression/[2,4]* predicate.

See Also — cbr_compulsory_columns/1

cbr_remove_compulsory/1

cbr_add_discrete_distance/4

add distance

cbr_add_discrete_distance(Column, Discrete1, Discrete2, Distance)

+Column	<string>
+Discrete1	<atom>
+Discrete2	<atom>
+Distance	<integer> in the range {0…100}

Comments — Adds a number representing the distance between two discrete values in the same input column.

Examples — ?- **cbr_add_discrete_distance(`PurposeOfLoan`, 'Education', 'Retraining', 30).** *<enter>*

Notes — A Distance of 0 means the two discrete values are identical; a distance of 100 means the two discrete values are completely different.

See Also — cbr_remove_discrete_distance/3

cbr_discrete_distance/4

cbr_add_input_expression/1

add an input expression

cbr_add_input_expression(Expression)

+Expression <term>

Comments: Add an input expression.

Examples: ?- **cbr_add_input_expression(`PurposeOfLoan` = 'Education').** *<enter>*

Notes: Any existing input expression for the same column will be removed.

Defaults to a deviation_percentage of 100.

All retrieved cases are removed.

See Also: cbr_add_input_expression/4

cbr_input_columns/1

cbr_input_expressions/1

cbr_remove_input_expression/1

cbr_add_input_expression/2

add an input expression

cbr_add_input_expression(Expression, Settings)

+Expression	<term>
+Settings	<list> containing any of {deviation_percentage = <integer>, weighting = <number>, compulsory}

Comments — Add an input expression.

Examples — ?- **cbr_add_input_expression(`PurposeOfLoan` = 'Education', [compulsory]).** *<enter>*

Notes — Any existing input expression for the same column will be removed.

All retrieved cases are removed.

See Also — cbr_add_input_expression/4

cbr_input_columns/1

cbr_input_expressions/1

cbr_remove_input_expression/1

cbr_add_input_expression/3

	add an input expression	
	cbr_add_input_expression(Column, Op, Value)	
	+Column	<string>
	+Op	<string>
	+Value	<list>, <string> or <term>
Comments	Add an input expression.	
Examples	?- **cbr_add_input_expression(`BirthDate`, `=`, date([1965,05,15],[0,0,0])).** *<enter>*	
Notes	Any existing input expression for the same column will be removed.	
	Defaults to a deviation percentage of 100.	
	All retrieved cases are removed.	
See Also	cbr_add_input_expression/4	
	cbr_input_columns/1	
	cbr_input_expressions/1	
	cbr_remove_input_expression/1	

cbr_add_input_expression/4

add an input expression

cbr_add_input_expression(Column, Op, Value, Settings)

+Column	<string>
+Op	<string>
+Value	<string> or <list>
+Settings	<list> containing any of {deviation_percentage = <integer>, weighting = <number>, compulsory}

Comments — Add an input expression.

Examples — ?- **cbr_add_input_expression(`BirthDate`, `=`, date([1965,05,15],[0,0,0]), [weighting = 2]).** *<enter>*

Notes — Any existing input expression for the same column will be removed.

All retrieved cases are removed.

See Also — cbr_add_input_expression/3

cbr_input_columns/1

cbr_input_expressions/1

cbr_remove_input_expression/1

cbr_add_output_column/1

	add an output column
	cbr_add_output_column(Column)
	+Column <string> or <list>
Comments	Add an output column.
Examples	?- **cbr_add_output_column(`Age`).** *<enter>*
	?- **cbr_add_output_column([`Age`,`PurposeOfLoan`]).** *<enter>*
Notes	All retrieved cases are removed.
See Also	cbr_output_columns/1
	cbr_remove_output_column/1
	cbr_removeall_output_columns/0

cbr_add_rule/1

add a rule

cbr_add_rule(Rule)

+Rule <string>

Comments

Add a rule.

Examples

?- **cbr_add_rule(`if PurposeOfLoan >= 0.8 and CreditAmount =< 0.9 then square`).** *<enter>*
yes

?- **cbr_add_rule(`if PurposeOfLoan >= 0.8 and CreditAmount =< 0.9 then power(2)`).** *<enter>*
yes

Notes

The keyword 'if' can be written as 'if', 'If' or 'IF'.

The keyword 'and' can be written as 'and', 'And' or 'AND'.

The keyword 'then' can be written as 'then', 'Then' or 'THEN'.

Leave one space only between each token.

The rule is automatically assigned an ID of 'a' through to 'z'.

See Also

cbr_add_rule/3

cbr_rules/1

cbr_remove_rule/1

cbr_add_rule/3

add a rule

cbr_add_rule(RuleID, If, Then)

+RuleID	<atom>
+If	<list> of (Column, Op, Dist)
	Column <string>
	Op <atom> one of {<,>,=,=<,>=,\=}
	Dist <number> in the range [0...1]
+Then	<term> one of { cube, cuberoot, square, squareroot, nearer(<integer>), power(<integer>), farther(<integer>), root(<integer>) }

Comments — Add a rule.

Examples

?- **cbr_add_rule(r1, [(`PurposeOfLoan`,>,0.8)], square).** *<enter>*
yes

?- **cbr_add_rule(r1, [(`PurposeOfLoan`,<,0.3)], power(3)).** *<enter>*
yes

?- **cbr_add_rule(r2, [(`PurposeOfLoan`,>=,0.8), (`CreditAmount`,=<,0.9)], square).** *<enter>*
yes

Notes

Each column scores a distance between 0 (identical to the ideal) and 1 (totally different from the ideal).

Squaring a column's score promotes the case as the column's revised score becomes nearer to 0.

Squarerooting a column's score demotes the case as the column's revised score becomes nearer to 1.

Given an initial score of 0.81, revised scores will be as follows once a particular 'then' value has been applied:

```
nearer(4) or power(4)                0.43

nearer(3),   power(3) or cube        0.53

nearer(2),   power(2) or square      0.65

initial score                        0.81

farther(2),   root(2) or squareroot  0.90

farther(3),   root(3) or cuberoot    0.93

farther(4) or root(4)                0.94
```

See Also

cbr_add_rule/1

cbr_rules/1

cbr_remove_rule/1

cbr_cases/2

get sorted list of cases

cbr_cases(Settings, Cases)

+Settings	<list> containing any of {
	top_n=<integer>,
	minimum_percentage=<integer>
	apply_rules = {none|all|<list> of <atom>}
	}

-Cases	<variable>

Comments — Get sorted list of cases as retrieved by *cbr_retrieve/1*.

Examples

?- **cbr_cases([], X).** *<enter>*
X = [84 - [7476,'Free','Management','Education',50],
58 - [1238,'Own','Management','Retraining',36],
58 - [1199,'Own','Management','Education',67],
51 - [1164,'Free','Management','Other',51],
50 - [727,'Free','Skilled Employee','Education',46]]

Notes — apply_rules=all is the default.

When given a list of rule IDs, the rules are applied in the order given; only the first successful rule is applied.

See Also

cbr_column_distincts/3

get list of distinct values for a column

cbr_column_distincts(Column, Format, Distincts)

+Column	<string>
+Format	<atom> one of { atom, string }
-Distincts	<variable>

Comments

Get list of distinct values for a column.

Examples

?- **cbr_column_distincts(`PurposeOfLoan`, atom, Distincts).**
<enter>
Distincts = ['Appliance', 'Business', 'Education', 'Furniture', 'New Car', 'Other', 'Repairs', 'Retraining', 'Television', 'Used Car']

Notes

See Also

cbr_column_statistics/2

get column statistics

cbr_column_statistics(Column, Statistics)

+Column	<string> or <variable>
+Statistics	<variable>

Comments

Get column statistics.

Examples

?- **cbr_column_statistics(`PurposeOfLoan`, Stats).**
<enter>
Stats = [(`Appliance`,12), (`Business`,97),
(`Education`,50), (`Furniture`,181), (`New Car`,234),
(`Other`,12), (`Repairs`,22), (`Retraining`,9),
(`Television`,280), (`Used Car`,103)]

Notes

See Also

cbr_compulsory_columns/1

get compulsory columns

cbr_compulsory_columns(CompulsoryColumns)

+CompulsoryColumns <variable>

Comments Get the columns which have their compulsory flag set.

Examples ?- **cbr_compulsory_columns(X).** *<enter>*
X = [`PurposeOfLoan`]

Notes

See Also cbr_add_compulsory/1

cbr_remove_compulsory/1

cbr_connect/1

	connect to the ODBC data source
	cbr_connect(DataSource)
	+DataSource <string>
Comments	Connect to the ODBC data source.
Examples	?- **cbr_startup, cbr_connect(`StatLog`).** *<enter>*
	?- **cbr_startup, cbr_connect(`StatLog;PWD=rhubarb`).** *<enter>*
Notes	
See Also	cbr_connected/1
	cbr_disconnect/0

cbr_connected/1

Check whether you are connected to the given ODBC data source

cbr_connected(DataSource)

+DataSource <string> or <variable>

Comments

Check whether you are connected to the given ODBC data source.

Examples

?- **cbr_connected(X).** *<enter>*
X = `StatLog`

?- **cbr_connected(`StatLog`).** *<enter>*
yes

Notes

See Also

cbr_connect/1

cbr_csv_export/1

export the cases as a CSV file

cbr_csv_export(FileName)

+FileName <atom>

Comments Export the cases as a Comma Separated Value file.

Examples ?- **cbr_csv_export('c:\temp\cbrcases.csv').** *<enter>*

Notes

See Also

cbr_disconnect/0

	disconnect from the current ODBC data source.
	cbr_disconnect
Comments	Disconnect from the current ODBC data source.
Examples	?- **cbr_disconnect.** *<enter>*
Notes	
See Also	cbr_connect/1

cbr_discrete_distance/4

get distance between two discrete values in a column

cbr_discrete_distance(Column, Discrete1, Discrete2, Distance)

+Column	<string>
+Discrete1	<variable> or <atom>
+Discrete2	<variable> or <atom>
-Distance	<variable>

Comments

Get distance between two discrete values in a column.

Examples

?- **cbr_add_discrete_distance(`PurposeOfLoan`, 'Education', 'Retraining', 30).** *<enter>*

?- **cbr_discrete_distance(`PurposeOfLoan`, 'Education', 'Retraining', Distance).** *<enter>*
Distance = 30

Notes

See Also

cbr_exact_retrieve/1

	Perform an exact search and return the number of exact cases found cbr_exact_retrieve(CaseCount) -CaseCount <variable>
Comments	Perform an exact search and return the number of exact cases found.
Examples	?- **cbr_exact_retrieve (CaseCount).** *<enter>* CaseCount = 1
Notes	The cases themselves are asserted into Prolog's memory and can be retrieved via the *cbr_cases/1* predicate.
See Also	

cbr_get_column_type/2

get column type

cbr_get_column_type(Column, Type)

+Column	<string>
?Type	<variable> or <atom> one of {continuous,discrete,time}

Comments Get column type.

Examples ?- **cbr_get_column_type(`PurposeOfLoan`, Type).** *<enter>*
Type = discrete

Notes

See Also cbr_set_column_type/2

cbr_get_deviation_percentage/2

	get the -/+ percentage
	cbr_get_deviation_percentage(GlobalLocal, Percentage)
	-GlobalLocal <variable> or <string>
	-Percentage <variable> or <integer>
Comments	Get the -/+ percentage.
Examples	?- **cbr_get_deviation_percentage(GlobalLocal, Percentage).** *<enter>* GlobalLocal = global , Percentage = 100
Notes	
See Also	cbr_set_deviation_percentage/2

cbr_ideal_case/1

	get the ideal case
	cbr_ideal_case(IdealCase)
	-IdealCase <variable>
Comments	Get the ideal case.
Examples	?- **cbr_ideal_case (IdealCase).** *<enter>* IdealCase = [999, 'Free', 'Management', ['Education','Retraining'], _31870]
Notes	A variable will appear where an output column is not also an input column.
See Also	

cbr_increment_step/1

get or set the increment step

cbr_increment_step(IncrementStep)

	?IncrementStep	<variable> or <integer> >= 1

Comments

Get or set the increment step.

Examples

?- **cbr_increment_step(X).** *<enter>*
X = 1

?- **cbr_increment_step(5).** *<enter>*

Notes

The default increment step is 1.

See Also

cbr_input_columns/1

get the input columns

cbr_input_columns(Columns)

-Columns <variable>

Comments Get the input columns.

Examples ?- **cbr_input_columns(Columns).** *<enter>*
Columns = [`PurposeOfLoan`]

Notes

See Also cbr_input_expressions/1

cbr_input_expression/8

get an input expression

cbr_input_expression(InputColumn, Op, Value, Expression, EndOfExpression, PlusMinusPercent, Min, Max)

?InputColumn	<variable> or <string>
?Op	<variable> or <atom>
?Value	<variable>, <number>, <list> or <string>
?Expression	<variable> or <string>
?EndOfExpression	<variable> or <string>
?PlusMinusPercent	<variable> or <number>
?Min	<variable> or <number>
?Max	<variable> or <number>

Comments

Get an input expression.

Examples

?- **cbr_input_expression(`PurposeOfLoan`, A,B,C,D,E,F,G).**
<enter>
A = `=` ,
B = 'Education' ,
C = `PurposeOfLoan = 'Education'` ,
D = `= 'Education'` ,
E = 100 ,
F = G = 1

Notes

See Also

cbr_input_columns/1

cbr_input_expressions/1

cbr_input_expressions/1

get the input expressions

cbr_input_expressions(Expressions)

-Expressions <variable>

Comments Get the input expressions.

Examples ?- **cbr_input_expressions(X).** *<enter>*
X = [`PurposeOfLoan = 'Education'`]

Notes

See Also cbr_input_columns/1

cbr_minimum_cases/1

	get or set the minimum number of cases required
	cbr_minimum_cases(MinCases)
	?MinCases <variable> or <number>
Comments	Get or set the minimum number of cases required.
Examples	?- **cbr_minimum_cases(5).** <*enter*>
	?- **cbr_minimum_cases(X).** <enter> X = 5
Notes	The default number of minimum cases is 20 as set by *cbr_startup*.
See Also	

cbr_non_input_columns/1

get those columns which are not input columns

cbr_non_input_columns(Columns)

-Columns <variable>

Comments

Get those columns in the current table which are not input columns.

Examples

?- **cbr_input_columns(In), cbr_non_input_columns(NonIn).**
<enter>
In = [`PurposeOfLoan`] ,
NonIn = [`SavingsAccount`, `PresentEmployment`, `InstallmentRate`, `StatusSex`, `Guarantors`, `PresentResidence`, `Property`, `Age`, `OtherPlans`, `Housing`, `NumberOfCredits`, `Job`, `Dependants`, `Telephone`, `Foreign`, `LoanApproved`, `CheckAccount`, `Duration`, `CreditHistory`, `CreditAmount`]

Notes

See Also

cbr_input_columns/1

cbr_output_columns/1

get the output columns

cbr_output_columns(Columns)

+Columns <variable>

Comments Get the output columns.

Examples ?- **cbr_output_columns(X).** <*enter*>
X = [`PurposeOfLoan`]

Notes

See Also cbr_add_output_column/1

cbr_remove_output_column/1

cbr_removeall_output_columns/0

cbr_removeall_output_columns/0

	remove all output columns that are not input columns
	cbr_removeall_output_columns
Comments	Remove all output columns that are not input columns.
Examples	?- **cbr_removeall_output_columns.** *<enter>*
Notes	This predicate removes, on mass, any output columns that are not input columns.
See Also	cbr_add_output_column/1
	cbr_output_columns/1
	cbr_remove_output_column/1

cbr_remove_compulsory/1

	remove compulsory column status
	cbr_remove_compulsory(Column)
	+Column <string> or <list>
Comments	Remove compulsory status from a column.
Examples	?- **cbr_remove_compulsory(`PurposeOfLoan`).** <*enter*> yes
Notes	
See Also	cbr_add_compulsory/1
	cbr_compulsory_columns/1

cbr_remove_discrete_distance/3

remove distance

cbr_remove_discrete_distance(Column, Discrete1, Discrete2)

+Column	<string>
+Discrete1	<atom>
+Discrete2	<atom>

Comments Remove the distance set between two discrete values in the same column.

Examples ?- **cbr_remove_discrete_distance(`PurposeOfLoan`, 'Education', 'Retraining').** *<enter>*
yes

Notes

See Also cbr_add_discrete_distance/4

cbr_discrete_distance/4

cbr_remove_input_expression/1

remove input expression

cbr_remove_input_expression(Column)

+Column <string>

Comments Remove input expression.

Examples ?- **cbr_remove_input_expression(`PurposeOfLoan`).** *<enter>*

Notes The compulsory status and weighting for the column are also removed.

All retrieved cases are also removed.

See Also cbr_add_input_expression/3

cbr_add_input_expression/4

cbr_input_expression/8

cbr_input_expressions/1

cbr_input_columns/1

cbr_remove_output_column/1

	remove an output column
	cbr_remove_output_column(Column)
	+Column <string> or <list>
Comments	Remove an output column.
Examples	?- **cbr_remove_output_column(`Age`).** *<enter>*
Notes	A removed output column will still appear if also an input column.
	All retrieved cases are also removed.
See Also	cbr_add_output_column/1
	cbr_output_columns/1
	cbr_removeall_output_columns/0

cbr_removeall_rules/0

	Remove all rules
	cbr_removeall_rules
Comments	Remove all rules.
Examples	?- **cbr_removeall_rules.** <*enter*>
Notes	
See Also	cbr_add_rule/1
	cbr_add_rule/3
	cbr_rules/1
	cbr_remove_rule/1

cbr_remove_rule/1

	Remove a rule
	cbr_remove_rule(RuleID)
	-RuleID <atom>
Comments	Remove a rule.
Examples	?- **cbr_remove_rule(r1).** <*enter*>
Notes	
See Also	cbr_add_rule/1
	cbr_add_rule/3
	cbr_rules/1
	cbr_removeall_rules/0

cbr_report/2

output a formatted report containing the required cases

cbr_report(Settings, Report)

+Settings	<list> containing any of { top_n=<integer>, minimum_percentage=<integer>, apply_rules={none|all|<list> of <atom>} }
-Report	<variable>

Comments — Output a formatted report containing the required cases.

Examples — ?- **cbr_report([top_n = 10, minimum_percentage = 75], Report).** *<enter>*

Notes — apply_rules=all is the default.

When given a list of rule IDs, the rules are applied in the order given; only the first successful rule is applied.

See Also

cbr_retrieve/1

	Perform a search and return the number of cases found
	cbr_retrieve(CaseCount)
	-CaseCount <variable>
Comments	Perform a search and return the number of cases found.
Examples	?- **cbr_retrieve (CaseCount).** *<enter>*
Notes	The cases themselves are asserted into Prolog's memory and can be retrieved via the *cbr_cases/1* predicate.
See Also	

cbr_rules/1

Get the current rules

cbr_rules(Rules)

-Rules <variable>

Comments: Get the current rules.

Examples:

?- **cbr_rules(Rules).** <*enter*>
Rules = [
(r1,[(`PurposeOfLoan`,>,0.8)],square),
(r2,[(`PurposeOfLoan`,>=,0.8),(`CreditAmount`,=<,0.9)],
square)
]

Notes

See Also: cbr_add_rule/3

cbr_remove_rule/1

cbr_set_column_type/2

set column type

cbr_set_column_type(Column, Type)

+Column <string>

+Type <atom> one of { continuous, discrete, time }

Comments Set column type.

Examples ?- **cbr_set_column_type(`LoanApproved`, discrete).** *<enter>*

Notes This predicate allows you, for example, to force a continuous column to be treated as a discrete column.

See Also cbr_get_column_type/2

cbr_set_deviation_percentage/2

set the -/+ percentage

cbr_set_deviation_percentage(GlobalLocal, DeviationPercentage)

+GlobalLocal	<atom> one of {global,local}
+DeviationPercentage	<integer> in the range {0...100}

Comments

The first argument should be set to global if you want all input expressions to be relaxed by the same -/+ percentage; the second argument is the threshold deviation percentage.

The first argument should be set to local if you want each input expressions to use its own local deviation percentage threshold. In such mode, the second argument is ignored, only being used when adding an input expression where the deviation_percentage is not specified.

Examples

?- **cbr_set_deviation_percentage(global, 100).**
<enter>

Notes

See Also

cbr_get_deviation_percentage/2

cbr_shutdown/0

	shutdown the CBR API
	cbr_shutdown
Comments	Shutdown the CBR API.
Examples	?- **cbr_shutdown.** <*enter*>
Notes	
See Also	cbr_startup/0

cbr_sql_query/1

get each SQL queries generated during the last search

cbr_sql_query(SQLQuery)

-SQLQuery <variable>

Comments Get each SQL queries, through backtracking, generated during the last search.

Examples ?- **cbr_sql_query(X).** <*enter*>
X = `SELECT DISTINCT PurposeOfLoan FROM GermanCredit WHERE PurposeOfLoan = 'Education'`

Notes

See Also

cbr_startup/0

	initialises the CBR API
	cbr_startup
Comments	Initialises the CBR API.
Examples	?- **cbr_startup.** *<enter>*
Notes	The minimum number of cases is set to 20.
See Also	cbr_shutdown/0

cbr_table/1

attach to a table or get the name of the current table

cbr_table(Table)

?Table <string> or <variable>

Comments Attach to a table or get the name of the current table.

Examples ?- **cbr_table(`GermanCredit`).** *<enter>*

?- **cbr_table(X).** <enter>
X = `GermanCredit`

Notes

See Also

cbr_weighting/2

	Get or set the weighting on a column	
	cbr_weighting(Column, Weighting)	
	+Column	<string>
	?Weighting	<number>
Comments	Get or set the weighting on a column.	
Examples	?- **cbr_weighting(`PurposeOfLoan`, 2).** *<enter>*	
Notes	The default weighting for a column is 1.	
	The weighting for a column can also be set when added via the *cbr_add_input_expression/[2,4]* predicate.	
	The weighting on a column is ignored when applying a rule.	
See Also		