

Introduction to Clementine[®]

30032-001

For more information about SPSS® software products, please visit our Web site at <http://www.spss.com> or contact

SPSS Inc.
233 South Wacker Drive, 11th Floor
Chicago, IL 60606-6412
Tel: (312) 651-3000
Fax: (312) 651-3668

SPSS is a registered trademark and its other product names are the trademarks of SPSS Inc. for its proprietary computer software. No material describing such software may be produced or distributed without the written permission of the owners of the trademark and license rights in the software and the copyrights in the published materials.

The SOFTWARE and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (c)(1)(ii) of The Rights in Technical Data and Computer Software clause at 52.227-7013. Contractor/manufacturer is SPSS Inc., 233 South Wacker Drive, 11th Floor, Chicago, IL 60606-6412.

TableLook is a trademark of SPSS Inc.
Windows is a registered trademark of Microsoft Corporation.
DataDirect, DataDirect Connect, INTERSOLV, and SequeLink are registered trademarks of MERANT Solutions Inc.
Portions of this product were created using LEADTOOLS © 1991-2000, LEAD Technologies, Inc. ALL RIGHTS RESERVED.
LEAD, LEADTOOLS, and LEADVIEW are registered trademarks of LEAD Technologies, Inc.
Portions of this product were based on the work of the FreeType Team (<http://www.freetype.org>).

General notice: Other product names mentioned herein are used for identification purposes only and may be trademarks or registered trademarks of their respective companies in the United States and other countries.

Introduction to Clementine
Copyright © 2003 by SPSS Inc.
All rights reserved.
Printed in the United States of America.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Introduction to Clementine

Table of Contents

CHAPTER 1

INTRODUCTION TO DATA MINING

INTRODUCTION TO DATA MINING	1-1
IS DATA MINING APPROPRIATE?	1-1
A STRATEGY FOR DATA MINING: THE CRISP-DM PROCESS METHODOLOGY	1-2
PLAN OF THE COURSE.....	1-5

CHAPTER 2

INTRODUCING CLEMENTINE

CLEMENTINE AND CLEMENTINE SERVER.....	2-2
STARTING CLEMENTINE	2-2
USING THE MOUSE	2-4
VISUAL PROGRAMMING	2-4
BUILDING STREAMS WITH CLEMENTINE.....	2-8
GETTING HELP	2-10

CHAPTER 3

READING DATA FILES

READING DATA FILES INTO CLEMENTINE	3-1
READING DATA FROM FREE-FIELD TEXT FILES	3-2
FIRST CHECK ON THE DATA	3-8
READING SPSS DATA FILES.....	3-13
READING DATA USING ODBC	3-16
OTHER DATA FORMATS	3-22
DEFINING DATA FIELD TYPES	3-23
FIELD DIRECTION	3-26
SAVING A CLEMENTINE STREAM.....	3-27
APPENDIX: READING DATA FROM FIXED-FIELD TEXT FILES	3-27

CHAPTER 4

DATA QUALITY

MISSING DATA IN CLEMENTINE	4-2
ASSESSING DATA QUALITY	4-2
OPENING A STREAM FILE	4-9
DATA AUDIT	4-10

CHAPTER 5**INTRODUCTION TO DATA MANIPULATION**

A BRIEF INTRODUCTION TO THE CLEM LANGUAGE.....	5-2
RECORD OPERATIONS AND THE SELECT NODE.....	5-3
FIELD OPERATIONS AND THE FILTER NODE.....	5-6
FIELD REORDERING.....	5-9
THE DERIVE NODE.....	5-10
RECLASSIFY NODE.....	5-18
EXECUTING FIELD OPERATION NODES SIMULTANEOUSLY.....	5-20
AUTOMATICALLY GENERATING OPERATIONAL NODES.....	5-21

CHAPTER 6**LOOKING FOR RELATIONSHIPS IN DATA**

STUDYING RELATIONSHIPS BETWEEN SYMBOLIC FIELDS.....	6-2
MATRIX NODE: RELATING TWO SYMBOLIC FIELDS.....	6-2
THE WEB NODE.....	6-5
CORRELATIONS BETWEEN NUMERIC FIELDS.....	6-12

CHAPTER 7**MODELING TECHNIQUES IN CLEMENTINE**

NEURAL NETWORKS.....	7-2
RULE INDUCTION.....	7-3
STATISTICAL PREDICTION MODELS.....	7-4
LINEAR REGRESSION.....	7-4
LOGISTIC REGRESSION.....	7-5
PRINCIPAL COMPONENTS.....	7-6
CLUSTERING.....	7-6
KOHONEN NETWORKS.....	7-6
K-MEANS CLUSTERING.....	7-7
TWO-STEP CLUSTERING.....	7-7
ASSOCIATION RULES.....	7-7
SEQUENCE DETECTION.....	7-8
WHICH TECHNIQUE, WHEN?.....	7-8

CHAPTER 8**NEURAL NETWORKS**

THE NEURAL NETWORK NODE.....	8-1
MODELS PALETTE.....	8-7
UNDERSTANDING THE NEURAL NETWORK.....	8-8
CREATING A DATA TABLE CONTAINING PREDICTED VALUES.....	8-9
COMPARING PREDICTED TO ACTUAL VALUES.....	8-10
UNDERSTANDING THE REASONING BEHIND THE PREDICTIONS.....	8-15
SAVING THE STREAM.....	8-18
MODEL SUMMARY.....	8-19

CHAPTER 9**RULE INDUCTION**

RULE INDUCTION IN CLEMENTINE	9-1
RULE INDUCTION USING C5.0	9-2
BROWSING THE MODEL	9-6
GENERATING AND BROWSING A RULE SET	9-10
UNDERSTANDING THE RULE AND DETERMINING ACCURACY	9-12
UNDERSTANDING THE MOST IMPORTANT FACTORS IN PREDICTION	9-16

CHAPTER 10**COMPARING AND COMBINING MODELS**

COMPARING MODELS	10-1
ANALYSIS NODE	10-3
EVALUATION CHARTS FOR MODEL COMPARISON	10-4
COMPARING MODELS USING VALIDATION DATA	10-4
USING RULE INDUCTION WITH NEURAL NETWORKS	10-7

CHAPTER 11**KOHONEN NETWORKS**

KOHONEN NODE	11-1
UNDERSTANDING THE KOHONEN NETWORK	11-5
FOCUSING ON THE MAIN SEGMENTS	11-9
CREATING A REFERENCE VALUE FOR EACH CLUSTER GROUP	11-10
USING OTHER FIELDS TO BUILD PROFILES	11-12
APPENDIX: VIEWING CLUSTERS IN A SCATTERPLOT	11-14

CHAPTER 12**ASSOCIATION RULES**

THE APRIORI NODE	12-2
USING THE ASSOCIATIONS	12-7

CHAPTER 13**SEQUENCE DETECTION**

DATA ORGANIZATION FOR SEQUENCE DETECTION	13-3
SEQUENCE NODE VERSUS CAPRI	13-3
THE SEQUENCE NODE	13-4
EXPLORING SEQUENCES	13-7
MODEL PREDICTIONS	13-10
SUMMARY	13-12

CHAPTER 14**OTHER TOPICS**

CLEMENTINE SERVER	14-2
CLEMENTINE SOLUTION PUBLISHER.....	14-3
CEMI	14-5
CLEMENTINE SCRIPTS	14-6
CLEO	14-7
TEXT MINING FOR CLEMENTINE.....	14-8
SPSS PREDICTIVEMARKETING AND PREDICTIVE WEB ANALYTICS	14-8
SUGGESTIONS FOR IMPROVING MODEL PERFORMANCE	14-9
DEMOS AND CATS	14-10

DATA MINING REFERENCES R-1**EXERCISES E-1**

Chapter 1

Introduction to Data Mining

Overview

- To introduce the concept of Data Mining
- To introduce the CRISP-DM process model as a general framework for carrying out Data Mining projects
- To sketch the plan of this course

Objectives

This section aims to provide an introduction to the process of Data Mining. You will gain an understanding of the terminology and key concepts used within Data Mining. Furthermore, we will see how Data Mining projects can be structured by using the CRISP-DM process model.

Introduction to Data Mining

With increasingly competitive markets and the vast capabilities of computers, many businesses find themselves faced with complex databases and a need to easily identify useful patterns and actionable relationships.

Data Mining is a general term, which describes a number of techniques used to identify pieces of information or decision-making knowledge in data. A common misconception is that it involves passing huge amounts of data through intelligent technologies that, alone, find patterns and give magical solutions to business problems. This is not true.

Data Mining is an interactive and iterative process. Business expertise must be used jointly with advanced technologies to identify underlying relationships and features in the data. A seemingly useless pattern in data discovered by Data Mining technology can often be transformed into a valuable piece of actionable information using business experience and expertise.

Many of the techniques used in Data Mining are referred to as “machine learning” or “modeling”. Historical data are used to generate models, which can be applied at a later date to areas such as prediction, forecasting, estimation and decision support.

Is Data Mining Appropriate?

Before considering which specific data mining technique is suitable, the business problem and the data need to be assessed for a potential data-mining project. There are several aspects, which should be considered:

1. Are data available?

Data need to be in an easily accessible format. It is often the case that relevant data files are held in several locations and/or in different formats and need to be pulled together before analysis. Data may even not be

in electronic format, possibly existing only on paper and needing data coding and data entry before data mining can be done. A data miner should also be aware of potential drawbacks, such as political or legal reasons why the data cannot be accessed.

2. Do data cover the relevant factors?

To make a data mining project worthwhile, it is important that the data, as far as possible, contain all relevant factors. Obviously, it is often the object of data mining to help identify relevant factors in the data. However, greater accuracy of predictions can be achieved if thought is given to this question.

3. Are the data too noisy?

“Noise” is a collective term given to errors in data, which can be present as missing data, or factors such as judgments, which can be variable due to their subjective nature. A level of noise in data is not unusual and the machine learning capabilities of Clementine have been shown to successfully handle data containing up to 50% noise. However, the more noise in data, the more difficult it will be to make accurate predictions.

4. Are there enough data?

The answer to this question depends on each individual problem. Very often it isn't the size of the data that causes difficulties in data mining, but more its representative nature and coverage of possible outcomes. As with the majority of data analysis techniques the more complex the patterns or relationships, the more records required to find them. If the data provide good coverage of possible outcomes, reasonable results can often be achieved using data sizes as small as a few thousand (or even a few hundred) records.

5. Is expertise on the data available?

Very often it is the expert who applies data mining techniques to her data and this problem need not be considered. However, if you are responsible for mining data from another organization or department, it is extremely desirable that experts who understand the data and the problem are available. They not only guide you in identifying relevant factors and help interpret the results, but also can often sort out the truly useful pieces of information from misleading artifacts often due to oddities in the data or relationships uninteresting from a business perspective.

A Strategy for Data Mining: the CRISP-DM Process Methodology

As with most business endeavors, data mining is much more effective if done in a planned, systematic way. Even with cutting edge data mining tools such as Clementine, the majority of the work in data mining requires the careful eye of a knowledgeable business analyst to keep the process on track. To guide your planning, answer the following questions:

- What substantive problem do you want to solve?
- What data sources are available, and what parts of the data are relevant to the current problem?
- What kind of preprocessing and data cleaning do you need to do before you start mining the data?
- What data mining technique(s) will you use?
- How will you evaluate the results of the data mining analysis?
- How will you get the most out of the information you obtained from data mining?

The typical data mining process can become complicated very quickly. There is a lot to keep track of—complex business problems, multiple data sources, varying data quality across data sources, an array of data mining techniques, different ways of measuring data mining success, and so on.

To stay on track, it helps to have an explicitly defined process model for data mining. The process model guides you through the critical issues outlined above and makes sure that the important points are addressed. It serves as a data mining road map so that you won't lose your way as you dig into the complexities of your data.

The data mining process model recommended for use with Clementine is the Cross-Industry Standard Process for Data Mining (CRISP-DM). As you can tell from the name, this model is designed as a general model that can be applied to a wide variety of industries and business problems. The first version of the CRISP-DM process model is now available. It is included with Clementine and can be downloaded from www.crisp-dm.org.

The general CRISP-DM process model includes six phases that address the main issues in data mining. The six phases fit together in a cyclical process.

These six phases cover the full data mining process, including how to incorporate data mining into your larger business practices. The six phases include:

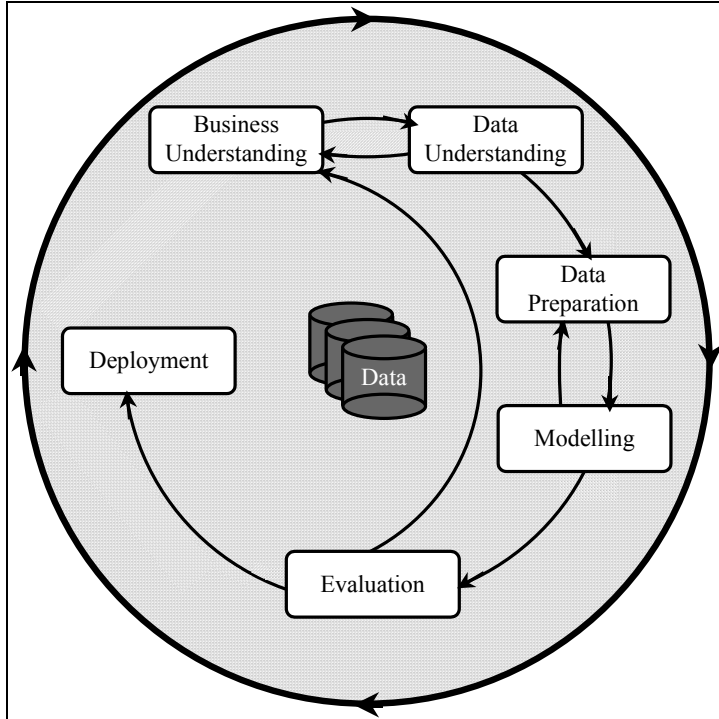
- **Business understanding.** This is perhaps the most important phase of data mining. Business understanding includes determining business objectives, assessing the situation, determining data mining goals, and producing a project plan.
- **Data understanding.** Data provides the "raw materials" of data mining. This phase addresses the need to understand what your data resources are and the characteristics of those resources. It includes collecting initial data, describing data, exploring data, and verifying data quality.
- **Data preparation.** After cataloging your data resources, you will need to prepare your data for mining. Preparations include selecting, cleaning, constructing, integrating, and formatting data.
- **Modeling.** This is, of course, the flashy part of data mining, where sophisticated analysis methods are used to extract information from the data. This phase involves selecting modeling techniques, generating test designs, and building and assessing models.
- **Evaluation.** Once you have chosen your models, you are ready to evaluate how the data mining results can help you to achieve your business objectives. Elements of this phase include evaluating results, reviewing the data mining process, and determining the next steps.
- **Deployment.** Now that you've invested all of this effort, it's time to reap the benefits. This phase focuses on integrating your new knowledge into your everyday business processes to solve your original business problem. This phase includes plan deployment, monitoring and maintenance, producing a final report, and reviewing the project.

There are some key points to this process model. First, while there is a general tendency for the process to flow through the steps in the order outlined above, there are also a number of places where the phases influence each other in a nonlinear way. For example, data preparation usually precedes modeling. However, decisions made and information gathered during the modeling phase can often lead you to rethink parts of the data preparation phase, which can then present new modeling issues, and so on. The two phases feed back on each other until both phases have been resolved adequately. Similarly, the evaluation phase can lead you to reevaluate your original business understanding, and you may decide that you've been trying to answer the wrong question. At this point, you can revise your business understanding and proceed through the rest of the process again with a better target in mind.

The second key point is the iterative nature of data mining. You will rarely, if ever, simply plan a data mining project, execute it and then pack up your data and go home. Using data mining to address your customers' demands is an ongoing endeavor. The knowledge gained from one cycle of data mining will almost invariably lead to new questions, new issues, and new opportunities to identify and meet your customers' needs. Those new questions, issues, and opportunities can usually be addressed by mining your data once again. This process of mining and identifying new opportunities should become part of the way that you think about your business and a cornerstone of your overall business strategy.

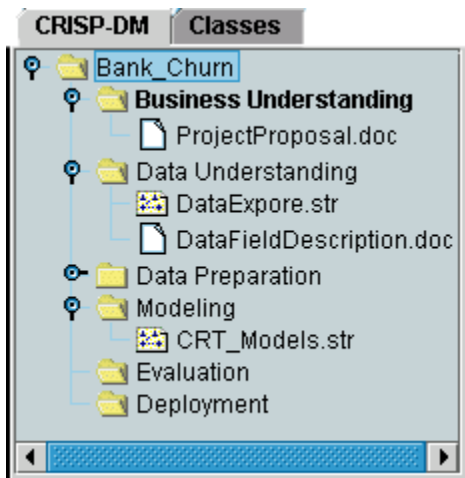
The figure below illustrates the main stages in a successful data mining process: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation and Deployment. The outer circle represents the fact that the whole process is iterative. The clockwise order of the tasks represents the common sequence.

Figure 1.1 Stages in CRISP-DM Process



To assist you in organizing your Clementine programs (streams) around the CRISP-DM framework, Clementine has a Project window. In the Project window, a project folder contains subfolders corresponding to the phases in CRISP-DM. This makes it easier to organize your Clementine programs and other documents that are associated with a data mining project. You can save a project file that contains links to Clementine streams and other documents.

Figure 1.2 Clementine Project Window



Plan of the Course

Clementine can be thought of as a “work bench”, combining multiple tools and technologies to support the process of Data Mining. This is to say that this course can be structured much in the same way along the lines of the CRISP-DM process model as Data Mining itself. As we will focus on the operational side of working with Clementine we won’t discuss the phase of Business Understanding here (see the Data Mining: Overview training course for a discussion), but we will cover the stages from Data Understanding to Deployment.

In the early chapters of this course we will develop a number of skills that can be used to perform Data Understanding, like reading data from diverse sources, browsing and visualizing the data using tables and graphs, how we can handle missing values, etc.

Next we will pay attention to Data Preparation. Conceptually, we can distinguish three types of manipulations.

- Manipulating records, like sorting and selecting records
- Manipulating fields, like deriving new fields
- Manipulating files, like adding records or merging fields

In this course we introduce record and field manipulation, while more detail is provided and file manipulation covered in the *Data Manipulation with Clementine* course.

Next, modeling techniques will be covered. Clementine provides a number of so-called “supervised learning” and “unsupervised learning” techniques:

- Supervised techniques model an output variable based on one or more input variables. These models can be used to predict or forecast future cases where the outcome is unknown. Neural Networks, Rule Induction (decision trees), Linear Regression, and Logistic Regression are some of these supervised techniques
- Unsupervised techniques are used in situations where there is no field to predict but relationships in the data are explored to discover its overall structure. Kohonen networks, Two Step, and K-means belong to this category.

Supervised and unsupervised techniques will be discussed in separate chapters. This is not to say that in the Data Mining practice these techniques are used in a standalone mode. On the contrary: supervised and unsupervised techniques are very often used together; for example, Kohonen cluster groups might be modeled separately or the cluster group field might be included as a predictor in a model. Association rules and sequence analysis may not fit neatly into one of the above categories, but these analysis techniques are discussed as well.

In the concluding chapter we will address, briefly, the Deployment phase of the CRISP-DM model and some additional Clementine features.

Summary

In this chapter we have introduced the concept of data mining. We identified a general approach towards data mining by introducing the CRISP-DM methodology.

Chapter 2

Introducing Clementine

Overview

- To provide an introduction to Clementine
- To familiarize yourself with the tools and palettes available in Clementine
- To introduce the idea of visual programming

Objectives

This session aims to provide an introduction to Clementine. You will gain an understanding of the capabilities of Clementine and see it in action.

Note Concerning Data for this Course

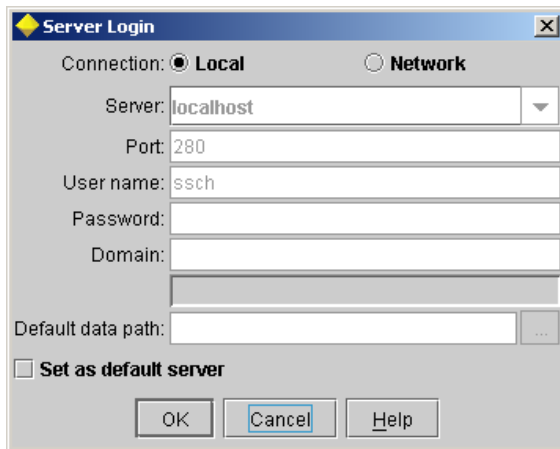
Data for this course are assumed to be stored in the directory *c:\Train\ClemIntro*. At SPSS training centers, the data is located in *c:\Train\ClemIntro* of the training PC. If you are working on your own computer, the *c:\Train\ClemIntro* directory can be created on your machine and the data copied from the accompanying floppy disk or CD-ROM. (Note: if you are running Clementine in distributed (Server) mode then the data should be copied to the server machine or the directory containing the data should be mapped from the server machine).

Clementine and Clementine Server

By default, Clementine will run in local mode on your desktop machine. If Clementine Server has been installed, then Clementine can be run in local mode or in distributed (client-server) mode. In this latter mode, Clementine streams are built on the client machine, but executed by Clementine Server. This architecture is introduced in Chapter 14.

Since the data files used in this training course are relatively small, we recommend you run in local mode. However, if you choose to run in distributed mode then make sure the training data are either placed on the machine running Clementine Server or that the drive containing the data can be mapped from the server. To determine in which mode Clementine is running on your machine, examine the connection status area of the Clementine status bar (left-most area of status bar) or click Tools..Server Login (from within Clementine) if the choice is active; if it is not active, then Clementine Server is not licensed. See within the Server Login dialog whether the Connection is set to Local or Network. This dialog is shown below.

Figure 2.1 Server Login Dialog Box in Clementine



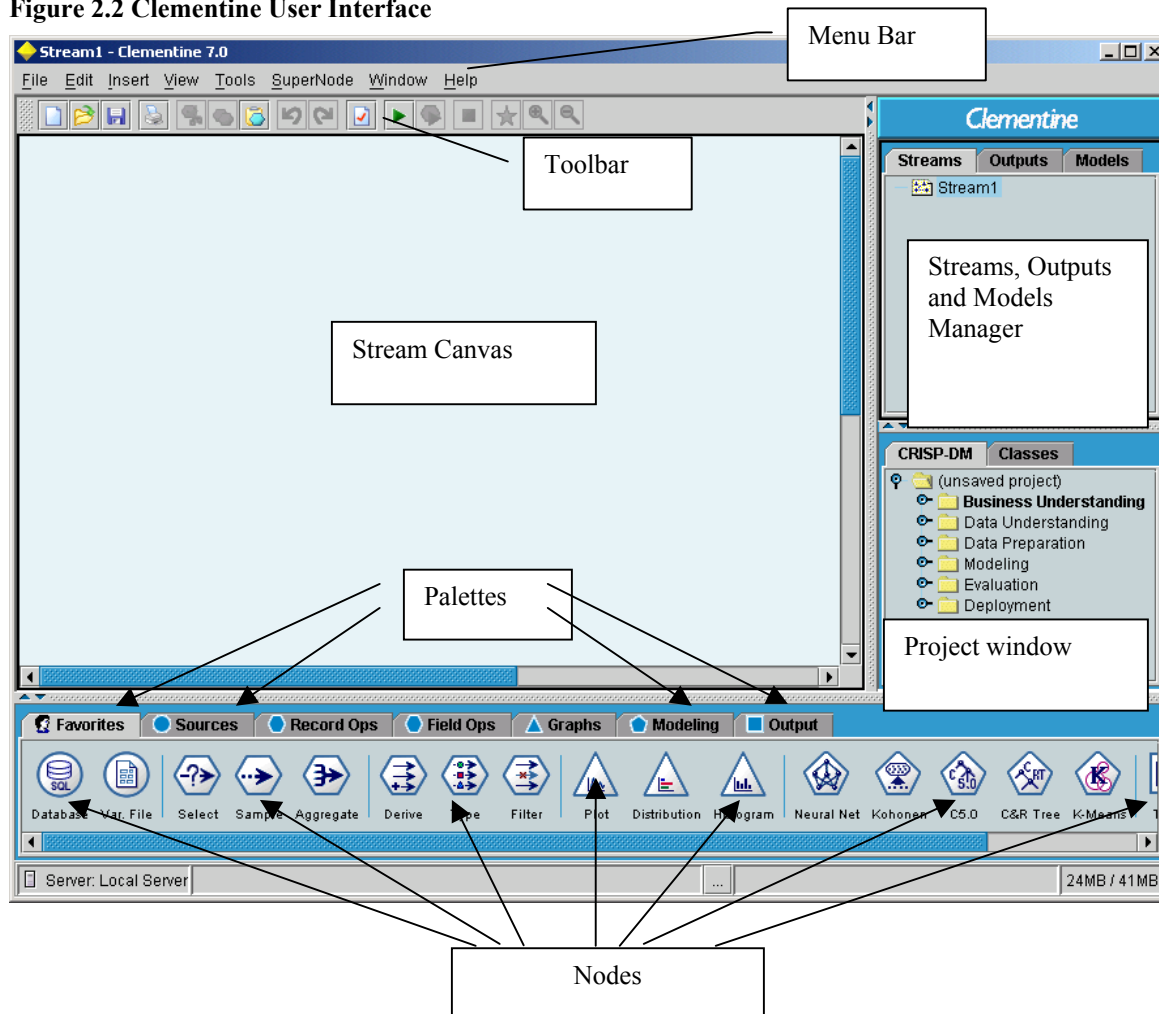
Starting Clementine

To run Clementine:

From the Start button, click **Programs..Clementine..Clementine**

At the start of a session, you see the Clementine User Interface.

Figure 2.2 Clementine User Interface



Clementine enables you to mine data by visual programming techniques using the Stream Canvas. This is the main work area in Clementine and can be thought of as a surface on which to place icons. These icons represent operations to be carried out on the data and are often referred to as nodes.

The nodes are contained in palettes, located across the bottom of the Clementine window. Each palette contains a related group of nodes that are available to add to the data stream. For example, the Sources palette contains nodes that you can use to read data into your model and the Graphs palette contains nodes that you can use to explore your data visually. Which icons are shown depends on the active, selected palette.

The Favorites palette is a customizable collection of nodes that the analyst uses most frequently. It contains a default collection of nodes, but these can be easily modified within the Palette Manager (reached by clicking Tools..Favorites).

Once nodes have been placed on the Stream Canvas, they can be linked together to form a stream. A stream represents a flow of data through a number of operations (nodes) to a destination that can be in the form of output (either text or chart) or a model.

At the upper right of the Clementine window (shown above), there are three types of manager tabs. Each tab (Streams, Outputs, and Models) is used to view and manage the corresponding type of object. You can use the Streams tab to open, rename, save, and delete streams created in a session. Clementine output, such as graphs and tables, are stored in the Outputs tab. You can save output objects directly from this manager.

The Models tab is the most important of the manager tabs as it contains the results of the machine learning and modeling conducted in Clementine. These models can be browsed directly from the Models tab or added to the current stream displayed in the canvas.

At the lower right of the Clementine window we have the Projects window. This window offers you a best-practice way to organize your data mining work. The CRISP-DM tab helps you to organize streams, output, and annotations according to the phases of the CRISP-DM process model (mentioned in Chapter 1). Even though some items do not typically involve work in Clementine, the CRISP-DM tab includes all six phases of the CRISP-DM process model so that you have a central location for storing and tracking all materials associated with the project. For example, the Business Understanding phase typically involves gathering requirements and meeting with colleagues to determine goals rather than working with data in Clementine. The CRISP-DM tab allows you to store your notes from such meetings in the Business Understanding folder of a project file for future reference and inclusion in reports.

The Classes tab in the Project window organizes your work in Clementine categorically by the type of objects created. Objects can be added to any of the following categories:

- Streams
- Nodes
- Models
- Tables, graphs, reports
- Other (non-Clementine files, such as slide shows or white papers relevant to your data mining work)

If we turn our attention to the Clementine menu bar there are eight menu options:

- File allows the user to create, open and save Clementine streams and projects. Streams can also be printed from this menu.
- Edit allows the user to perform editing operations: for example copy/paste objects; clear manager tabs; edit individual nodes.
- Insert allows the user to insert a particular node, as alternative to dragging a node from the palette.
- View allows the user to toggle between hiding and displaying items (for example: the toolbar or the Project window).
- Tools allows the user to manipulate the environment in which Clementine works and provides facilities for working with Scripts.
- Supernode allows the user to create, edit and save a condensed stream. Supernodes are discussed in the *Data Manipulation with Clementine* training course.
- Window allows the user to close related windows (for example, all open output windows).
- Help allows the user to access help on a variety of topics or view a tutorial.

Using the Mouse

When working with Clementine, the mouse plays an important role in performing most operations. Clementine takes advantage of the middle button in three-button mouse (or a Microsoft IntelliMouse), yet works with a standard two-button mouse.

There are alternatives to using the mouse, like using function keys or menus. Throughout this course, however, we will mainly use the mouse in our demonstrations.

Visual Programming

As mentioned earlier, data mining is performed by creating a stream of nodes through which the data pass. A stream, at its simplest, will include a source node, which reads the data into Clementine, and a destination, which can be an output node, such as a table, a graph, or a modeling operation.

When building streams within Clementine, mouse buttons are used in the following ways:

Left button	Used for icon or node selection, placement and positioning on the Stream Canvas.
Right button	Used to invoke Context (pop-up) menus that, among other options, allow editing, renaming, deletion and execution of the nodes.
Middle button [optional]	Used to connect two nodes and modify these connections. [When using a two-button mouse, you can right-click on a node, select Connect from the context menu, and then click on the second node to establish a connection.]

Note

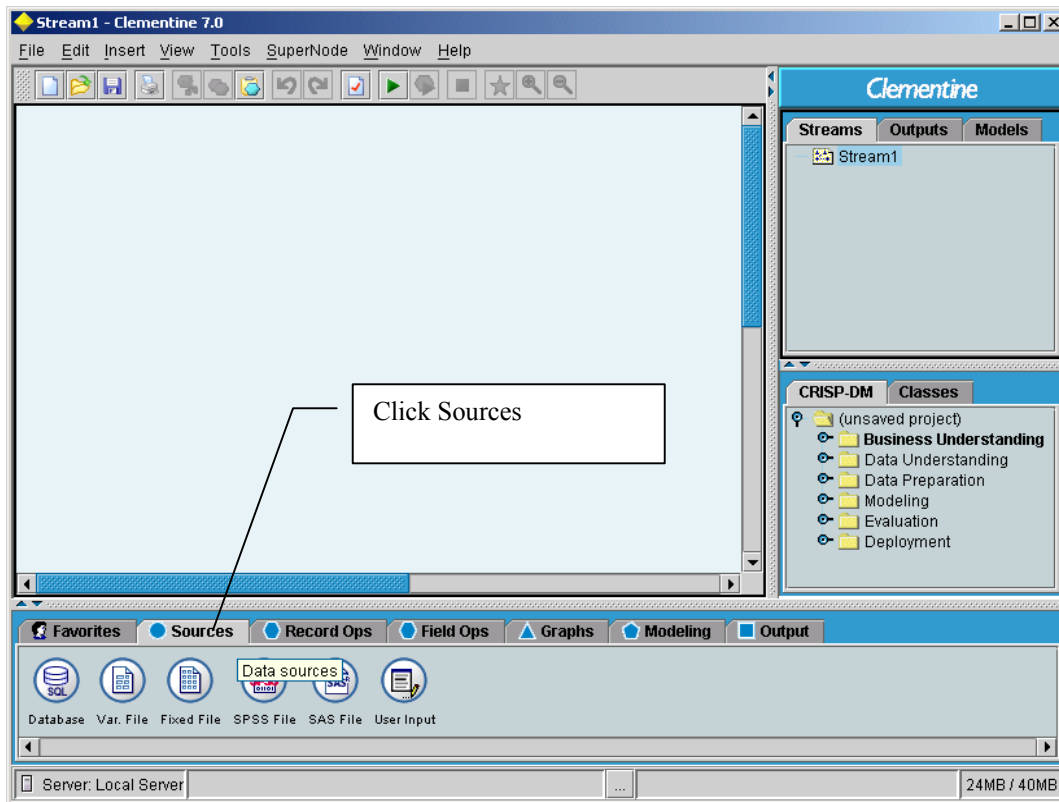
In this guide, the instruction to “click” means click with the primary (usually the left) mouse button; "right-click" means to click with the secondary (usually the right) mouse button; “middle-click” means to click with the middle mouse button.

Adding a Node

To begin a new stream, a node from the Sources palette needs to be placed on the Stream Canvas. In this example we will assume that data are being read from a previously saved SPSS data file (we will cover methods of reading data into Clementine in the next chapter).

Activate the Sources palette by clicking the **Sources** tab

Figure 2.3 Sources Palette



Select the **SPSS File** node from the Sources palette by clicking



This will cause the icon to be highlighted.

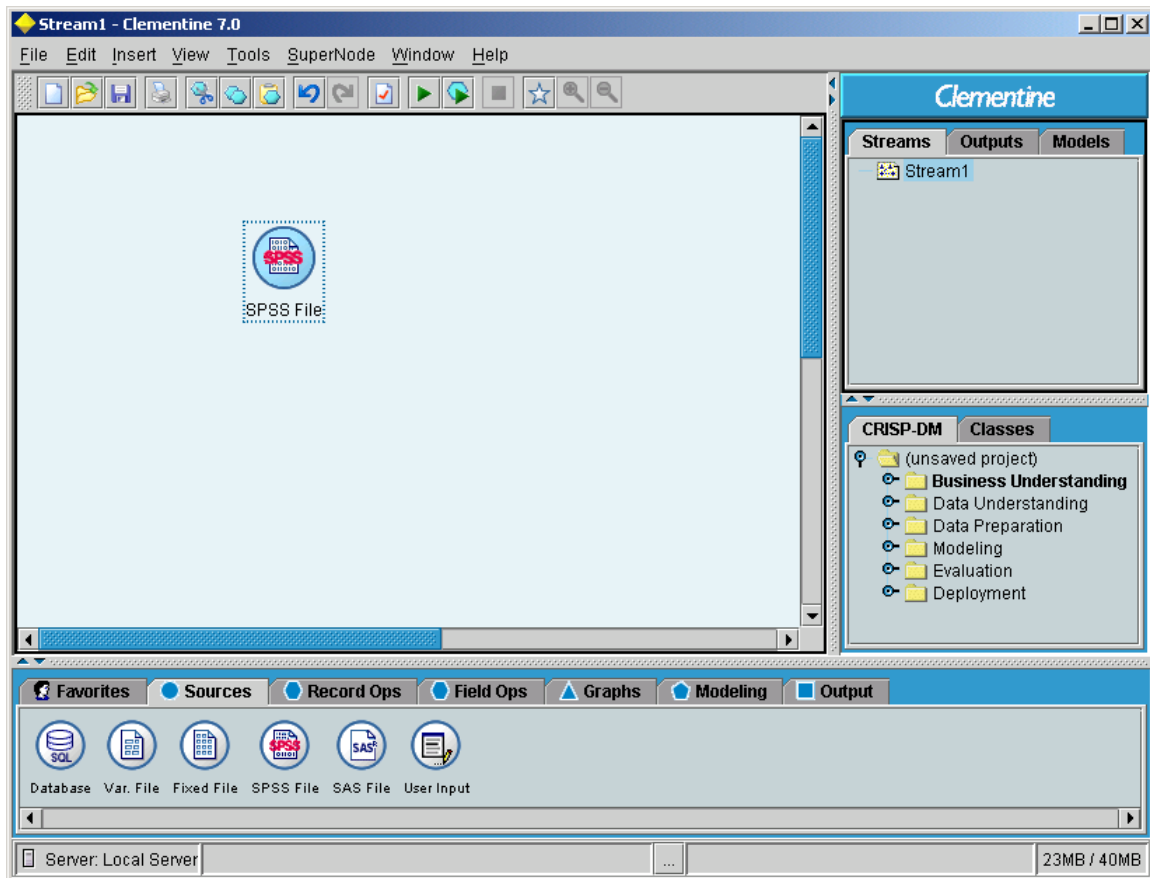
Move the **cursor** over the **Stream Canvas**

The cursor will change to a positioning icon when it reaches the Stream Canvas.

Click **anywhere** in the Stream Canvas

A copy of the icon should appear on the Stream Canvas. This node now represents the action of reading data into Clementine from a SPSS data file.

Figure 2.4 Placing a Node on the Stream Canvas



Moving a Node

If you wish to move the node within the Stream Canvas, select it (using the left mouse button), and while holding this button down, drag the node to its new position.

Editing a Node

In order to view the editing facilities of a node, right click on the icon to reveal a Context (pop-up) menu.


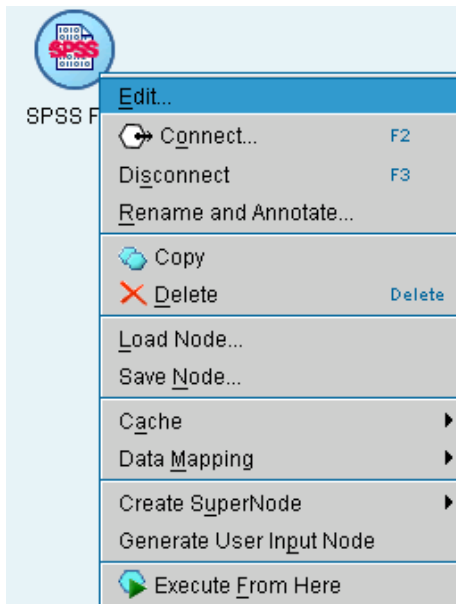
Right click on the **SPSS File** node  in the Stream Canvas

Figure 2.5 Context (Pop-up) Menu When a Source Node Is Right Clicked

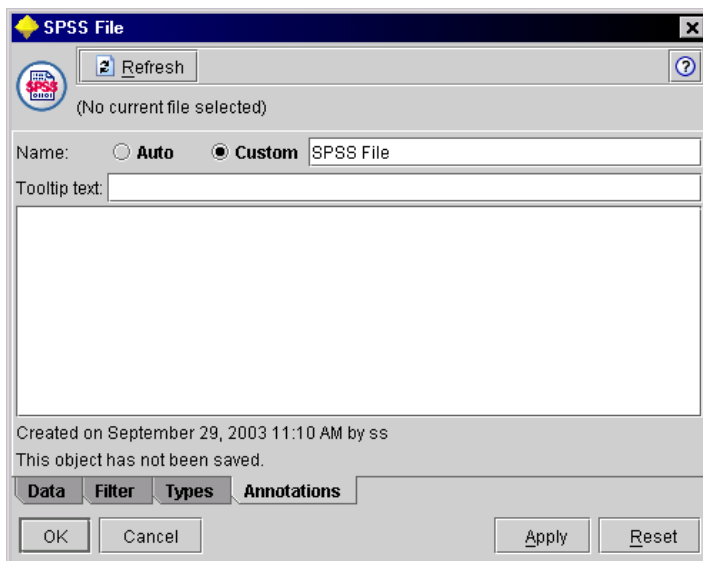
The Edit option opens a dialog box specific to each node type.

Double clicking on the node also accesses the editing facilities of a node. We will skip the edit options for the moment as we cover the edit options for many of the nodes later in the course.

Renaming and Annotating a Node

In order to label a node in the Stream Canvas with a more descriptive name:

- Right click on the **SPSS File** node
- Click **Rename and Annotate** on the context menu

Figure 2.6 Rename and Annotate Dialog

We can specify a name and even tooltip text for the node. In the text area, additional information can be attached to the node in order to aid interpretation or to act as a reminder to what it represents. For the moment, however, we will cancel and look at the other options.

Click **Cancel**

Copy and Paste a Node

Copy and paste helps you to, for instance, duplicate a node (an action later used in this course). To duplicate a node:

Right click the node and select **Copy** from the context menu
Right click in an empty area of the Stream Canvas
Select **Paste** from the context menu

A duplicate of the copied node will appear in the Stream Canvas. If needed, the node can be moved, renamed and annotated as described previously.

Deleting a Node

To delete a node:

Right click on the node
Select **Delete** from the context menu (Alternatively, select the node and then press the Delete key)

Building Streams with Clementine

Once two or more nodes have been placed on the Stream Canvas, they need to be connected to produce a stream. This can be thought of as representing a flow of data through the nodes.

To demonstrate this we will place a Table node in the Stream Canvas, next to the SPSS File node (if you just deleted the SPSS File node, please replace it on the Stream Canvas). The Table node presents the data in a table format, similar to a spreadsheet view.

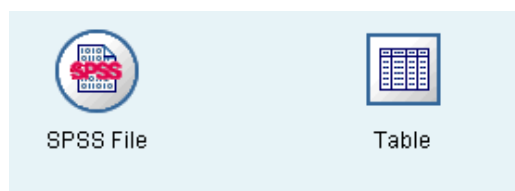
Click the **Output** tab to activate the Output palette



Click on the **Table** node Table in the Output palette

Place this node to the right of the **SPSS File** node by clicking in the Stream Canvas


Figure 2.7 Unconnected Nodes



Connecting Nodes

To connect the two nodes:

Right-click on the SPSS File node, and then select **Connect** from the context menu (note

the cursor changes to include a connection icon )

Click the **Table** node

Alternatively, with a three-button mouse:

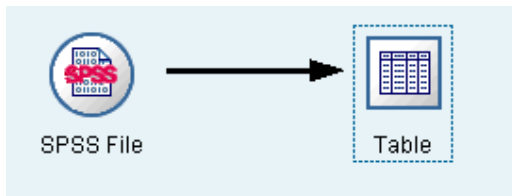
Click with the middle mouse button on the **SPSS File** node

While holding the middle button down, drag the cursor to the **Table** node

Release the middle mouse button

A connecting arrow appears between the nodes. The head of the arrow indicates the data flow direction.

Figure 2.8 Stream Representing the Flow of Data



Disconnecting Nodes

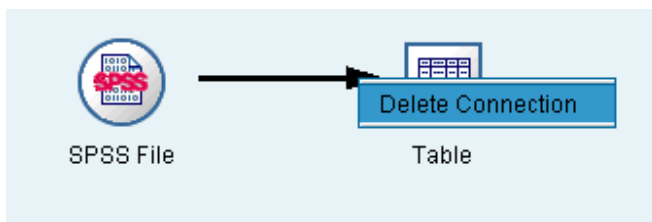
Nodes can be disconnected in several ways:

- By right clicking on one of the nodes and selecting the Disconnect option from the context menu
- By right clicking on the actual connection and selecting the Delete Connection option
- By double clicking with the middle mouse button on one of the connected nodes (for intermediate nodes this will make existing arrows “bypass” the node)

We will demonstrate one of these alternatives.

Right click on the connecting arrow

Figure 2.9 Disconnecting Nodes



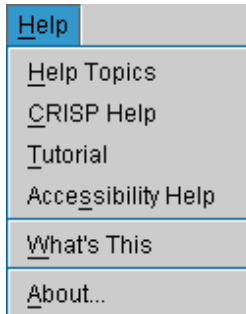
Click **Delete Connection**

Getting Help

Help can be accessed via the Help menu:

Click **Help**

Figure 2.10 Help Menu

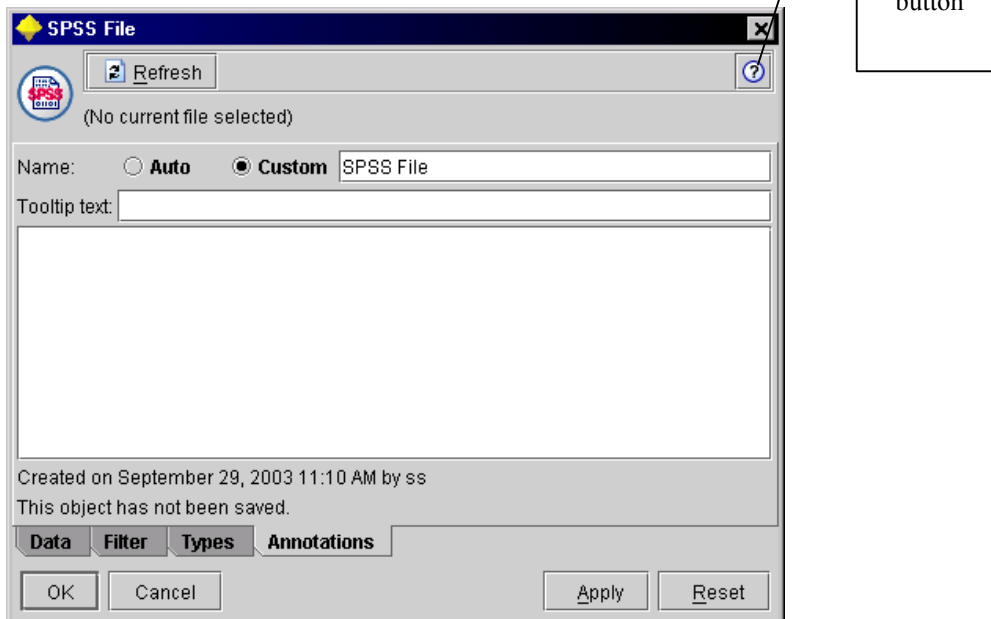


The Help menu contains several options. The Help Topics choice takes you to the Help system. CRISP Help gives you an introduction to the CRISP-DM methodology. Tutorial leads to a tutorial on the use of Clementine (valuable when first working with Clementine). Accessibility Help informs you about keyboard alternatives to using the mouse. “What’s This” changes the cursor into a question mark and provides information about any Clementine item you click.

Besides the help provided by the Help menu, you always have context sensitive help available in whatever dialog box you are working. As an example we look at the help when we rename or annotate the SPSS File node.

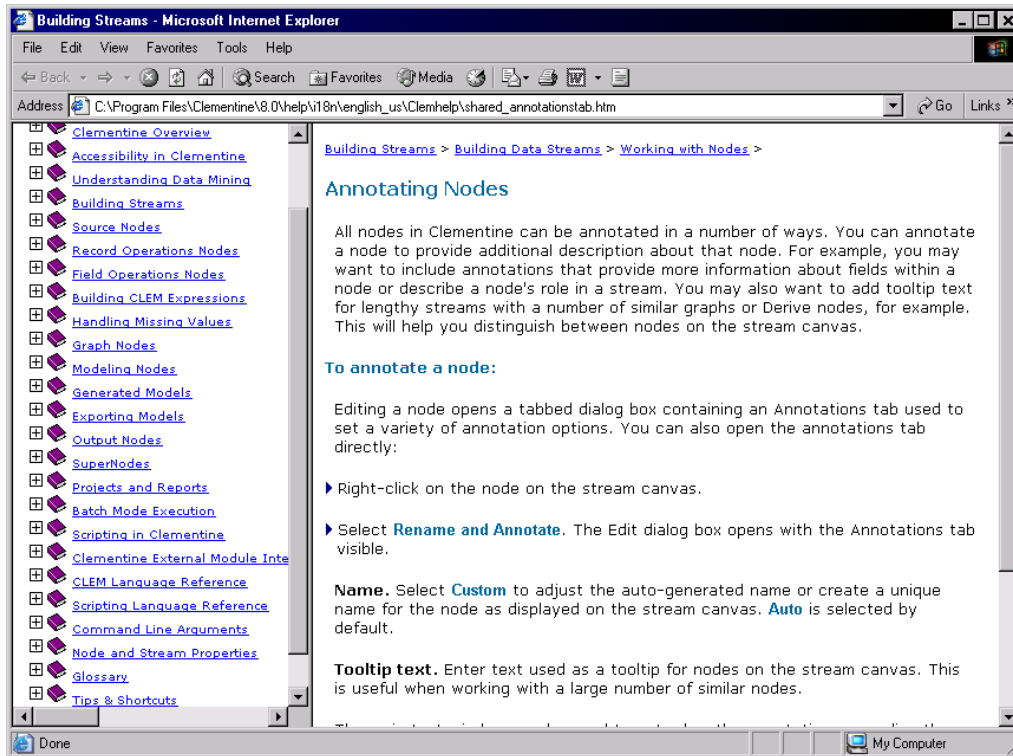
Right-click the **SPSS File** node, then click **Rename and Annotate**

Figure 2.11 Context Sensitive Help



Click the **Help** button 

Figure 2.12 Context Sensitive Help on Annotating Nodes



Information about Clementine's features and operations can be obtained through the Help system.

Click the close button  to close the Help and to return to the Stream Canvas

Summary

In this chapter you have been given a brief tour of the Clementine User Interface.

You should now be able to:

- Place an icon on the Stream Canvas to create a node
- Move, duplicate and delete these nodes
- Rename and annotate nodes
- Connect two or more nodes to create a stream
- Obtain help within Clementine

Chapter 3

Reading Data Files

Overview

- Data formats read by Clementine
- Reading free-field text data files
- Reading SPSS data files
- Reading databases using ODBC
- Viewing the data
- Data types in Clementine
- Field direction
- Saving Clementine Streams
- Appendix: Reading fixed-field text data files

Objectives

This session aims to introduce some of the ways of reading data into Clementine. By the end of the chapter you should be able to read in data from both text and SPSS files and view the data in a table. You will also be aware of the other data formats Clementine can read and the different data field types within Clementine.

Data

In this chapter a data set in several different formats will be used to demonstrate the various ways of reading data into Clementine. The data file contains information concerning the credit rating and financial position of 514 individuals. The file also contains basic demographic information, such as marital status and gender.

In order to demonstrate ODBC we will use an Access database, *custandhol.mdb*. The database has three tables, *custtravel1*, *custtravel2* and *holtravel*, containing information on the customers of a travel company.

Reading Data Files into Clementine

Clementine reads a variety of different file types, including data stored in spreadsheets and databases, using the nodes within the Sources palette.

Data can be read in from text files, in either free-field or fixed-field format, using the Var. File and Fixed File source nodes.

SPSS and SAS data files can be directly read into Clementine using the SPSS File and SAS File nodes.

If you have data in an ODBC (Open Database Connectivity) source, you can use the Database source node to import data from server databases, such as OracleTM or SQL ServerTM and from variety of other packages including ExcelTM, AccessTM, dBaseTM, and FoxProTM.

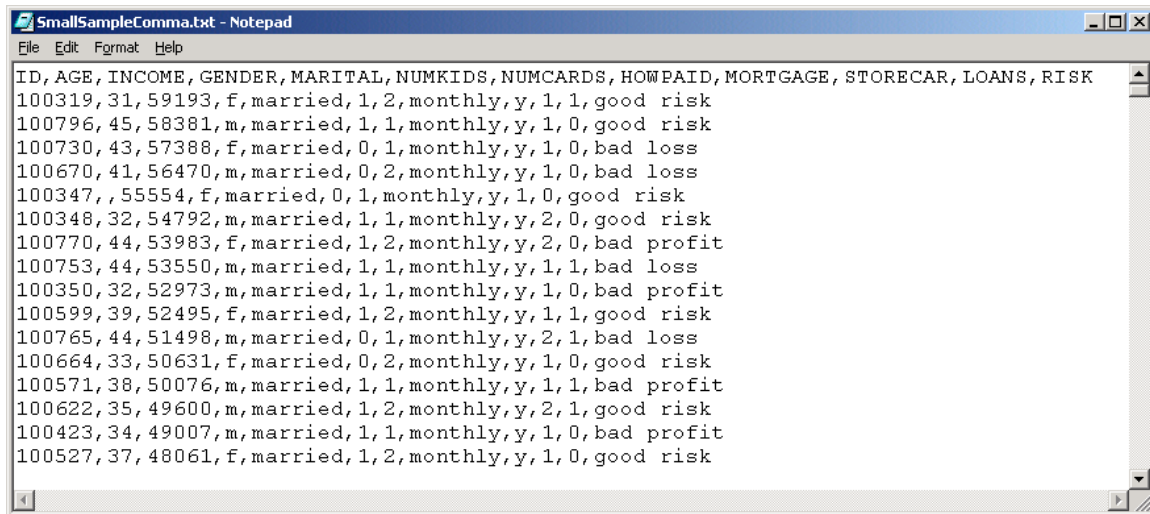
Clementine can also simulate data with the User Input node in the Sources palette. This node is useful for generating data for demonstrations or testing.

Further information on the types of data imports available in Clementine can be found in the *Clementine User's Guide*.

Reading Data from Free-Field Text Files

The Var. File node reads data from a free-field (delimited) text file. We demonstrate this by reading a comma-separated data file with field names in the first record. The figure below shows the beginning of the file (using Notepad).

Figure 3.1 Free-field Text File



The file contains demographic information like age, income, gender, marital status, number of children and also information about the credit risk group of the person (good risk, bad profit and bad loss). Note that we have a mix of numeric (e.g. id, age, income) and discrete values (e.g. gender, marital, risk).

We will read this file into Clementine, using the Var File source node. Before we do this, however, we advise you to empty the stream canvas and to start from scratch.

If the Stream Canvas isn't empty, clear the Stream Canvas by choosing **Edit..Clear Stream**

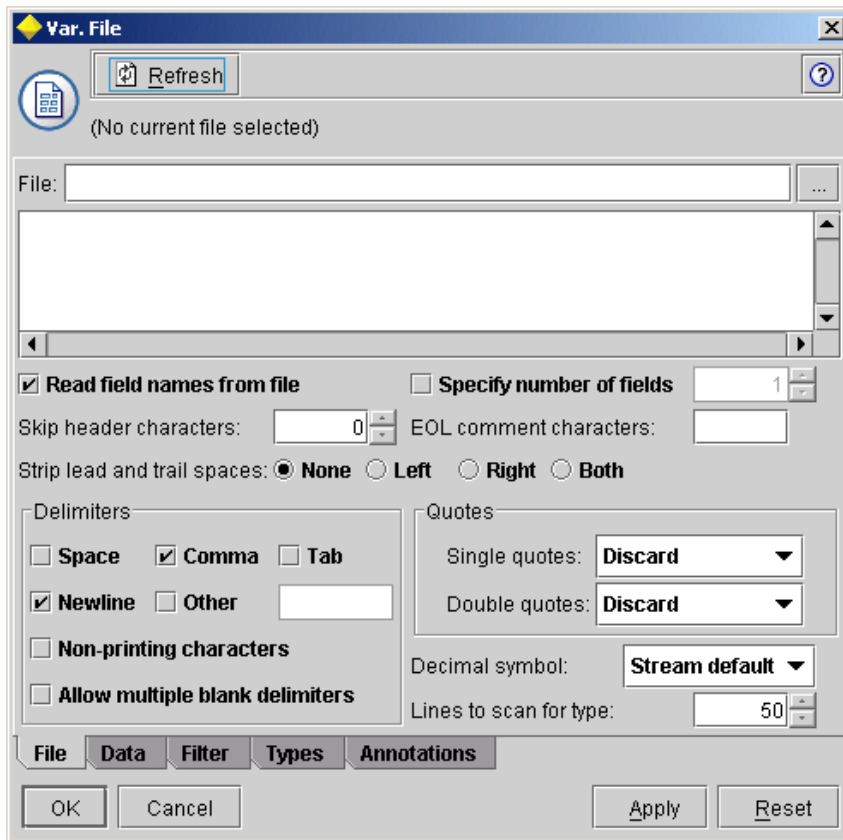
Click the **Var. File** node in the Sources palette


Position the cursor on the left side of the Stream Canvas and click once (not shown)

A copy of the icon should appear in the Stream Canvas. This source node represents the process of reading a data file into Clementine. To link this node to a specific file, it needs to be edited.

Right-click on the **Var. File** node, then click **Edit** (alternatively, double-click the **Var. File** node)

Figure 3.2 Variable File Node Dialog Box



First thing to do is to specify the file name. The file list button  is used to browse through directories and to identify the data file.


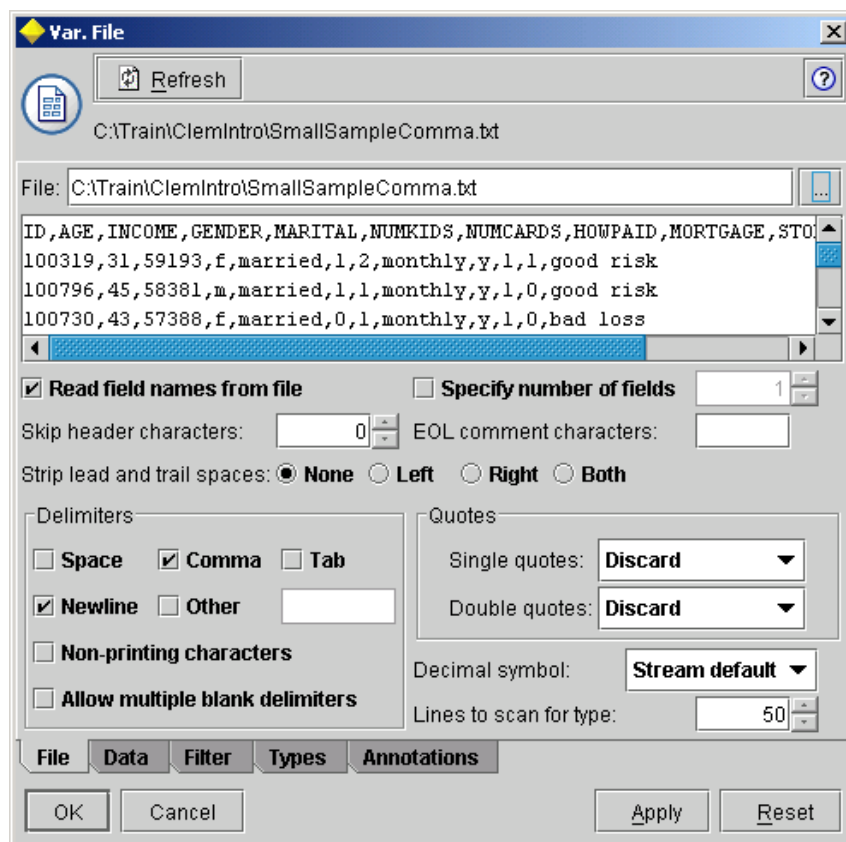
Click the file list button , and then move to the **c:\Train\ClemIntro** directory
Click **SmallSampleComma.txt**, and then click **Open**

Figure 3.3 Variable File Node Dialog



The Var. File dialog gives a preview of the first lines of data. Note, that the first line of data contains field names. These names can be read directly into Clementine by checking the *Read field names from file* check box. By default, this option is already checked.

The *Skip header characters* text box allows you to specify how many characters are to be read and ignored before the first record begins. This is not relevant here.

The *EOL comment characters* text box allows the declaration of one or more characters that denote the start of a comment or annotation. When Clementine comes across these characters in the file, it will ignore what follows until a new line is started. This is not relevant in our situation.

The *Specify number of fields* option allows you to specify how many fields are in each record in the data file. Alternatively, if all fields for a data record are contained on a single line and the number of fields is left unspecified, then Clementine automatically calculates the number of fields.

Leading and trailing blanks can be removed from symbolic fields in several ways using their respective options.

The characters used to separate the individual fields within the file are specified under *Delimiters*. White space (blanks) and tabs can also be declared as delimiters using their check box options. Single and double quotes are dealt with using the drop-down menus and can either be discarded (*Discard*), matched in pairs and then discarded (*Pair & Discard*) or included as text within the field (*Include as text*).

The *Decimal symbol* is the same as the Windows decimal symbol and could be set to either a comma or a period by using the drop-down menu.

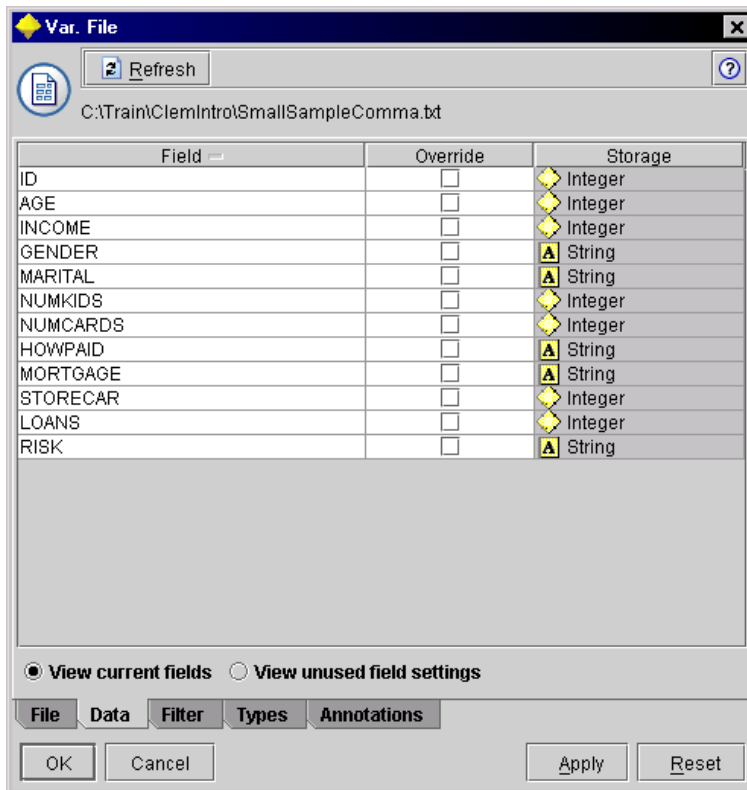
By default Clementine will scan 50 rows of data to determine the field type. Field type is one of the main concepts to be aware of when working with Clementine and we will discuss this in detail later in this chapter.

Let's now see how Clementine reads the field names from the specified data file.

Make sure that **Read Field Names from File** is checked
Click the **Data** tab

Clementine scans the first 50 lines of data and reports the field names found. These field names are displayed within the dialog box shown below.

Figure 3.4 Data Tab Displaying Field Names



In Figure 3.3, the field names looked reasonable. If there were no field names in the file and the *Get field names from file* option were not checked, Clementine would assign the names, field1, field2, etc.

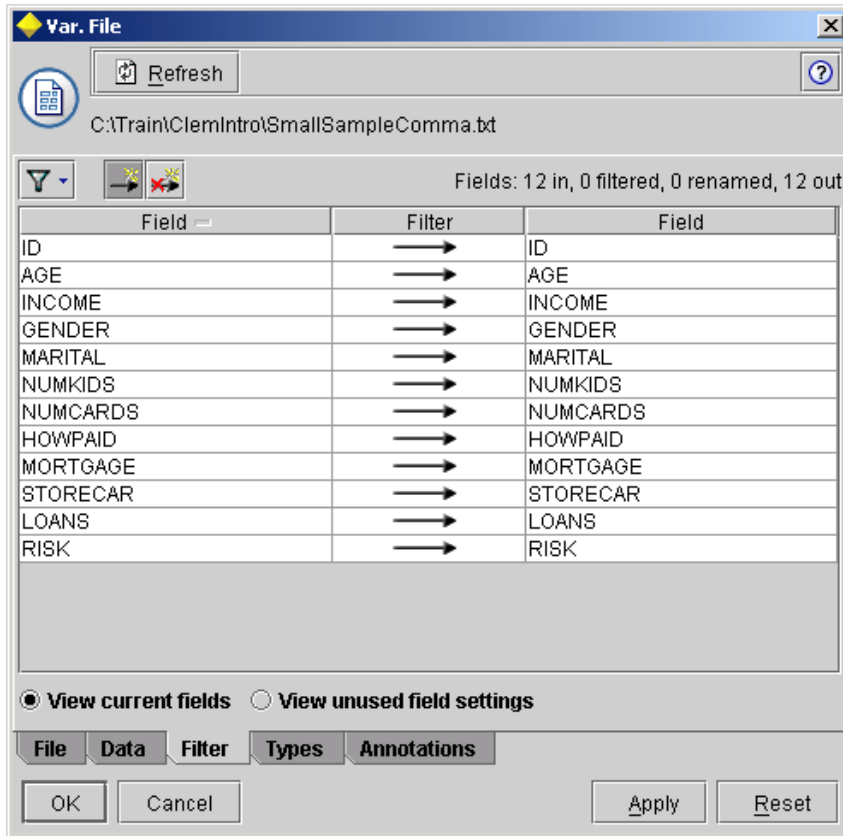
In the figure above we see *Override* and *Storage* columns. *Storage* describes the way data for a field is stored by Clementine and this has implications for how the field can be used within Clementine. For example, fields with integer or real data storage would be treated as numeric by Clementine modeling nodes (their type would be numeric, unless changed). If you had numeric data values for a field that should be treated as symbolic (categorical), for example numeric codes for marital status, one way to accomplish this would be to override the default data storage for such a field and set it to string. Storage options include string, integer (integer numeric), real (decimal numeric), time, date, timestamp, and unknown.

We can set the data storage for a field by checking its *Override* box, clicking in its *Storage* cell, and then selecting the storage from the drop-down list. Generally, the storage and type determined automatically by Clementine will be appropriate for your analyses, but if needed, can be changed.

We might want to change one or more field names or even decide that some fields should not be read into Clementine. The Filter tab allows us to control this.

Click the **Filter** tab

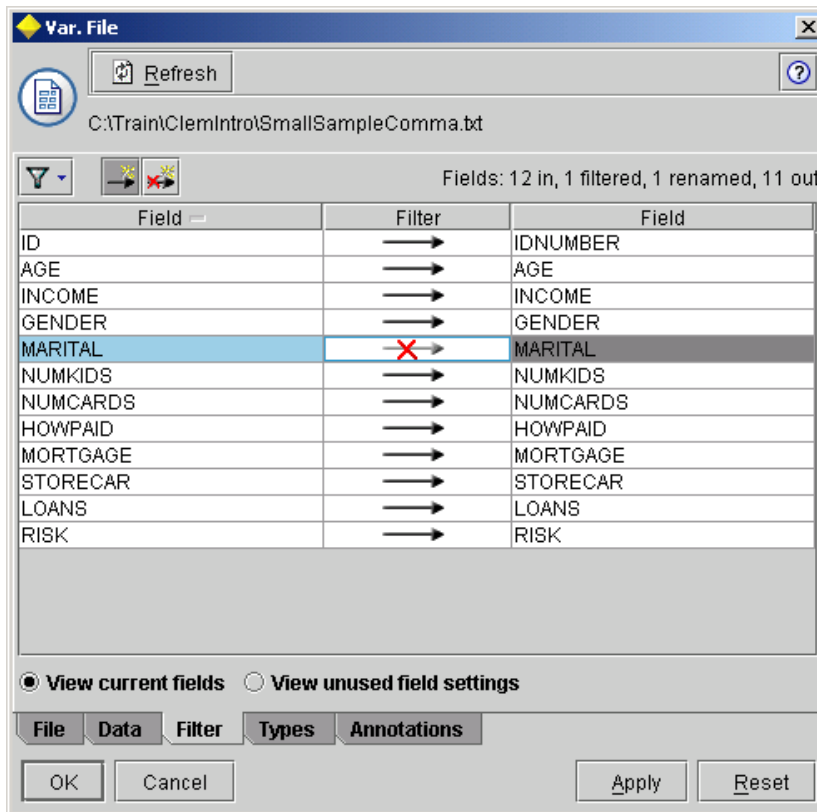
Figure 3.5 Filter Tab in Var. File Node






The left column contains the field names as read from file. We can specify new names in the right column. The middle column shows an arrow that can be interpreted as “becomes”. As an example, suppose we would like to rename ID to IDNUMBER and would like to exclude MARITAL.

- Double-click on **ID** in the **right Field** column
- Change **ID** to **IDNUMBER**
- Click once on the arrow in the **MARITAL** row


Figure 3.6 Changing Field Names and Dropping Fields



ID is renamed IDNUMBER. The crossed arrow in the MARITAL row indicates that data for MARITAL won't be read into Clementine.

Within the Filter tab, you can sort the fields (just click on column header Field), exclude all fields at once by clicking the  button or include all fields at once by clicking the  button. Furthermore, the button  gives access to numerous filter options such as: including/excluding all fields, toggling between fields, removing or renaming duplicates automatically and truncating fieldnames.

As an example we will undo the previous changes.

Click the  button and select **Include All Fields**

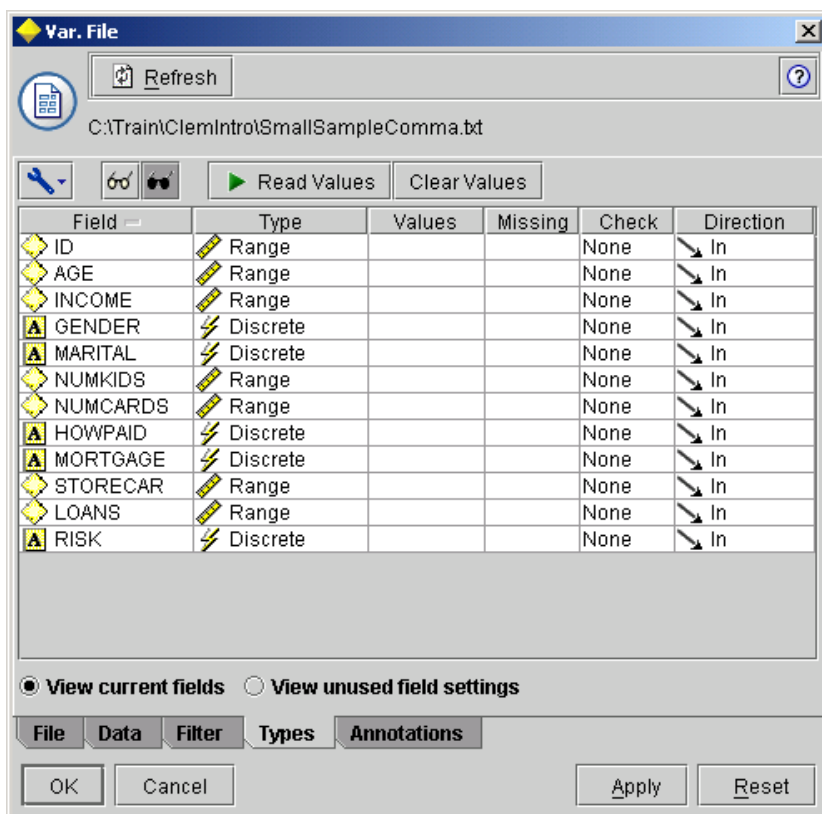
Click the  button and select **Use Input Field Names**

The window should be the same as it was prior to changing ID into IDNUMBER and excluding MARITAL (not shown).

The last tab to mention in this window is the Types tab. This tab displays properties of the fields.

Click the **Types** tab

Figure 3.7 Types Tab in Var. File Dialog



Field type determines, in general, how the field will be used by Clementine in data manipulation and modeling. Initially, fields with numeric values are typed as Range, and fields with symbolic values are typed as Discrete. For the moment we will postpone a detailed discussion about field types. Later in the chapter we will elaborate upon field types and see how they can be set manually or assigned automatically by Clementine as the data values are read.

The display above is based on 50 lines of data and thus presents partially instantiated types; types are fully instantiated when data pass through the node while the field Values are set to Read or Read+. If any field types are incorrect, Clementine allows you to set types before a full data pass is made. This can eliminate an unnecessary pass through the data, which is valuable when dealing with large files.

Click **OK**

By clicking OK you will return to the Stream Canvas, where the Var. File source node will have been labeled with the file name (not shown).

First Check on the Data

Once a data source node has been positioned in the Stream Canvas and linked to the data file, it is often advisable to check whether Clementine is accessing the data correctly. The nodes within the Output palette display data, provide summary reports and analyses, and export data from Clementine.

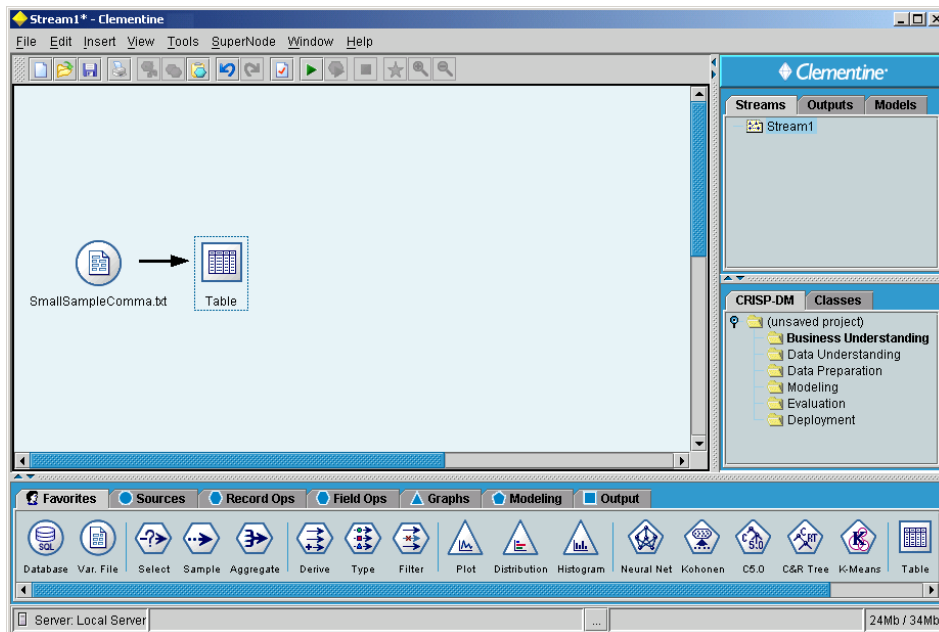
The Table node displays the data in tabular form, with one row per record and field names heading the columns. To view the data file, this node must be positioned in the data stream, downstream of the data source node that accesses the data file. Since it is an often-used node, the Table node is also present on the Favorites palette.

Click the **Table** node on the Favorites palette
Click to the right of the Var. File node named **SmallSampleComma.txt**

To connect the nodes:

Right-click the **Var. File** node (SmallSampleComma.txt), select **Connect** from the context pop-up menu, and then click the **Table** node in the Stream Canvas
(Alternatively, click the middle mouse button on the Var. File node [SmallSampleComma.txt] and drag the cursor to the **Table** node in the Stream Canvas)

Figure 3.8 Var. File Node Connected to Table Node

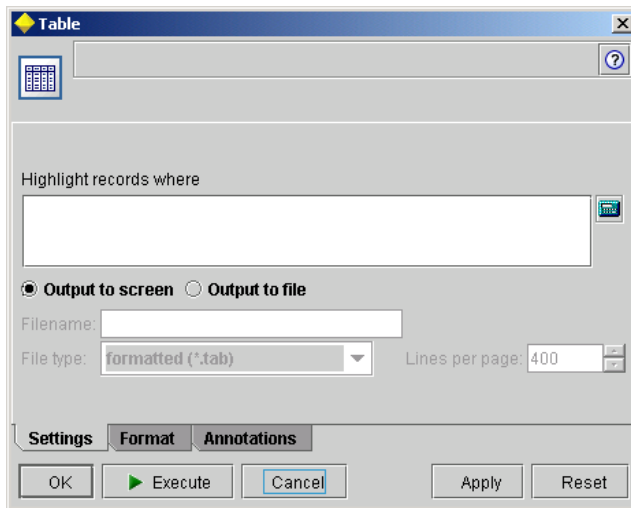


An arrow appears connecting the source node to the Table node. The direction of the connecting arrow indicates the direction of data flow, passing from the source node into the Table output node.

The output style can be chosen, by editing the Table node:

Double-click on the **Table** node

Figure 3.9 Table Node Dialog



The *Highlight records where* option allows you to enter an expression that identifies a set of records to be highlighted within the table. Two output styles are available for the Table (and other display nodes) in the Output palette:

- Output to Screen: Displays the output on the screen
- Output to File: Sends output to file

If you choose to send output to file several formats are available:

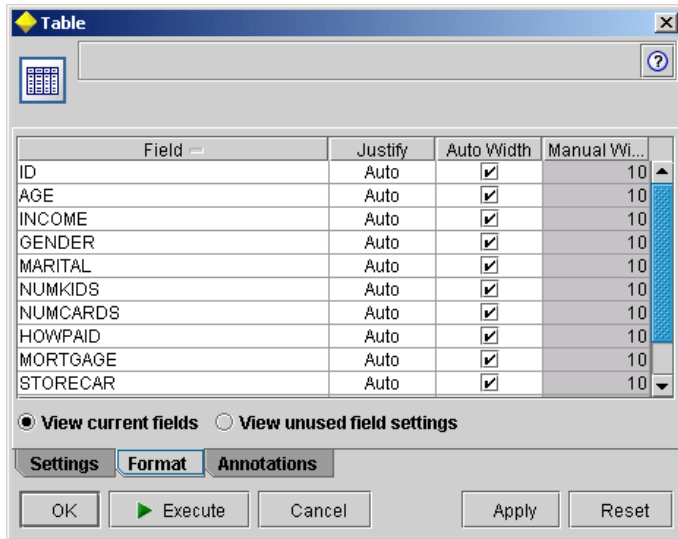
- Formatted (*.tab): a tab delimited text file
- Data (comma-delimited) (*.dat): a comma-delimited text file
- Html document (*.htm): an HTML document
- Transposed (*.dat): a comma-delimited text file, similar to the Data option, except that the rows represent fields and the columns represent records

We will stay with *Output to screen*, in order to see the table instantly.

Using the Format tab, the format of each individual field can be changed by selecting the field and choosing either Auto or a Manual width (which you can specify). In addition, a field can be left-, centered-, or right-justified within the column.

Click the **Format** tab

Figure 3.10 Format of Fields in Table Output



Once the dialog box has been edited, you can run the table by clicking the Execute button. Alternatively, you can return to the Stream Canvas by clicking OK and then execute the table by right-clicking on the Table node and selecting Execute from the context menu. A third alternative, which we will use in this instance is to execute the stream by using the *Execute the current stream* button in the Toolbar.

Click **OK**

Click the **Execute** (the current stream) button in the Toolbar (button shown below)

Figure 3.11 Execute Button in the Toolbar



Execute the current stream

After having executed the Table using any of the methods mentioned, the table window opens.

Figure 3.12 Table Window Showing Data from Text File


	ID	AGE	INCOME	GENDER	MARITAL	NUMKIDS	NUMCARDS	HOWPAID	MORTGAGE
1	100319	31	59193	f	married	1	2	monthly	y
2	100796	45	58381	m	married	1	1	monthly	y
3	100730	43	57388	f	married	0	1	monthly	y
4	100670	41	56470	m	married	0	2	monthly	y
5	100347	\$null\$	55554	f	married	0	1	monthly	y
6	100348	32	54792	m	married	1	1	monthly	y
7	100770	44	53983	f	married	1	2	monthly	y
8	100753	44	53550	m	married	1	1	monthly	y
9	100350	32	52973	m	married	1	1	monthly	y
10	100599	39	52495	f	married	1	2	monthly	y
11	100765	44	51498	m	married	0	1	monthly	y
12	100664	33	50631	f	married	0	2	monthly	y
13	100571	38	50076	m	married	1	1	monthly	y
14	100622	35	49600	m	married	1	2	monthly	y
15	100423	34	49007	m	married	1	1	monthly	y
16	100527	37	48061	f	married	1	2	monthly	y
17	100595	39	47161	m	married	1	2	monthly	y
18	100565	38	46823	m	married	0	1	monthly	y
19	100470	36	45949	f	married	1	1	monthly	y

The title bar of the Table window displays the number of fields and records read into the table. If needed, scroll bars are available on the right side and bottom of the window allowing you to view hidden records and fields.

Note the \$null\$ value for the 5th record in the AGE field. Looking back at the file SmallSampleComma.txt (Figure 3.1), we see that this person had no value for AGE, so Clementine assigned AGE a missing value, represented by the value \$null\$. (We will discuss missing values in detail later.)

The File menu in the Table window allows you to save, print or export the table. Using the Edit menu you can copy values or fields. The Generate menu allows you to automatically generate Select (data selection) and Derive (field calculation) nodes.

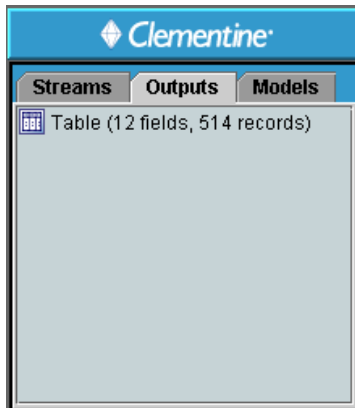
Now we have checked that the data file has been correctly read into Clementine, we will close the window and return to the Stream Canvas.

Click **Close**  to close the Table window

Although we have closed the table, the table is still available in the Outputs manager, so we don't have to execute the table again to see the data. To activate the Outputs manager:

Click the **Outputs** tab (shown below)

Figure 3.13 Outputs Tab in Manager



The table is still available from the manager. In fact, each output produced (table or chart) will automatically be added as separate item in the Outputs tab and is available for later use.

For the moment we will clear this stream and take a look at reading other data sources. To clear the stream:

Click **Edit..Clear Stream**

To clear the output:

Right-click in an empty area in the **Outputs** tab
Click **Delete** (or **Delete All**) from the context menu
(Alternatively, click Edit..Clear Outputs)

Reading SPSS Data Files

Clementine can import and export SPSS data files. Importing is done using the SPSS File node in the Sources palette, while exporting involves the SPSS Export node in the Output palette. As an example we will open the SPSS data file *SmallSample.sav*. The data are shown below in SPSS.

Figure 3.14 Data File in SPSS: Data Values Displayed

	id	age	income	gender	marital	numkids	numcards	howpaid	mortgag	str
1	100319	31	59193	1	2	1	2	monthly	y	
2	100796	45	58381	2	2	1	1	monthly	y	
3	100730	43	57388	1	2	0	1	monthly	y	
4	100670	41	56470	2	2	0	2	monthly	y	
5	100347	.	55554	1	2	0	1	monthly	y	
6	100348	32	54792	2	2	1	1	monthly	y	
7	100770	44	53983	1	2	1	2	monthly	y	
8	100753	44	53550	2	2	1	1	monthly	y	
9	100350	32	52973	2	2	1	1	monthly	y	
10	100599	39	52495	1	2	1	2	monthly	y	
11	100765	44	51498	2	2	0	1	monthly	y	
12	100664	33	50631	1	2	0	2	monthly	y	

Typically, data in SPSS are encoded. For instance, *marital* does not have the values *divsepwid*, *married*, *single*, but the values 1, 2, 3. In SPSS we typically attach value labels to these codes to explain what the codes represent. Also when using SPSS, users often attach a label to the variable (field) name, so it's clear what the variable (field) represents. A label attached to the field is called a variable label in SPSS.

You can view either the data values or value labels in the Data View tab of the SPSS Data Editor window.

Figure 3.15 Data File in SPSS: Value Labels Displayed

	id	age	income	gender	marital	numkids	numcards	howpaid	mortgag	str
1	100319	31	59193	Female	Marital Status	1	2	monthly	y	
2	100796	45	58381	Male	married	1	1	monthly	y	
3	100730	43	57388	Female	married	0	1	monthly	y	
4	100670	41	56470	Male	married	0	2	monthly	y	
5	100347	.	55554	Female	married	0	1	monthly	y	
6	100348	32	54792	Male	married	1	1	monthly	y	
7	100770	44	53983	Female	married	1	2	monthly	y	
8	100753	44	53550	Male	married	1	1	monthly	y	
9	100350	32	52973	Male	married	1	1	monthly	y	
10	100599	39	52495	Female	married	1	2	monthly	y	
11	100765	44	51498	Male	married	0	1	monthly	y	
12	100664	33	50631	Female	married	0	2	monthly	y	

Instead of the values, we now see the value labels for some variables. We also see that *marital* has a variable label attached to it: *Marital Status*.

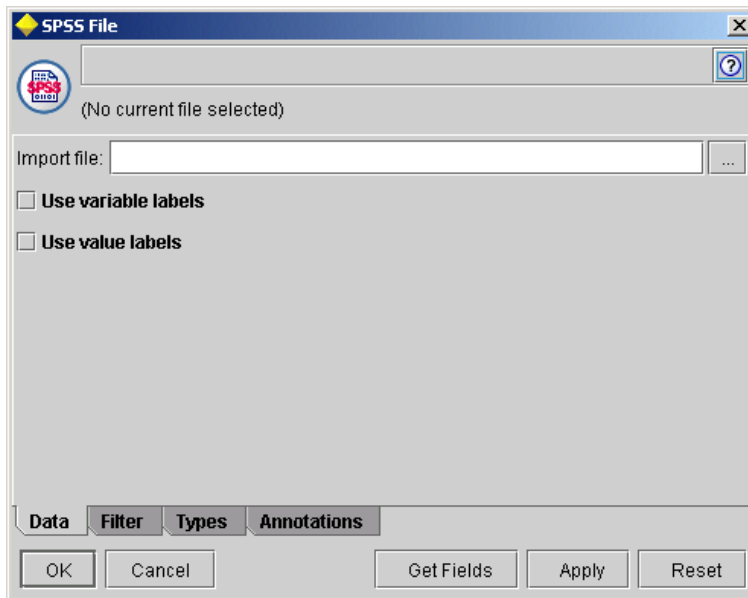
To read an SPSS data file into Clementine, place the SPSS File node on the Stream Canvas.


Click the **SPSS File** node from the Sources palette
Click in an open area on the left side of the Stream Canvas

To connect this node to the required SPSS data file:

Double-click on the **SPSS File** node, which will bring you into editing mode

Figure 3.16 SPSS Import Node Dialog



As in the previous examples, the file name and directory can be specified using the file list button  (or typed directly).

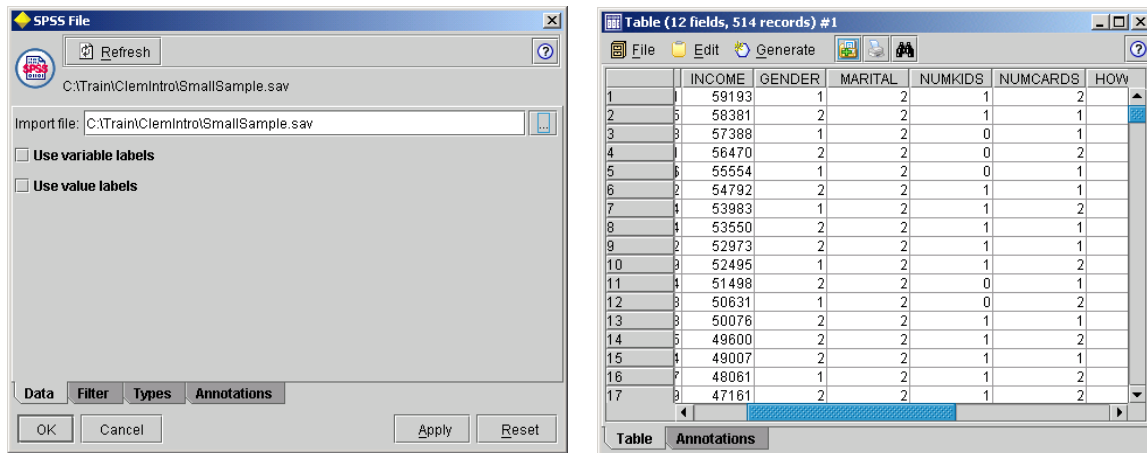
Select **SmallSample.sav** from **C:\Train\ClemIntro**

The SPSS File dialog contains check boxes for whether Variable Labels and Value labels should be imported from SPSS. Selecting the *Use Variable Labels* check box allows the use of the descriptive variable label to represent the field name in Clementine, as opposed to the usually shorter variable name. Similarly, the *Use Value Labels* option will import the SPSS value labels from the file, rather than the original numeric or symbolic values.

We will show the effect of different choices. First, we do not check any options.

Click **OK** to close the **SPSS File** dialog box
Add a **Table** node and connect the **SPSS File** node to the **Table** node
Execute the stream

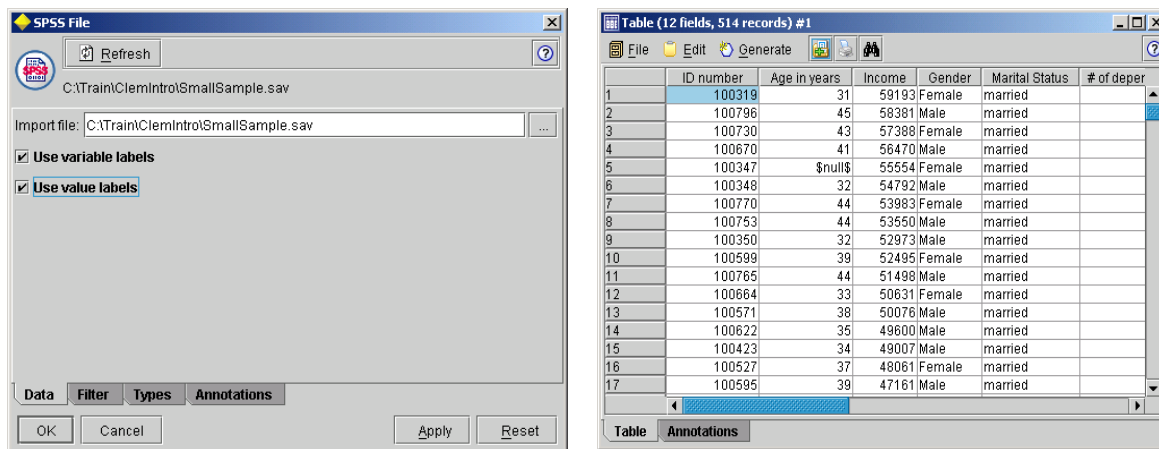
Figure 3.17 SPSS Import Dialog Box: Skipping Variable and Value Labels



The field names in Clementine are the same as the SPSS variable names. We lose the variable label information. This isn't a problem, since the field names themselves are pretty clear. The data values in Clementine are the same as the SPSS values. Having Clementine read the values instead of the value labels is not recommended. The first reason is that identification information is lost. Secondly Clementine, by default, will treat marital status coded 1,2,3 as numeric instead of discrete, which could well lead to potential problems later in the data-preparation and modeling phases if the modeling algorithm of choice requires a discrete outcome. Often, a better choice is to check both options.

- Double-click on the **SPSS File** node
- Click the **Use variable labels** and **Use value labels** check boxes
- Check **OK** to close the **SPSS File** dialog box
- Execute** the **Table** node

Figure 3.18 SPSS Import Dialog: Using Variable and Value Labels



When variable and values labels are imported, we have a more descriptive dataset that corresponds to the original SPSS file. It not only contains more informative values, but fields like *Gender* and *Marital Status* are correctly treated as symbolic.

Reading Data Using ODBC

Using its Database source node, Clementine can read data directly from databases and other data sources supporting ODBC (Open Database Connectivity protocol). Within the Database node, fields can be selected from a database table or SQL can be used to access data. Before reading data in this way, ODBC drivers must be installed (drivers for a number of databases are included in the SPSS Data Access Pack on the Clementine CD) and data sources must be defined. We will illustrate how to declare a database as a data source within Windows™ and then how to access that database using the Database source node.

The data tables we will access are found in the Access database *custandhol.mdb*. The database contains three tables: *custtravel1*, *custtravel2* and *holtravel*. The customer tables contain information on the customers of a travel company, including the holiday reference code, length of holiday and cost of holiday. The holiday table contains information on each of the individual holidays offered by the company including the location and type of accommodation.

Declaring a Data Source

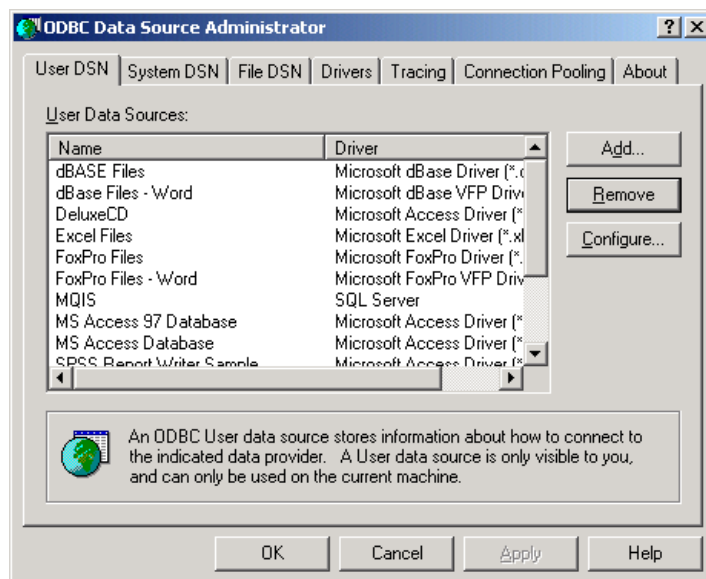
Before a database can be accessed via the ODBC method, it must be declared as an ODBC User Data Source using the ODBC Data Source Administrator, found within the Windows Control panel. An ODBC User Data Source stores information about how to connect to an indicated data provider. In this section we will demonstrate how to declare a database as an ODBC User Data Source using the Access database *custandhol.mdb*, working with Windows 2000.

- Go to the **Start** menu
- Click **Settings..Control** panel
- Double-click the icon **Administrative Tools**

If ODBC is installed on your machine, there will be an icon labeled Data Sources (ODBC) within the Administrative Tools.

- Double-click on the icon labeled **Data Sources (ODBC)**

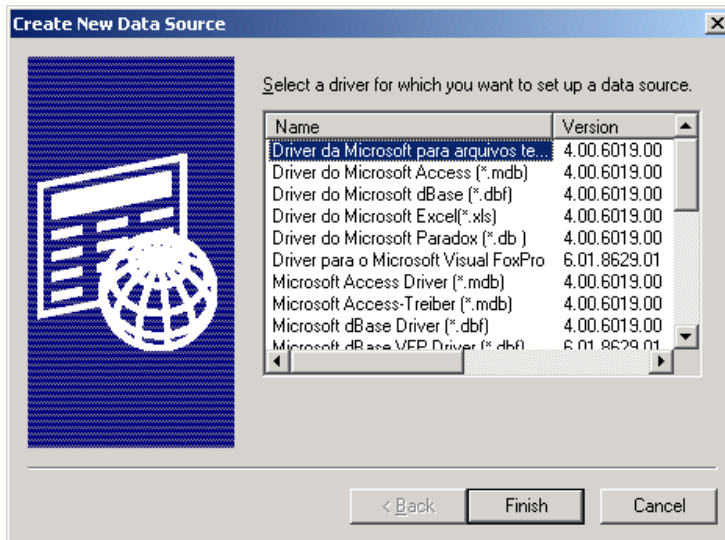
Figure 3.19 ODBC Data Source Administrator



To declare a database as a Data Source we must add it to the User Data Sources list.

Click on the **Add** button

Figure 3.20 Create New Data Source Dialog

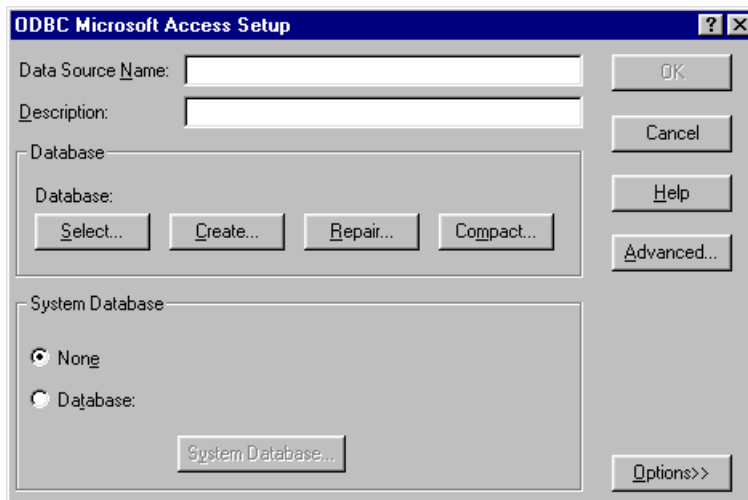


This dialog box contains a list of drivers that are installed on your machine.

From the driver list select **Microsoft Access Driver (*.mdb)**
Click on the **Finish** button

A Setup dialog box will appear and will have various controls dependent on the type of driver selected. The Microsoft Access Setup dialog box is shown below.

Figure 3.21 Microsoft Access Dialog Used to Set Up a Data Source

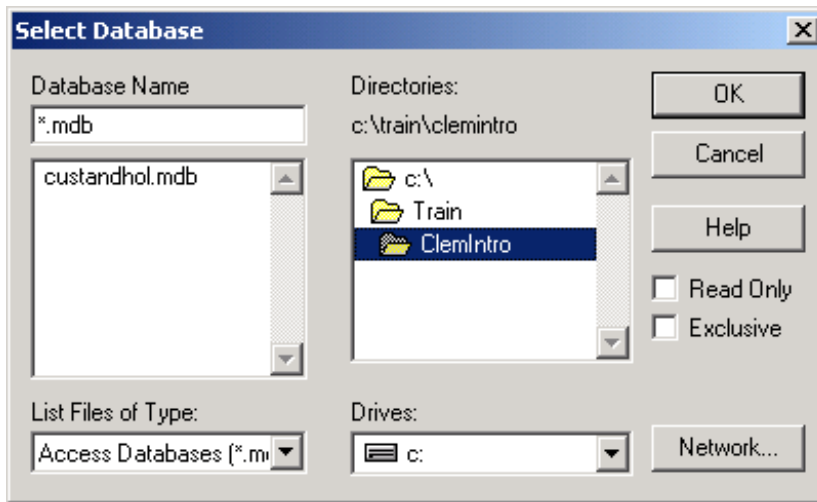


For the Access driver, the data source needs to be named and the particular database selected.

Type the name **HOLIDAYS** in the **Data Source Name** text box
Click the **Select** button

The Select Database dialog box, shown below, allows you to select the particular database that you wish to add as a data source. The file can be located using the drives and directories controls.

Figure 3.22 Select Database Dialog



Select the database **custandhol.mdb** in the **c:\Train\ClemIntro** directory
Click **OK** to return to the Microsoft Access Setup dialog box
Click **OK** to return to the ODBC Data Source Administrator

The data source name should now be listed in the User Data Sources list.

Click **OK** to return to the Control panel

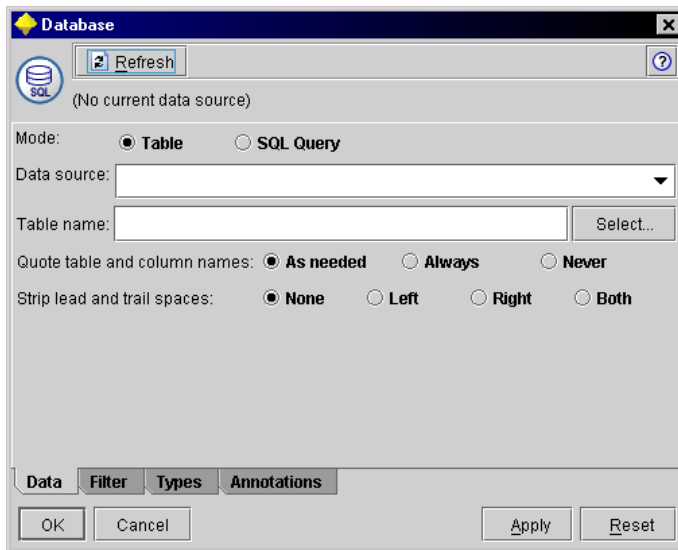
The database has now been declared as a data source and is available to Clementine and other applications via ODBC.

Accessing a Database Using the Database Source Node

In this section we will illustrate the use of the ODBC source node to read data from an Access database from within Clementine:

Clear the stream by choosing **Edit..Clear Stream**
Select the **Database** node from the Sources or Favorites palette and place it on the Stream Canvas
Edit (double-click) the **Database** source node

Figure 3.23 Database Node Dialog



The Mode options (Table or SQL Query) allow you select whether you wish to view the Tables/Views within the database, or to enter/load a SQL Query against the database. In our example we have to read a table from the Holidays database, so Table mode is applicable.

To access the data:

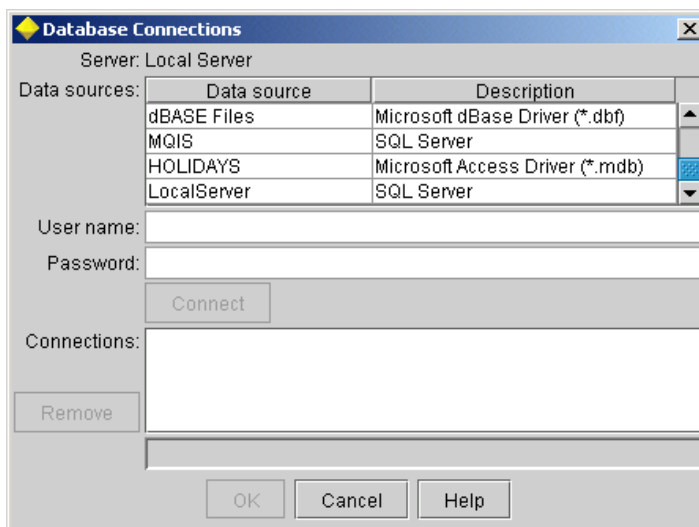
Click the **Data source** drop-down arrow

Since we have not connected to a database yet, we have to add a database connection.

Click **Add New Database Connection**

A dialog box will appear allowing you to select the database:

Figure 3.24 Connect to ODBC Data Source Dialog

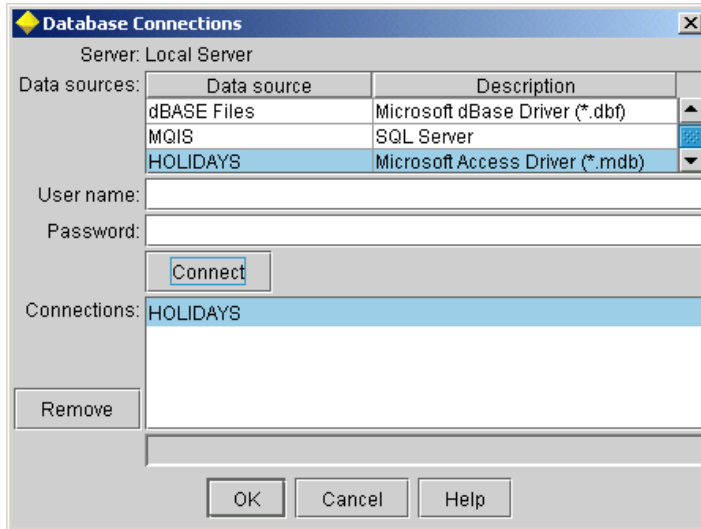


By scrolling down the Data sources list, the database defined as a data source in the previous section can be selected. If required, you can specify a username and password for the database.

Scroll down the **Data sources:** list and select the **HOLIDAYS** data source
Click the **Connect** button

The Holidays data source now appears in the Connections box.

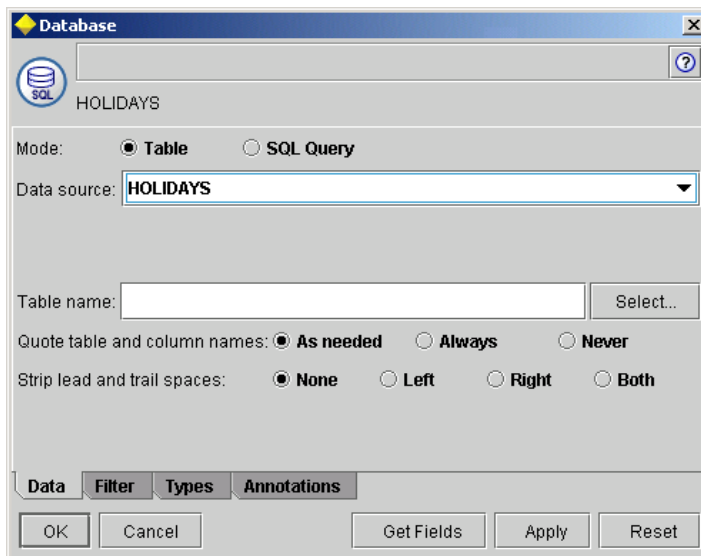
Figure 3.25 Database Connections Dialog (After Connection to Data Source)



Click **OK**

Returning to the Database dialog, we see that Holidays is selected as the Data source.

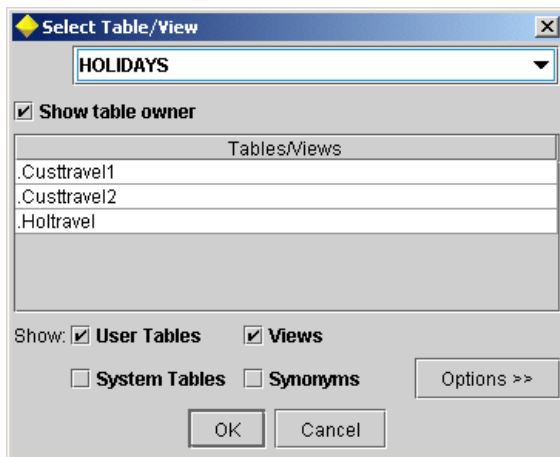
Figure 3.26 Data Source Defined in Database Dialog



The next step is to select the database table.

Click on the **Select...** button

Figure 3.27 Select Table/View Dialog



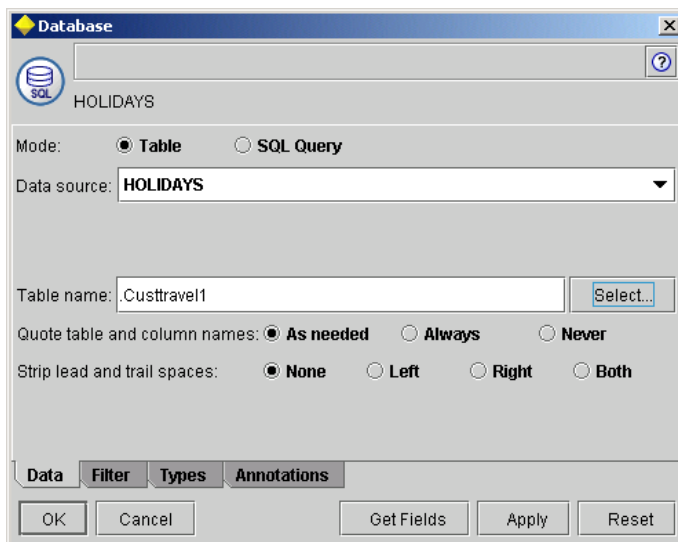
The tables within the selected database appear. When *Show table owner* is checked the owner of each table/view is shown. This will raise an error if the ODBC driver in use does not support the table owners.

Uncheck the Show table owner check box

The user-defined tables are shown in the Tables/Views list. You may choose whether or not to see system tables. Note that only one Table/View may be selected at any one time.

Click on **custtravel1** in the Tables/Views list
Click **OK** to return to the Database dialog box

Figure 3.28 Completed Database Dialog



In this dialog, options control how spaces in string fields are handled. Strings can be left and/or right trimmed. There is a further specification for quotes in relation to tables and field names. By default, Clementine will quote table and column names that include field names. These settings can be altered to always or never quote table and column names.

If you wish to read in all of the fields from the selected table then the OK button can simply be clicked to return to the Stream Canvas. Alternatively you can select which fields you wish to read into Clementine

(use Filter tab), and then return to the Stream Canvas. This node, as a Source node, contains a Types tab that can be used to examine and set field types.

Click **OK** to return to the Stream Canvas
 Connect the **Database** source node to a **Table** node
Execute the stream

Figure 3.29 Data from Custravell Table in Access Database

	CUSTID	NAME	DOB	GENDER	REGION
1	GS101001	Roberta Go...	1966-02-0...	Female	Northern Ir...
2	GS101002	Joan Ranger	1953-07-1...	Female	South West
3	GS101003	Mrs B. Emp...	1961-04-2...	Female	North West
4	GS101004	Mr W. Gum...	1958-09-2...	Male	Scotland
5	GS101005	Mr B. Emw	2025-05-1...	Male	East Anglia
6	GS101006	Mandy Stev...	1948-05-2...	\$null\$	North East
7	GS101007	Ivor Square...	1962-09-2...	Male	North East
8	GS101008	Wendy Pen...	1956-01-0...	Female	North East
9	GS101009	Mr K.N. Owl...	1966-08-1...	Male	North East
10	GS101010	Jill Banker	2026-07-0...	Female	North East
11	GS101011	Robert Exec	1951-09-0...	Male	South West
12	GS101012	Ms F. Chip	1962-04-1...	Female	South West
13	GS101013	Cheryl Smith	1959-01-2...	Female	South West
14	GS101014	Mr B. Ilder	2010-04-2...	Male	South West
15	GS101015	Tracey Wal...	1966-10-2...	Female	London & ...
16	GS101016	Julian Bond	1952-06-2...	Male	London & ...
17	GS101017	Blodwin Sh...	1956-10-1...	Female	London & ...
18	GS101018	Jack Potato	1937-05-0...	Male	North West
19	GS101019	Mr T. Brown	1979-09-0...	Male	North West
20	GS101021	Mr B. Ilder	1950-08-2...	Male	North West

The data table from the Access database has been read into Clementine.

If you wish to access a number of fields from different tables within a database, just use the following procedure:

- Use a different Database source node for each table within the database
- Link the relevant fields together in a stream containing Append and Merge nodes (examples appear in the *Data Manipulation with Clementine* training course).

Other Data Formats

The SAS Import node allows you to bring SAS data into your data mining session. You can import four types of files:

- SAS for Windows/OS2 (.sd2)
- SAS for UNIX (.ssd)
- SAS Transport File (.tpt)
- SAS version 7/8 (.sas7bdat)

When the data are imported, all variables are kept and no variable types are changed. All cases are selected. Similar the SAS File import node, the SAS Export File node allows you to write data in SAS format to be read into SAS. You can export in three SAS file formats: SAS for Windows/OS2, SAS for UNIX, or SAS Version 7/8.

Defining Data Field Types

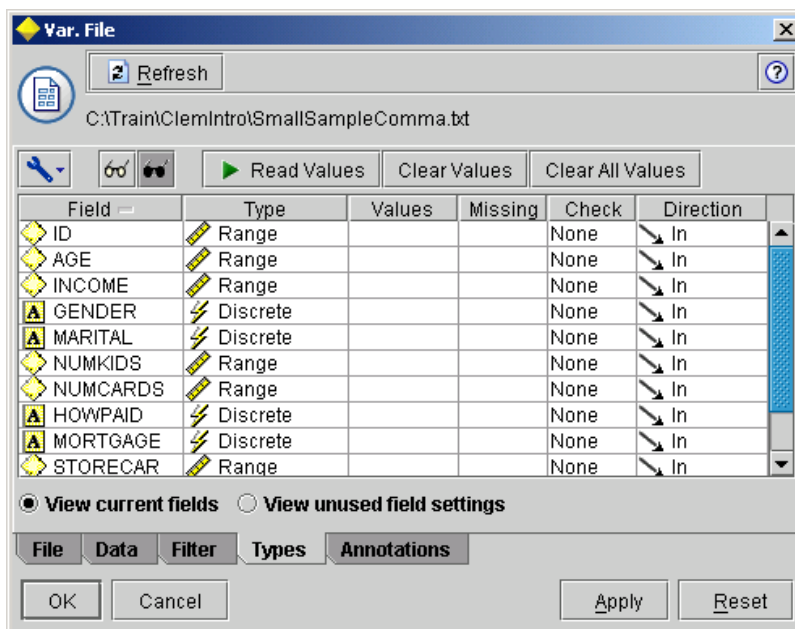
After specifying your data source, the next step before data mining is to define the type information for each of the fields within the data. The type information for each field must be set before the fields can be used in the Modeling nodes.

Field types can be set in most source nodes (click on the Types tab) at the same time you define your data, or in the Type Node (found in the Field Ops palette) if you need to define a field type later in your stream. In some earlier examples we have seen the Type tab in data source nodes. We now turn to our postponed discussion of type definitions. We will define the field type in a source node, but we could also use a Type node to do this.

As a data source we will open *SmallSampleComma.txt*.

- Clear the stream by choosing **Edit..Clear Stream**
- Place a **Var. File** node on the Stream Canvas
- Edit** the node and specify **SmallSampleComma.txt** in the **c:\Train\ClemIntro** directory as data file
- Click **Types** tab

Figure 3.30 Types Tab of Var. File Node



The Types tab in a data source node or the Type node controls the properties of each field: type, direction, and missing value definitions. This node also has a Check facility that, when turned on, examines fields to ensure that they conform to specified type settings: for example, to check whether all the values in a field are within a specified range. This option can be useful for cleaning up data sets in a single operation.

In this section we concentrate on the type and direction definitions. Other type specifications (missing values) will be discussed in later chapters.

Field Type Definition

The Type column in the Types tab of source nodes (and the Type node) describes the data type of the field, which determines how Clementine will use the field. Clementine distinguishes among:

- **Range.** Used to describe numeric values such as a range of 0-100 or 0.75-1.25. A range value may be an integer, real number, or date/time.
- **Discrete.** Used for string values when an exact number of distinct values is unknown.
- **Flag.** Used for data with two distinct values such as Yes/No or 1, 2.
- **Set.** Used to describe data with multiple distinct values, each treated as a member of a set, such as small/medium/large.
- **Typeless.** Used for data that does not conform to any of the above types or for set types with too many members. It is useful for cases in which the type would otherwise be a set with many members (such as an account number). When you select Typeless for a field's type, the field direction is automatically set to None (meaning the field cannot be used in modeling). The default maximum size for sets is 250 unique values. This number can be adjusted or disabled in the Stream Properties dialog.

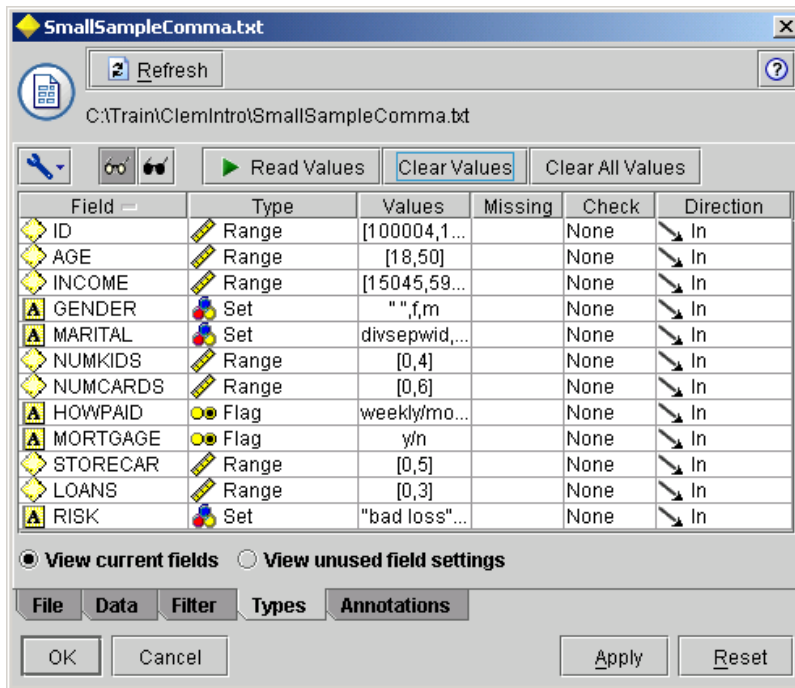
At this stage (see figure above), the fields in *SmallSampleComma.txt* are in a partially instantiated state. Instantiation refers to the process of reading or specifying information such as type and values for a data field. Data with totally unknown types are considered uninstantiated. Fields are referred to as partially instantiated if the program has some information about how they are stored (symbolic or numeric), but the details are incomplete. For example, the discrete type is temporarily assigned to a symbolic field, until it can be determined if it is either a Set or Flag type. The Range type is given to all numeric fields, whether they are fully instantiated or not. When all the details about a field are known, including the type and values, it is considered fully instantiated and Set, Flag, or Range is displayed in the Type column.

During the execution of a data stream instantiation occurs when the field Values settings in the Type tab are set to Read or Read+ (meaning that values should be read, or current values retained and new values added when the data are read). Once all of the data have passed through the data source or Type node, all fields become fully instantiated.

In reading the data values through the source node, Clementine identifies the type of each field (when the field's Values property is set to Read or Read+). To check the definition of types, edit the source node (or Type node) after data have passed through it. We can force data through the source node by placing and executing a node downstream of it; alternatively we can click the Read Values button, which reads the data into the source node or Type node (if Read Values is clicked from within a Type node).

Click the **Read Values** button, then click **OK**

Figure 3.31 Types Tab after Values Read (Fully Instantiated Types)



Fields ID, AGE, and INCOME are typed as Range (with the lower and upper bounds in the Values column). HOWPAID (with values weekly/monthly) and MORTGAGE (with values y/n) are typed as Flag. MARITAL and RISK are typed as Set. Notice that GENDER is typed as Set, due to the fact that not only f and m appear as values, but also a space " ".

If execution is interrupted, the data will remain partially instantiated. Once the types have been instantiated, the values of a field in the Values column of the Types tab are static at that point in the stream. This means that any upstream data changes will not affect the stored Values of a particular field, even if you re-execute the stream. To change or update the Values based on new data or added manipulations, you need to edit them in the Types tab (or re-instantiate the field, by setting its Values column entry to Read or Read+ and passing data through the node).

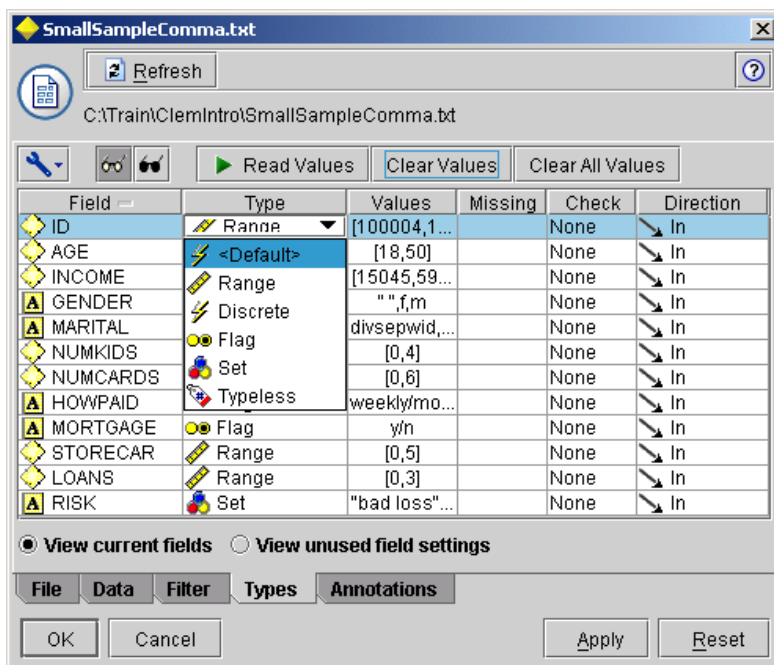
Numeric fields that represent sets or flags must be forced to the correct field type by specifying Discrete. They will then be assigned flag or set type when fully instantiated.

The easiest way to define the type of each data field is to initially allow Clementine to autotype by passing the data through the source node (or Type node), and then manually editing any incorrect types. We will demonstrate this approach using the previously constructed stream.

As an example of changing the field type, consider ID. This field contains a unique reference number and is better defined as Typeless.

Click in the **Type** column for the field **ID**
Choose **Typeless** from the list

Figure 3.32 Context Menu for Entries in Type Column



After selecting Typeless, the ID's type will change to Typeless and its direction will change to NONE (not shown). We discuss direction in the following section.

Click **OK**

Field Direction

The direction of a field is relevant only to modeling nodes. The four available directions are:

IN	The field will be used as an input or predictor to a modeling technique. (i.e. a value on which predictions will be based).
OUT	The field will be the output or target for a modeling technique. (i.e. the field to be predicted).
BOTH	Direction suitable for the Apriori, GRI, and Sequence modeling nodes. Allows the field to be both an input and an output in an association rule. All other modeling techniques will ignore the field.
NONE	The field will not be used in modeling.

Setting the direction for a field is done in the same way as setting the type: click on the Direction value for a field and choose the appropriate direction from the drop-down list. Multiple fields can be selected and properties like direction or type changed from the context menu (right-click on any of the selected fields).

Note

Setting the direction of fields may be performed later in the project if you are in the initial stages of mining or are not planning on using any of the Modeling techniques available.

Saving a Clementine Stream

To save our Clementine stream for later work:

- Click **File..Save Stream As** and move to the **c:\Train\ClemIntro** directory
- Type **SmallCommaDef** in the File name text box
- Click the **Save** button

The File menu also allows you to save (and Open) a State file (which contains the stream and any models stored in the Models palette – discussed in later chapters) and a Project file (which can contain streams, graphs, reports, and generated models– thus organizing elements related to an analysis project). Also, you can add the saved stream to the current project by clicking the *Add file to project* check box.

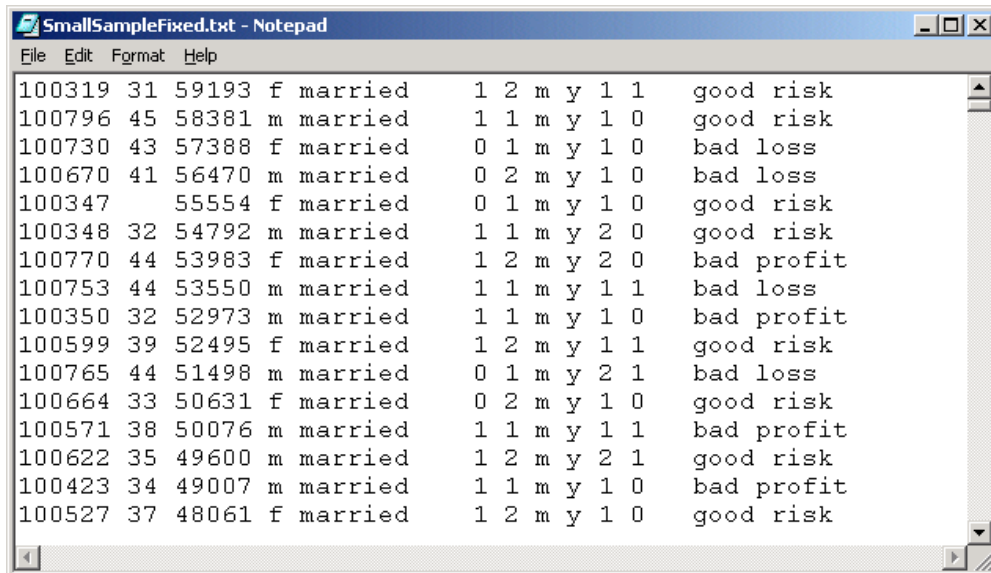
Summary

In this chapter you have been given an introduction on how to read data into Clementine, define the types of the fields within the data, and view the data file.

Appendix: Reading Data from Fixed-field Text Files

Data in fixed column format can be read into Clementine with the Fixed File node in the Sources palette. We see an example of such a file below (shown in Notepad).

Figure 3.33 Fixed-field Text File



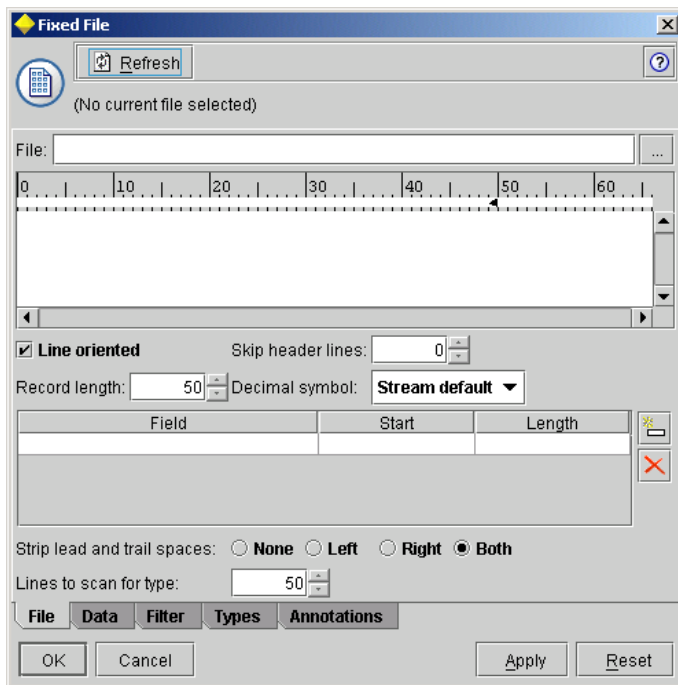
Each field is located in the same column positions on every record in the data file. When instructing Clementine how to read such a file, you must know the column position(s) that each variable occupies. Typically the program or individual creating the data file can supply this information. In our example, we have the following information available.

Table 3.1 Information about Field Start Position and Length

Field	Start Position	Length
ID	1	6
AGE	8	2
INCOME	11	5
GENDER	17	1
MARITAL	19	7
NUMKIDS	30	1
NUMCARDS	32	1
HOWPAID	34	1
MORTGAGE	36	1
STORECAR	38	1
LOANS	40	1
RISK	44	10

Data in fixed column format can be read into Clementine using the Fixed File node in the Sources palette.

- Start a new stream by choosing **File..New Stream**
- Click the **Fixed File** node in the **Sources** palette
- Click in an empty area on the left side of the Stream Canvas
- Double-click on the **Fixed File** node in the Stream Canvas

Figure 3.34 Fixed File Dialog

This dialog has much in common with the Var. File dialog, such as the handling of leading and trailing spaces in symbolic fields, how many lines to scan for type, skipping lines, a Data tab (to manage data storage), a Filter tab (to include/exclude fields or to rename fields) and a Types tab (to manage field types).

To read the data, we first have to specify the data file


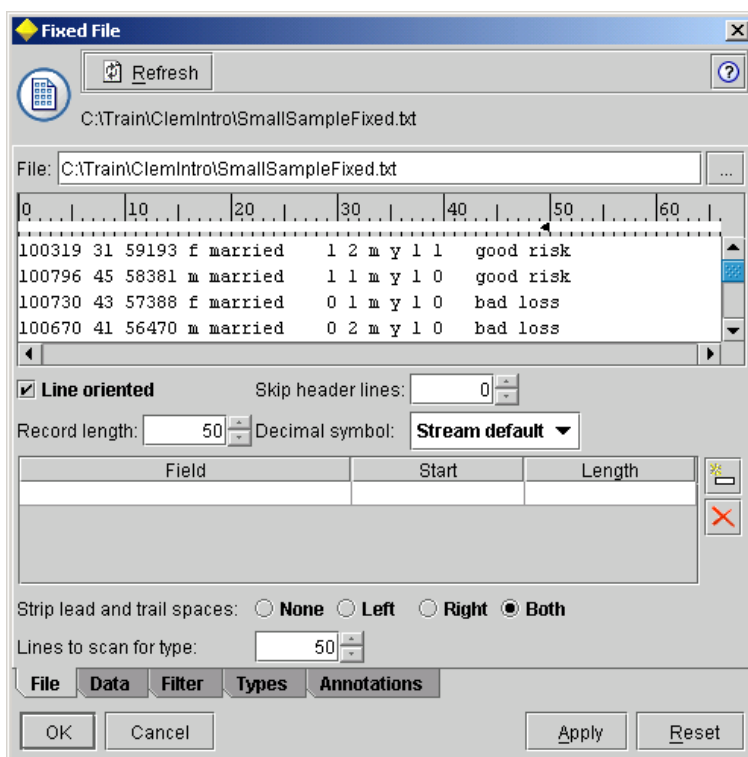

- Click the file list button , and then move to the **c:\Train\ClemIntro** directory
- Select **SmallSampleFixed.txt** in this directory, and then click **Open**

Figure 3.35 Fixed File Node Dialog: Data Preview



There are two ways to define fields:

- Interactively: You specify fields using the data preview above. The ruler at the top of the preview window helps you to measure the length of variables and specify the breakpoint between them. You can specify breakpoint lines by clicking in the ruler area above the fields. Each breakpoint line automatically adds a new field to the field table below. Start positions indicated by the arrows are automatically added to the Start column in the table below. Breakpoints can be moved by dragging and can be discarded by dragging them outside the data preview region.
- Manually: You specify fields by adding empty field rows to the table below. Double-click in a cell or click the New Field button  to add new fields. Then, in the empty field row, enter a field name, a start position and a length. These options will automatically add breakpoint lines (arrows) to the data preview canvas that can be easily adjusted.

As we have information about starting positions of the fields and field lengths readily available, we choose the second alternative.

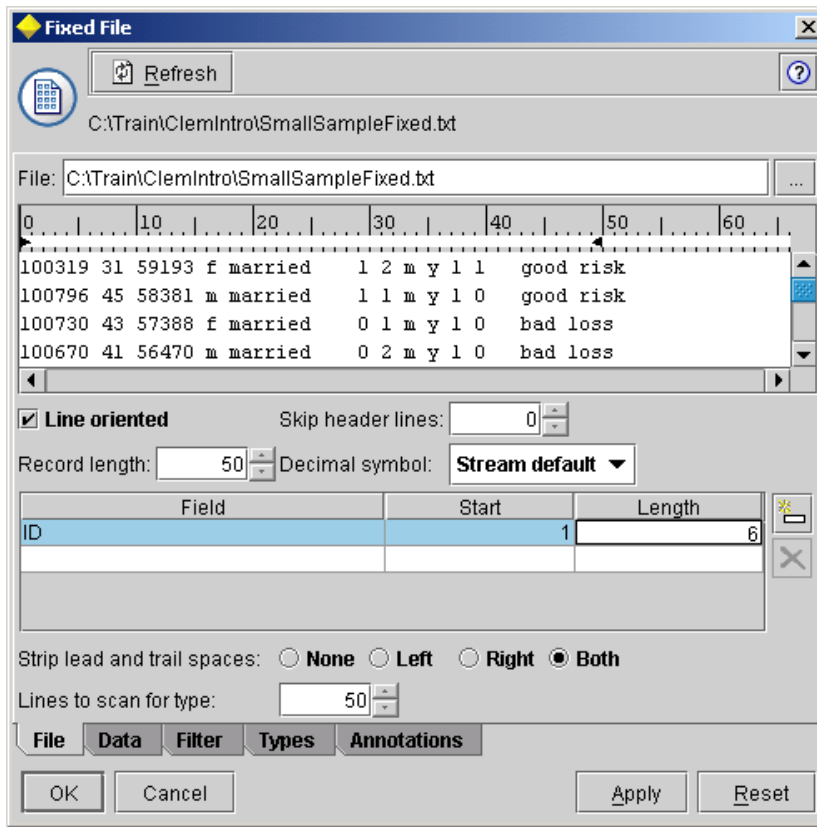
Double-click in the cell in the **Field** column

Specify **ID** as fieldname

Press the tab key to move on to the **Start** column (or double-click in the cell in the **Start** column). Note, that the start position is already specified as 1 by default, so we can move to the next specification

Press the tab key to move on to the **Length** column. Replace the default value 1 with **6**

Figure 3.36 Fixed File Dialog: Specifying Field Information

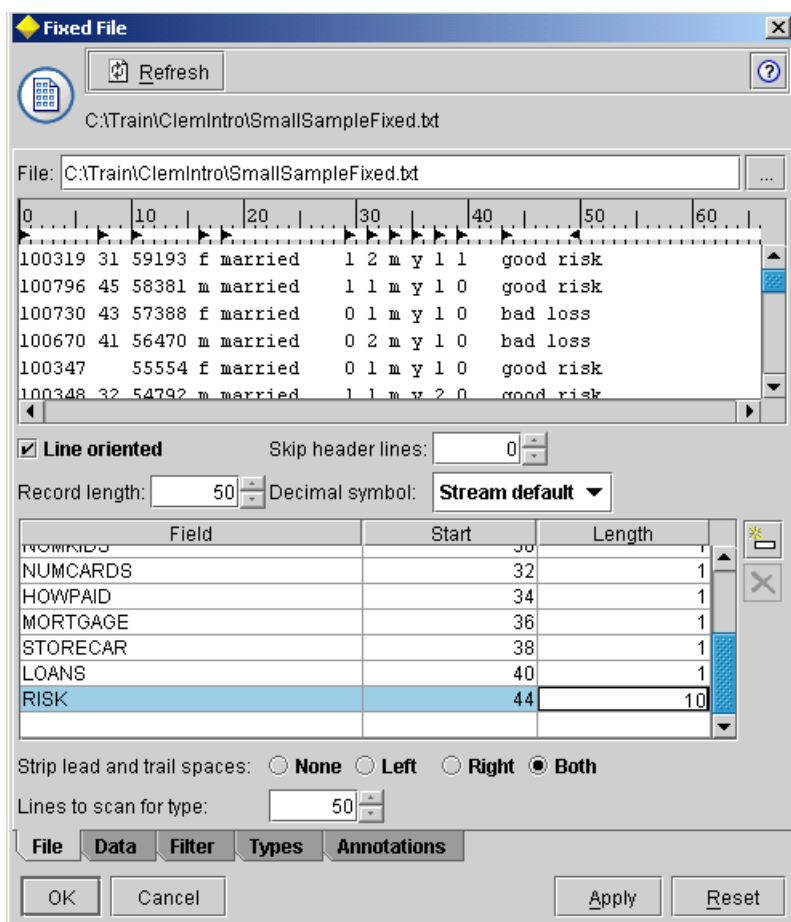


To define the next field (AGE, starting at position 8, length 2):

- Double-click in cell in the **Field** column below **ID**
- Specify **AGE** as fieldname
- Move on to the **Start** column and type **8**
- Move on to the **Length** column and type **2**

The rest of the fields are defined in the same way. The final result is shown below.

Figure 3.37 Fixed File: All Fields Defined



Any fields that are not defined are skipped when the file is read.

Although the definitions look correct, there is still a detail remaining. Notice that at position 50 in the preview pane, there is an end-of-line indicator (↵). This is a result of the default record length of 50. Unless we make a change, the last characters of RISK won't be read. To correct this:

Set the record length to, say, 60, either by moving the end-of-line character ↵ or by typing **60** in the **Record length:** text box

Now that our definitions are complete, we can move on and check if the data are read correctly.

Click **OK**
 Connect a **Table** to the data source node
Execute the stream

Figure 3.38 Table Window Showing Data from Text File

	ID	AGE	INCOME	GENDER	MARITAL	NUMKIDS	NUMCARDS	HOWPAID	MORTGAGE	STORECAR	LOANS	RISK
1	100319	31	59193	f	married	1	2	m	y	1	1	good risk
2	100796	45	58381	m	married	1	1	m	y	1	0	good risk
3	100730	43	57388	f	married	0	1	m	y	1	0	bad loss
4	100670	41	56470	m	married	0	2	m	y	1	0	bad loss
5	100347	\$null\$	55554	f	married	0	1	m	y	1	0	good risk
6	100348	32	54792	m	married	1	1	m	y	2	0	good risk
7	100770	44	53983	f	married	1	2	m	y	2	0	bad profit
8	100753	44	53550	m	married	1	1	m	y	1	1	bad loss
9	100350	32	52973	m	married	1	1	m	y	1	0	bad profit
10	100599	39	52495	f	married	1	2	m	y	1	1	good risk
11	100765	44	51498	m	married	0	1	m	y	2	1	bad loss
12	100664	33	50631	f	married	0	2	m	y	1	0	good risk
13	100571	38	50076	m	married	1	1	m	y	1	1	bad profit
14	100622	35	49600	m	married	1	2	m	y	2	1	good risk
15	100423	34	49007	m	married	1	1	m	y	1	0	bad profit
16	100527	37	48061	f	married	1	2	m	y	1	0	good risk
17	100595	39	47161	m	married	1	2	m	y	1	1	good risk
18	100565	38	46823	m	married	0	1	m	y	1	1	good risk
19	100470	36	45949	f	married	1	1	m	y	2	0	bad profit
20	100282	30	45715	m	married	0	2	m	y	1	1	good risk
21	100679	42	45584	f	married	1	1	m	y	1	0	good risk

Again, note the \$null\$ value for the 5th record, field AGE. Looking back at the file *SmallSampleFixed.txt* we see that this person had no value for AGE, so a missing value, represented by the value \$null\$, was assigned to AGE.

Chapter 4

Data Quality

Overview

- Missing value definitions
- Introduce the Quality node
- Use the Data Audit node to examine the distribution of values for all fields

Objectives

This session aims to introduce some of the ways in which Clementine can be used to discover the accuracy, completeness, and overall behavior of your data.

Data

To illustrate how Clementine deals with missing information, we use a small data file containing missing values, *SmallSampleMissing.txt*. This file has one record per account held by customers of a financial organization. It contains demographic details on the customer, such as income, gender, and marital status.

To illustrate the Data Audit node, a version of the data file introduced in the previous chapter will be used, *Risk.txt*. The file contains information concerning the credit rating and financial position of individuals, along with basic demographic information such as marital status and gender.

Introduction

Data sets always contain problems or errors such as missing information and/or spurious values. Therefore, before data mining can begin, the quality of the data must be assessed. This involves both checking for blank or missing information and understanding the range and distribution of values within each field. Through examining the properties of each field the user will be able to have a better understanding of the fields themselves and any models that are built. The higher the quality of the data used in data mining, the more accurate the predictions or results.

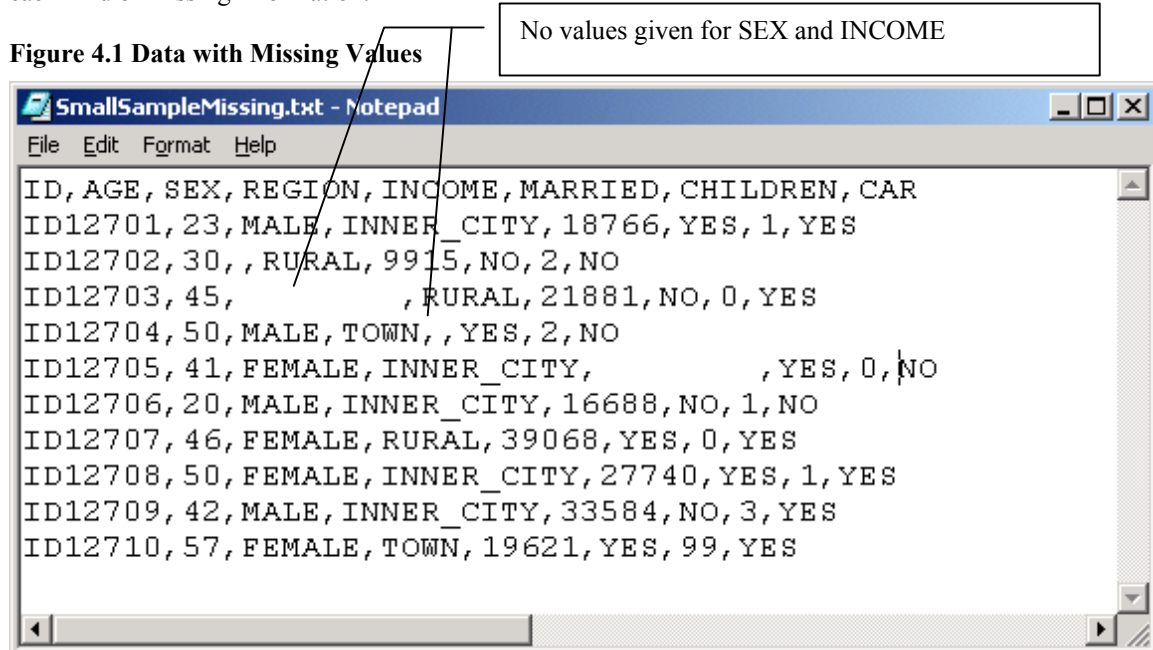
Clementine provides several nodes that can be used to investigate the integrity of data. In the following sections we will introduce the Quality node, to study the completeness of the data, the Data Audit node to examine both symbolic and numeric fields in a single analysis.

Missing Data in Clementine

In the previous chapter we covered two of the field properties that the Type tab controls: type and direction. Another important property available in this node concerns blanks or missing data.

In Clementine there are a number of different types of representations of missing data. First, a field may be completely blank. Clementine calls such missing information white space string if the field is symbolic and null value (non-numeric) if the field is numeric. There is also the instance in which a non-numeric character appears in a numeric field. Clementine also refers to this as a null value or non-numeric missing. Finally, when entering data, predefined codes may be used to represent missing or invalid information. Clementine refers to such codes as value blanks. The file *SmallSampleMissing.txt* (shown below) contains examples of each kind of missing information.

Figure 4.1 Data with Missing Values



Note that SEX is not given for ID12702 and has the value " " for ID12703. Similarly, INCOME is not given for ID 12704 and ID12705. Furthermore, ID12710 has the value 99 for CHILDREN (number of children) because the number of children was unknown and the database administrator decided to put the value 99 in this field to represent unknown.

So in this file we have different kinds of missing information. In the next section we will see how Clementine deals with it.

Assessing Data Quality

To illustrate the handling of missing data we will open *SmallSampleMissing.txt* and assess the quality of the data. We will have Clementine identify the field types while creating a Table.

- If the Stream Canvas is not empty, start a new stream by clicking **File..New Stream**
- Select the **Var. File** node and place it on the Stream Canvas.
- Edit** the node and set the file to **SmallSampleMissing.txt** held in the **c:\Train\ClemIntro** directory
- Make sure the **Read field names from file** option is checked
- Click the **Types** tab, then right-click any field and click **Select All** from the context menu
- Right-click any field, and then click **Set Values..<Read>** from the context menu
- Click **OK**

Add a Table node and connect the **Var. File** node to the **Type** node
Execute the **Table** node

Figure 4.2 Data Table Showing Blanks and Missing Values

	ID	AGE	SEX	REGION	INCOME	MARRIED	CHILDREN	CAR
1	ID12701	23	MALE	INNER_CITY	18766	YES	1	YES
2	ID12702	30		RURAL	9915	NO	2	NO
3	ID12703	45		RURAL	21881	NO	0	YES
4	ID12704	50	MALE	TOWN	\$null\$	YES	2	NO
5	ID12705	41	FEMALE	INNER_CITY	\$null\$	YES	0	NO
6	ID12706	20	MALE	INNER_CITY	16688	NO	1	NO
7	ID12707	46	FEMALE	RURAL	39068	YES	0	YES
8	ID12708	50	FEMALE	INNER_CITY	27740	YES	1	YES
9	ID12709	42	MALE	INNER_CITY	33584	NO	3	YES
10	ID12710	57	FEMALE	TOWN	19621	YES	99	YES

The table shows three examples of missing information.

- SEX has been left blank for the records with ID12702 and ID12703 (see the text file: ID12702 has no value for SEX in the text file, ID12703 has spaces " " as value in the text file).
- INCOME has a non-numeric value, appearing as \$null\$ in the table, for record ID12704 and ID12705. The value \$null\$ is assigned by Clementine in case a value is undefined and is considered by Clementine as missing information. The reason that Clementine assigned the \$null\$ value, instead of leaving it empty as with SEX, is that INCOME is typed as Range (as opposed to the discrete type of SEX).
- CHILDREN has a user defined missing value of 99 for record ID12710.

Click **File..Close** to close the **Table** node
 Double-click the **Var. File** node and click the **Types** tab

Figure 4.3 Types Tab: File with Missing Data

Field	Type	Values	Missing	Check	Direction
ID	Set	ID12701,I...		None	In
AGE	Range	[20,57]		None	In
SEX	Set	"" " " ,F...		None	In
REGION	Set	INNER_CI...		None	In
INCOME	Range	[9915,390...		None	In
MARRIED	Flag	YES/NO		None	In
CHILDREN	Range	[0,99]		None	In
CAR	Flag	YES/NO		None	In

View current fields View unused field settings

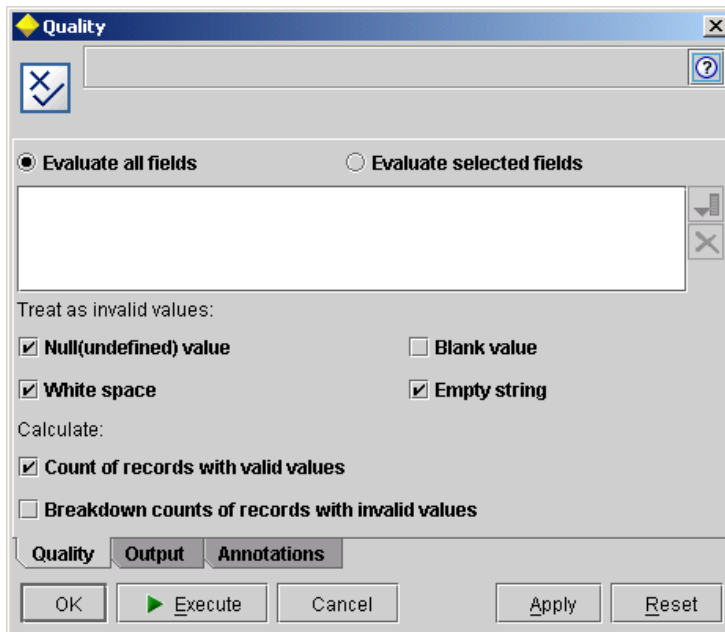
File Data Filter Types Annotations

Notice that SEX is assigned type set because four discrete values were found – see Values column (FEMALE, MALE, a set of space characters (white space), and an empty string) and that CHILDREN has a range of 0 through 99. The point is that for symbolic fields, Clementine will not automatically declare such values as missing and you, the user, should do so when appropriate. The Quality node, which will be discussed shortly, will report on such missing values even if they are not declared as missing in the Types tab of a source node (or Type node). However, if you know that such values should be identified as missing values, then there is an advantage in declaring them before data are read, since they then will not appear on the Values list for the field. In this case, SEX would be properly identified as type flag with values FEMALE and MALE. We will return to this point later.

The Quality node provides a report about the missing values in a data stream. It checks for missing values or blanks, is located in the Output palette, and is a terminal node (no connections can lead from an output node). The Quality node can take into account all of the missing value definitions previously mentioned.

Place a **Quality** node from the Output palette into the Stream Canvas and connect the **Var. File** node to it
 Edit the **Quality** node

Figure 4.4 Quality Node Dialog



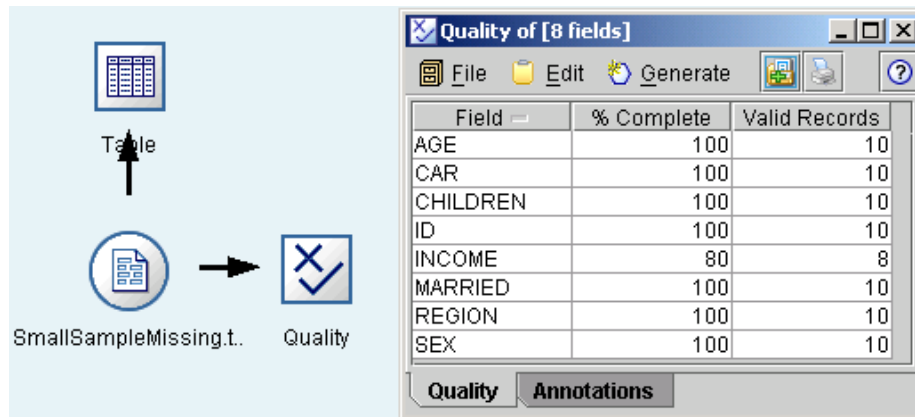
Check boxes control what the Quality node will report as missing. These include:

- **Null (undefined) value.** Considers system (\$null\$) values as invalid
- **Empty string.** Considers empty strings (no characters, not even spaces) as invalid
- **White space.** No visible characters, for example spaces; also includes empty strings
- **Blank value.** Considers user-defined missing values as invalid

To illustrate, we will begin by checking only the Null (undefined) value option and then extend it to all four modes.

Deselect the **White space** option
 Deselect the **Empty string** option
 Click the **Execute** button

Figure 4.5 Quality Output Window: Checking for Null Values Only



All fields but INCOME are reported to have 100% valid values. Income has missing information, the \$null\$ value, for two records. The missing values for SEX were not reported, nor the 99 for CHILDREN.

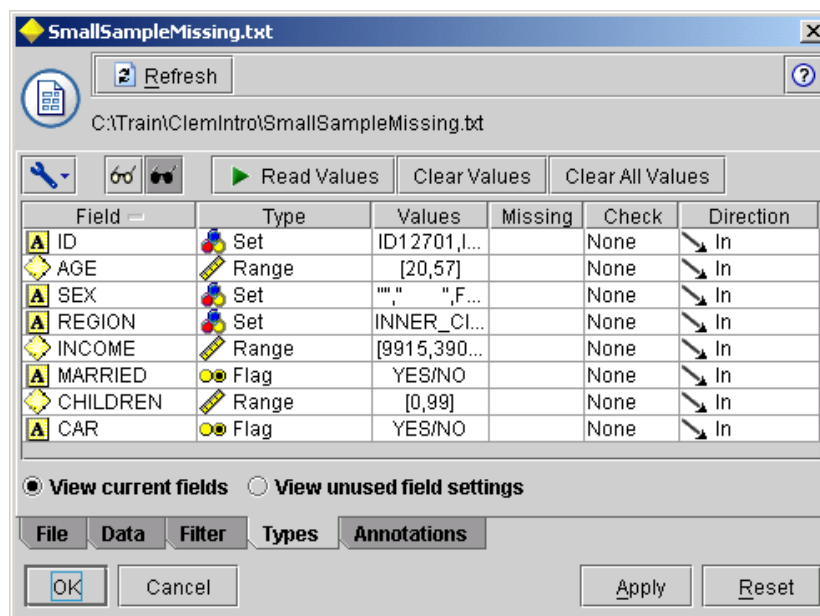
Note

This report can be sorted by values within any of the columns by clicking on the column header (here *Field*, *% Complete*, or *Valid Records*); choices will cycle through ascending order, original order, and descending order. When sorted by ascending or descending order, an upward or downward pointing icon indicates the sort order, while a dash indicates the original order (see *Field* column)

Next, let's check for the other forms of missing values. In order to do so, we need to define the values that we want considered as missing: in our example the value 99 for CHILDREN. To declare the value 99 for CHILDREN as missing we return to the Types tab in the source node (or we could add a Type node).

Edit the **Var. File** node (double-click the Var. File node)

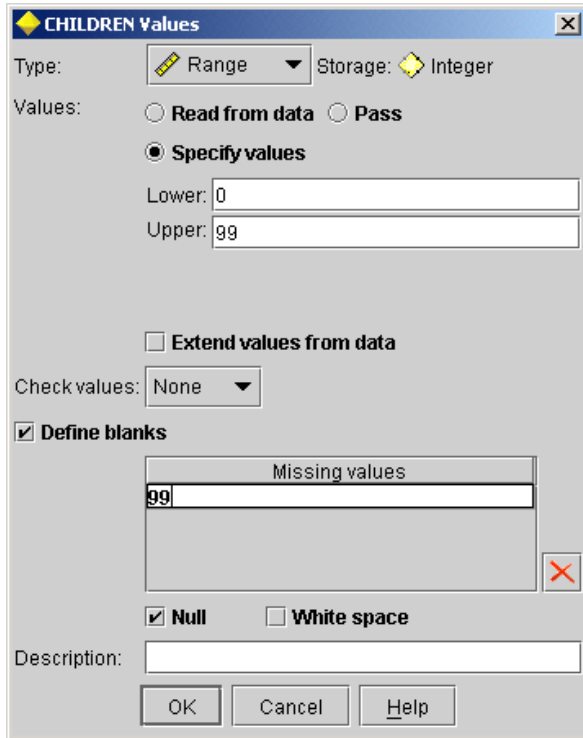
Figure 4.6 Types Tab: Missing Column



The Missing column controls whether some data values within a field will be defined as missing. We will declare the value 99 as missing value for CHILDREN.

- Click the cell in the **Missing** column and **CHILDREN** row
- Select **Specify** from the drop-down menu
- Click the **Define blanks** check box
- Click in the cell under **Missing values** and type **99**

Figure 4.7 Defining a Missing Value



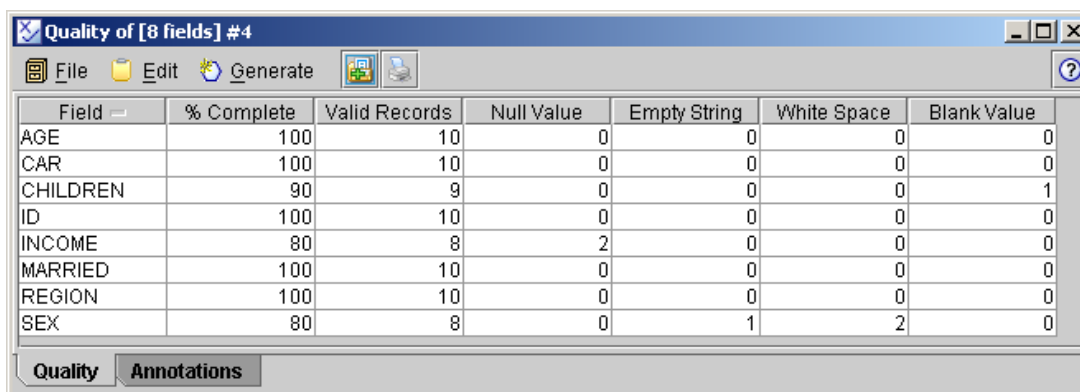
Notice that the *Null* check box is checked; so non-numeric values will be considered missing for the CHILDREN field. The *White space* check box, while not relevant for a numeric field, would serve to define white space (no visible characters, including empty strings) as missing for symbolic fields (for example, SEX).

- Click **OK** to return to the Var. File node dialog box
- Click **OK** to close the **Var. File** dialog box

Having defined 99 as missing for INCOME, we next ask the Quality node to check for it and other missing values.

- Edit the **Quality** node
- Click the check boxes for **White space**, **Blank value**, and **Empty string**
- Click the **Breakdown counts of records with invalid values** check box
- Click the **Execute** button in the **Quality** dialog box

Figure 4.8 Quality Output Window: Checking for All Missing Value Types



Field	% Complete	Valid Records	Null Value	Empty String	White Space	Blank Value
AGE	100	10	0	0	0	0
CAR	100	10	0	0	0	0
CHILDREN	90	9	0	0	0	1
ID	100	10	0	0	0	0
INCOME	80	8	2	0	0	0
MARRIED	100	10	0	0	0	0
REGION	100	10	0	0	0	0
SEX	80	8	0	1	2	0

The Quality report indicates that for INCOME, 80% of the records are complete and there are two records with null values (*Null Value* column).

SEX also has 80% of the records complete and two records are missing (record ID12702, has no value and record ID12703 has blank spaces). Note that both instances are counted in the *White Space* column (since white space includes empty strings), while the *Empty String* column contains a single instance. Since empty strings are included in the white space count, it's a matter of personal preference whether you want to check the *Empty string* option or not in the Quality dialog.

Clementine identifies one record with the user-missing value 99 for CHILDREN, as can be concluded from the 90% complete values and the one instance of Blank Value reported for CHILDREN.

Recall that we you can sort the report on any of the columns (by clicking the column header), which would make it easier to compare fields on any of the quality summaries when there are many fields.

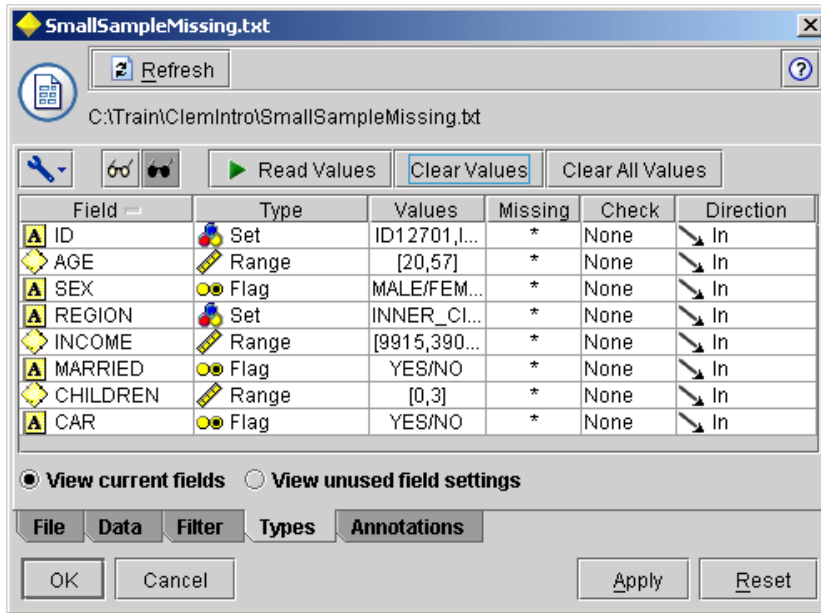
At this point, we have made a preliminary assessment of the Quality of the data. Although the Quality node reports on missing values, it does not define missing values. In other words, if we want Clementine to properly treat the white space, empty strings, and nulls as missing values in modeling and other nodes, we should declare them as missing in the Types tab. There is a very easy way to accomplish this.

- Edit the **Var. File** node (double-click the Type node) and click on the **Types** tab
- Click **Clear Values** button
- Right-click any field, and then click **Select All** on the context menu
- Right-click any field, and then click **Set Missing..On**

An asterisk appears in the Missing column for each field. This indicates that missing values have been declared. As we will see, by setting Missing On, all selected numeric fields will have the null value declared as missing and all selected symbolic fields will have white space and the null value declared as missing. Thus you can quickly declare missing values for many fields. Blank fields (user-defined missing values) need to be defined manually. To view the result:

- Click **Read Values** button

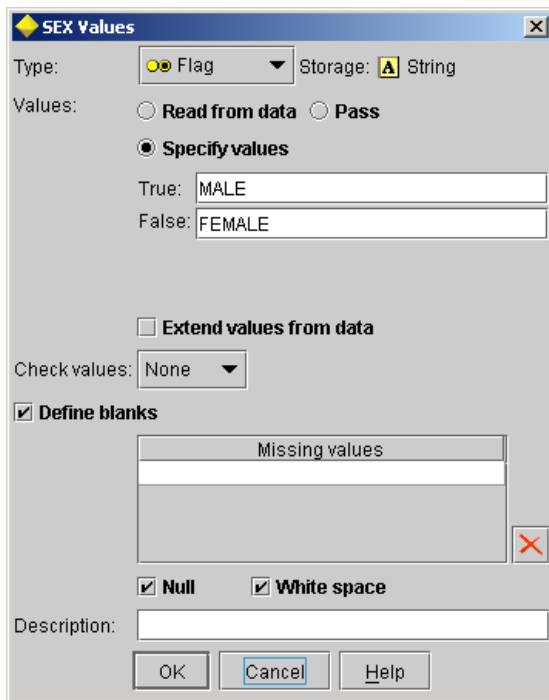
Figure 4.9 Types Tab after Data Read when Missing Values Declared



Compared to the original Types tab (Figure 4.6), there are differences for the SEX and CHILDREN fields. Since white space is considered missing for SEX, the type for SEX is now correctly identified as flag with values FEMALE and MALE. Also, the range for CHILDREN is now 0 through 3 because 99 is declared as a blank value. The point to remember is that declaration of missing values can influence the autotyping of symbolic fields and the range values for range fields, so it is advantageous to do this early in the data mining process. To verify that missing values were, in fact, declared for all fields:

Click the cell in the **Missing** column and **SEX** row, then click **Specify**

Figure 4.10 Missing Values for a Symbolic Field (SEX)

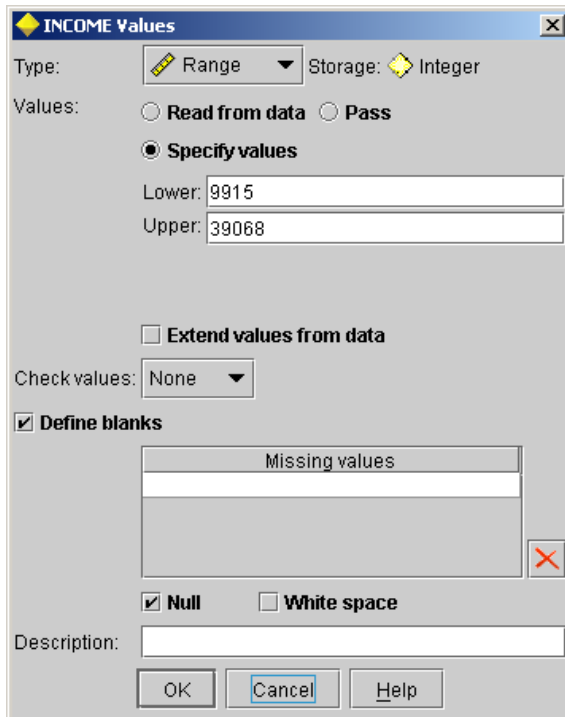


The *Define blanks* check box is checked, along with the *Null* and *White space* check boxes. Turning missing values on for a symbolic field automatically declares null values and white space as missing. It may seem odd that null values are declared as missing for a symbolic field, but some databases code empty symbolic fields as null, and so Null is checked to accommodate this.

Click **Cancel** button

Click the cell in the **Missing** column and **INCOME** row, then click **Specify**

Figure 4.11 Missing Values for a Numeric Field (INCOME)



After setting *Missing On*, the *Define blanks* and *Null* check boxes are checked for the selected numeric fields. In addition, user-defined missing values can be declared (as we did for CHILDREN).

In summary, if you want null values, white space, and empty strings to be treated as missing within Clementine, then selecting *Set Missing..On* from the right-click context menu is a convenient way to accomplish it.

Having seen the different types of missing values and how they are declared and reported, we now will look at other anomalies in the data.

Opening a Stream File

We will now switch to a larger and richer version of the data file. In addition to having more records (4117) and no missing values (blanks), it is a tab-delimited file. Rather than modifying the current Var. File node to read this file or building a new stream, we will open a previously saved stream. First we clear the current stream.

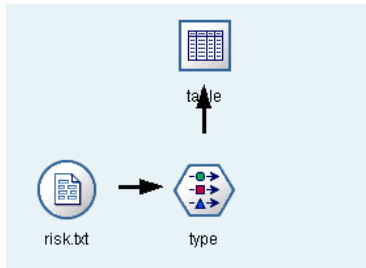
Click **Edit..Clear Stream**

Click **File..Open Stream**

Move to the **c:\Train\ClemIntro** directory

Double-click on **Riskdef.str**

Figure 4.12 Stream to Read Risk.txt Data



Notice that this stream contains a Type node, which is an alternative to using the Types tab in the source node. The Type node is not necessary here, but would be needed to properly type fields modified or added in the course of a Clementine stream.

Data Audit

A data set could contain 100% complete data but still have inaccurate entries or outliers. It is therefore important, before modeling takes place, to see how records are distributed for the fields in the data set. This can identify values which, on the surface, appear to be valid, but when compared to the rest of the data are either out of range or inappropriate.

Clementine provides a number of ways of examining the distribution of data fields. In this section we introduce the Data Audit node, which provides comprehensive information about each field.

When a data field is of a symbolic (flag or set) type it is of primary interest to see how many unique values there are and how the records are distributed among the categories of that field. For numeric fields there is usually interest in the distribution of the data values (histogram) and summary statistics (mean, minimum, maximum, and standard deviation). The Data Audit node provides such displays and summaries for fields in the data file. It thus provides much useful information about the data.

Execute the **Table** node (right-click Table node, then click Execute)

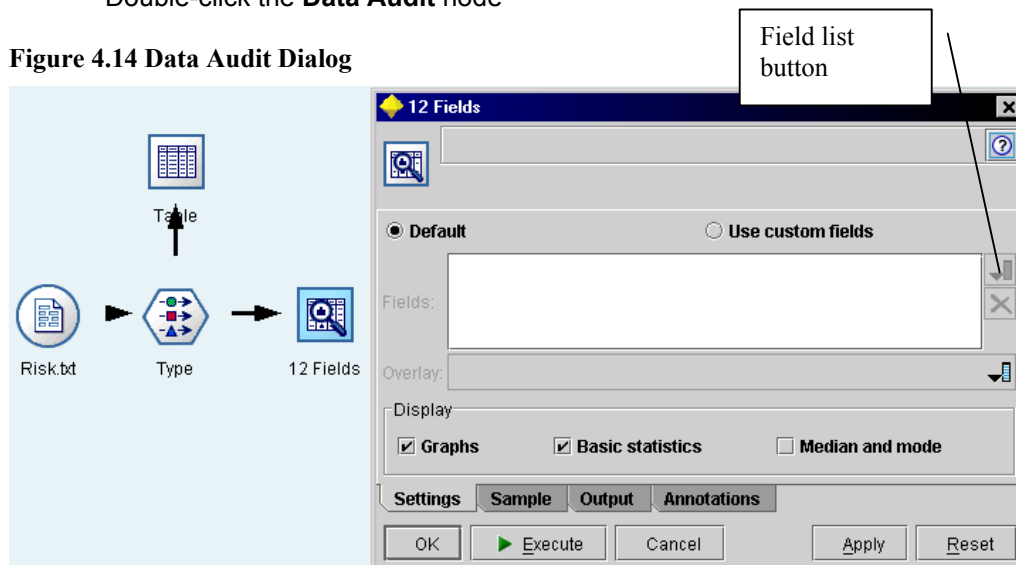
Figure 4.13 Data from Risk.txt

	ID	AGE	INCOME	GENDER	MARITAL	NUMKIDS	NUMCARDS	HOWPAID	MORTGAGE	S
1	100756	44	59944	m	married	1	2	monthly	y	
2	100668	35	59692	m	married	1	1	monthly	y	
3	100418	34	59508	m	married	1	1	monthly	y	
4	100416	34	59463	m	married	0	2	monthly	y	
5	100590	39	59393	f	married	0	2	monthly	y	
6	100657	41	59276	m	married	1	2	monthly	y	
7	100702	42	59201	m	married	0	1	monthly	y	
8	100319	31	59193	f	married	1	2	monthly	y	
9	100666	28	59179	m	married	1	1	monthly	y	
10	100389	30	59036	m	married	1	1	monthly	y	
11	100758	38	58914	m	married	0	1	monthly	y	
12	100695	36	58878	f	married	1	1	monthly	y	
13	100698	42	58785	f	married	0	2	monthly	y	
14	100769	44	58529	m	married	0	1	monthly	y	
15	100376	33	58505	f	married	0	2	monthly	y	

We have several symbolic fields (GENDER, MARITAL, etc.) and several numeric fields (AGE, INCOME, etc.) and will use the Data Audit node to explore the file.

Click the **Data Audit** node in the **Output** palette
 Click in the Stream Canvas to the **right** of the **Type** node
 Connect the **Type** node to the **Data Audit** node
 Double-click the **Data Audit** node

Figure 4.14 Data Audit Dialog

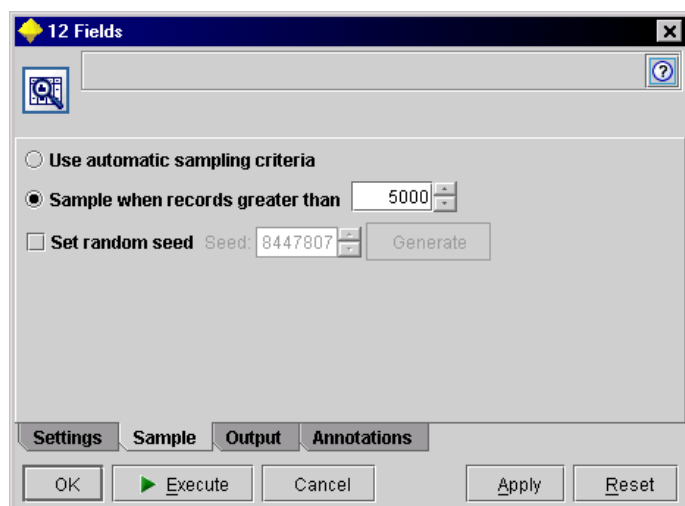


By default, all fields are included in the data audit analysis. However, the *Field list* button can be used to select specific fields for analysis when the *Use custom fields* option is chosen. If custom fields are selected, the *Overlay* field list allows the distribution of a symbolic field to appear over the distribution of the fields selected in the *Field list*. For example, a distribution of marital status with credit risk status as an overlay would yield a graph showing the number of records within each category of marital status broken down by credit risk category.

The Display group controls whether graphs are created and which summary statistics will be calculated. Since median and mode statistics require more computational resources than the basic statistics, they constitute a separate option.

Click the **Sample** tab
 Click **Sample when records greater than** option button
 Enter **5000** in the records box (or use the spin control)

Figure 4.15 Data Audit: Sample Tab



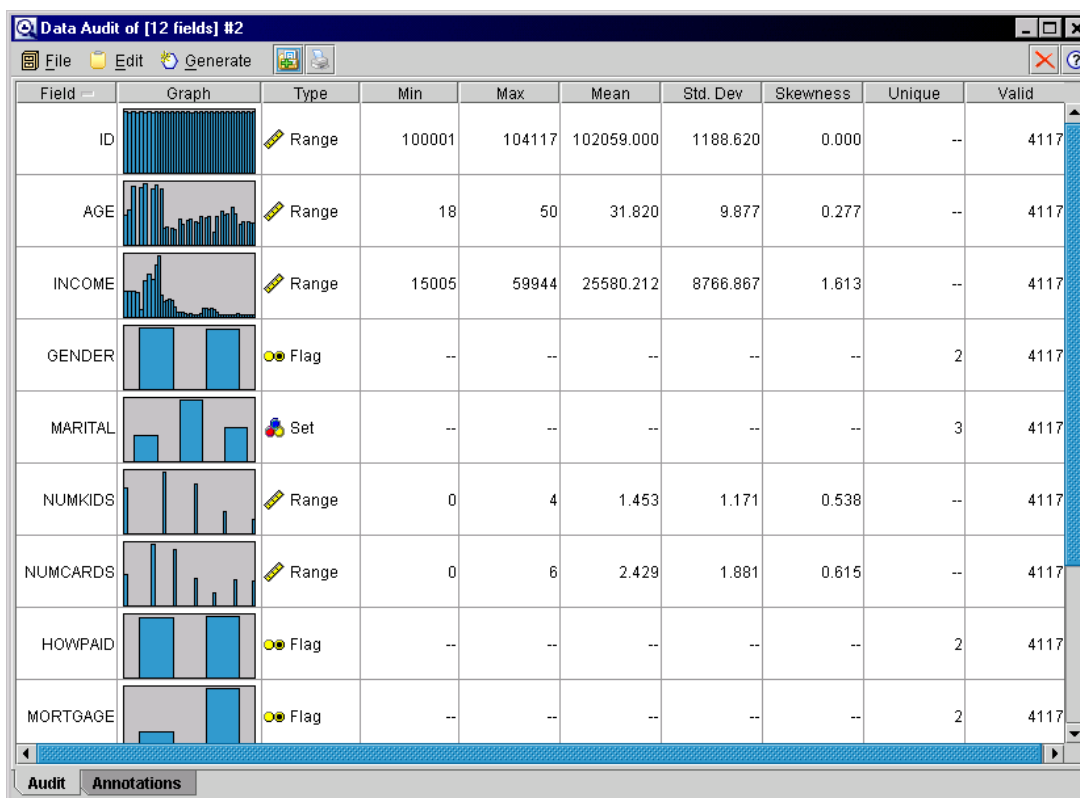
The Sample tab allows you to specify when data will be sampled for the audit. For large data sets, sampling will reduce processing time while producing an initial assessment of the data. Sampling applies only to graphs and the median statistic, which is not calculated by default. When sampling is in effect, the column label for graphs or statistics based on the sample will be prefixed with “Sample” (for example, “Sample Graph”). When the *Use automatic sampling criteria* option is chosen, 2000 records will be randomly sampled if there are less than 250 fields analyzed (otherwise 1000 records will be sampled). The *Sample when records greater than* option will run the audit on a random sample of records when the number of records exceeds the specified value. To turn off sampling altogether, choose the *Sample when records greater than* option and then enter a value that exceeds the total number of records. This is what we did (since there are 4117 records, see Figure 4.13).

When many fields are involved, sampling can reduce the processing requirements for the initial data audit.

The *Set random seed* option allows you to specify a starting seed value for the random number algorithm used in sampling. This allows you to reproduce your data audit results when a Data Audit node with sampling is run later (by default, a seed is chosen based on time of day, so a different sample would be drawn each time the Data Audit node is executed).

Click **Execute** button

Figure 4.16 Data Audit Output



Each row of the Data Audit output represents a field and the columns contain graphs, type information, and statistical summaries. Under default settings in the Data Audit node, every field will have a graph, type information, and a summary of the number of records with valid values for that field (*Valid* column). For fields of Range type, the graph in the *Graph* column is a histogram, while fields of Flag or Set type are graphed using bar charts (in Clementine they are called distribution charts).

The summary statistics for a field of Range type are minimum (*Min*), maximum (*Max*), mean, standard deviation (*Std. Dev*), skewness, and number of valid values (*Valid*). Skewness is a measure of symmetry in a distribution; a perfectly symmetric distribution would have a skewness value of 0, while distributions with long tails to the right (see INCOME) would have positive skewness. The AGE field has an observed range of 18 to 50 with a mean of 31.82, based on 4,117 records. Interest would be attracted by unexpectedly low or high values or odd distributions. For example, since this is a credit risk data file, an AGE value of 11 would suggest a data error. Similarly, a concentration of high-income values would suggest data errors or a sample not representative of the population at large.

For fields of Flag or Set type, in addition to a distribution (bar) chart, the *Unique* column displays the number of unique values found for the field in the data file (or sample). As expected, GENDER has two unique values and the distribution plot suggests the file has roughly equal numbers of males and females.

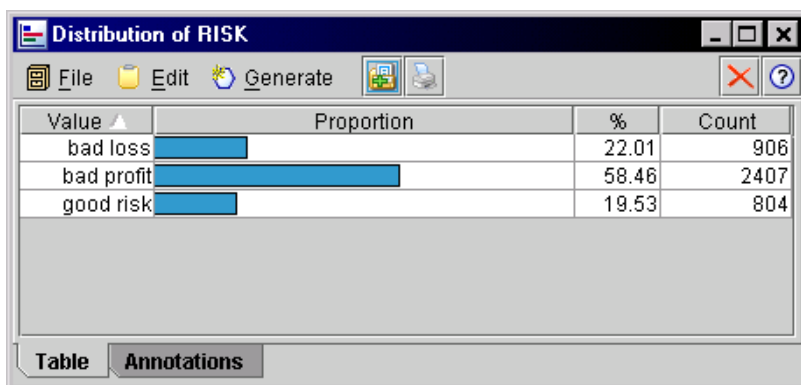
Examine the graphs and summaries for the other fields appearing in the Data Audit output window.

The graphs in the Data Audit output clearly display the general distribution of the fields, but are too small to present the scale and category identifiers. However, more detailed versions of the graphs can be easily produced.

Distribution Plots

Double click on the **graph** for **RISK** in the Data Audit output window

Figure 4.17 Distribution (Bar) Graph for Risk



Double-clicking on a graph in the Data Audit output window creates the graph (distribution plot or histogram) in its own graph window. For distribution plots, category labels appear along with count and percent summaries. This graph is added as a new object in the Outputs manager.

We examine the distribution of the field we are going to try to model in future chapters: RISK. This field contains three categories: *good*, *bad profit* and *bad loss*, which represent that in the view of a credit card company an individual may be a good credit risk, a bad credit risk but be profitable, or a bad credit risk and cause a loss.

The largest group within the data contains 2407 individuals or 58.46% of the sample and is composed of those who are considered bad credit risks but profitable to the organization. The other two groups, bad loss and good risk, are roughly proportional, with 22.01% and 19.53% of the records, respectively.

The Distribution File menu allows you to Save, Print or Close the window. The Generate menu can create different nodes: including a Select node (used to select records for analysis) and a Balance node that can either boost the size of smaller groups or reduce the size of larger groups (preferable), which can be useful

when modeling. The reader is referred to the *Clementine User's Guide* or the *Advanced Modeling with Clementine* training course for more detail on data balancing. The Edit node allows you to combine groups in the plot.

A detailed distribution plot can be viewed for every distribution graph in the Data Audit output and helps understand the data, as well as identify out-of-range or inappropriate values. In addition, a distribution plot for a single symbolic field can be created from the Distribution node located in the Graphs palette.

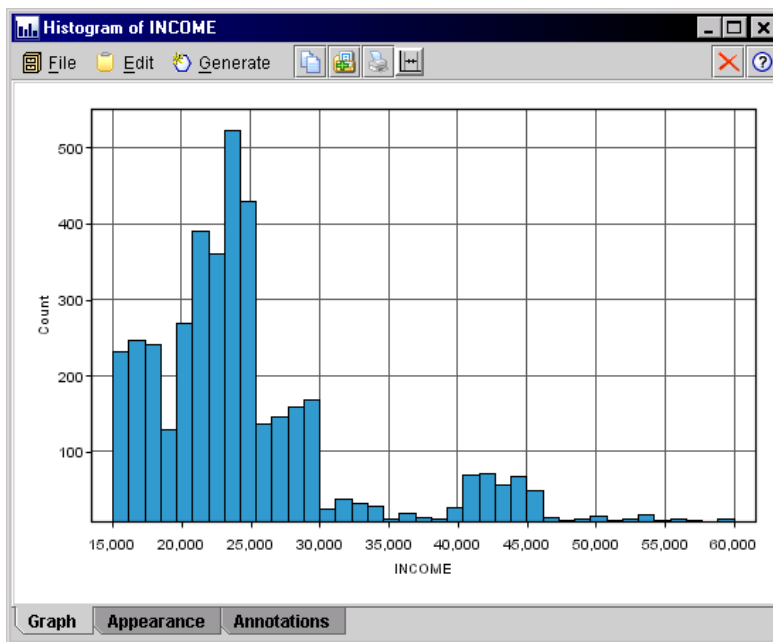
Click **File..Close** to close the Distribution graph window

Histograms

Double click on the **graph** for **INCOME** in the Data Audit output window

The Histogram node shows the frequency of occurrence of values for numeric fields. In a histogram, the range of data values is split into bands (buckets) and bars representing the number of records falling into each band are displayed.

Figure 4.18 Histogram of INCOME



Income values range between approximately 15,000 and 60,000 with a large proportion of cases between 20,000 and 25,000. The distribution is also concentrated at the lower end of the income scale. This analysis can be repeated for all numeric fields in the data.

When the Data Audit node generates histograms, the range displayed and binning of data values are determined automatically. You can control these properties for individual histograms produced by the Histogram node using its Options tab.

Notes

The Data Audit node automatically produces thumbnail distribution (bar) charts and histograms for all fields included in the analysis, which is a great convenience. However, you have greater control over chart options if the graph is individually produced using the Distribution or Histogram node.

When sampling is in effect in the Data Audit node, the thumbnail graphs are based on the sample. However, if you double-click on a thumbnail graph to view a full-sized distribution plot or histogram, these plots will be based on the full sample.

The Statistics node in the Output palette can produce a more extensive set of summary statistics for numeric fields than the Data Audit node (sums, variances, standard error of means, correlations). It does not produce summaries for Flag or Set type fields and does not present graphs. It will be seen in the *Looking for Relationships in Data* chapter (Chapter 6).

Summary

In this chapter you have been given an introduction to a number of methods that can be used to explore the quality of your data.

You should now be able to:

- Use the Quality node to assess the completeness of data
- Use the Types tab to define user-missing values
- Visualize the distribution of fields and examine their summary statistics using the Data Audit node
- Examine Data Audit distribution (bar) charts and histograms in more detail

Chapter 5

Introduction to Data Manipulation

Overview

- Introduce the Select node
- Introduce several field operations: Filter, Field Reorder, Derive, and Reclassify
- See how to automatically generate Field and Record operation nodes

Objectives

This session aims to introduce some of the data manipulation techniques available in Clementine. We will show how these techniques can be used to clean and refine data for mining.

Data

In this chapter we will use the text data file *Risk.txt*. The data file contains information concerning the credit rating and financial position of 4117 individuals, along with basic demographic information, such as marital status and gender. A small data file containing missing values, *SmallSampleMissing.txt*, is also used. This file has one record per account held by customers of a financial organization. It contains demographic details on the customer, such as income, gender, and marital status.

Introduction

We introduced a number of ways to check the quality of data. Once this task has been completed it is often necessary to manipulate the data further. For example, you may be interested in creating new fields in the data that are combinations of existing fields.

Such techniques are available within Clementine and can be found in either the Record Ops palette (containing tools for manipulating records) or Field Ops palette (containing tools for manipulating fields).

In this chapter we will introduce how the Select node can be used for selecting a group of records that conform to some criteria. We will also introduce several field operation nodes: the Filter node, which removes unwanted fields from analysis; the Reorder node, which reorders fields in the data stream and dialogs; the Derive node, used to create new fields in the data stream; and the Reclassify node, which is used to change the coding or collapse categories for symbolic fields.

We will demonstrate manual creation by placing the relevant node on the Stream Canvas and using the CLEM language. We will also demonstrate how the Derive, Filter and Select nodes can be automatically created using the Generate menu available in the output windows of nodes introduced in the previous chapters. Before discussing the nodes themselves we will introduce the CLEM language.

A Brief Introduction to the CLEM Language

Clementine Language for Expression Manipulation, or CLEM, is a language for reasoning about and manipulating the data that flow along streams. CLEM is used in Derive, Select, Filter, Balance and Report nodes, and, among other things, permits you to:

- Compare and evaluate conditions
- Derive new fields
- Insert data from records into reports

For a detailed introduction to the CLEM language the reader is referred to the *Clementine User's Guide*. In this section we will introduce the basic concepts and commonly used functions available in CLEM.

CLEM expressions are constructed from values, fields, operators, and functions.

Values can be:

Integers - e.g. 3, 50, 10000

Real Numbers – e.g. 4.51, 0.0, -0.0032

Strings (within single quotes) - e.g. 'male', 'married' etc.

Field names can be referred to:

Directly - e.g. risk, income etc.

Within quotes if it is a special field name (usually produced by Clementine when machine learning) – e.g. '\$R-risk', '\$N-profit'

Operators commonly used are given in the table below.

Table 5.1 Commonly Used Operators in CLEM

+	Add	>	greater than
-	Subtract	<	less than
*	Multiply	>=	greater than or equals
/	Divide	<=	less than or equals
**	Raise to the power	=	equal to
div	return the quotient	/=	not equal to
rem	return the remainder on dividing	mod	return the modulus
><	Joins string expressions together (concatenation)		

A few of the commonly-used functions are given in table below:

Table 5.2 Commonly Used Functions in CLEM

round	rounds to the nearest integer away from 0.5
abs	gives the absolute value
sqrt	Takes the square root
log	Natural logarithm
exp	Raises e to the power of
sin/cos	Trigonometric functions
min / max	Returns the minimum or maximum of its arguments
substring(start, length, string)	Returns part of a string, from start for a specified length

For example:

Sqrt (abs (famincome - income))	will return a number equal to the square root of the absolute difference between the fields income and famincome (family income).
'Mr ' >< surname	will return a string consisting of 'Mr Name', where 'Name' represents the value of the field called surname
Age >= 65	will return T (True) if the age field is greater than or equal to 65 and F (False) if not.

Note about Case Sensitivity

CLEM expressions are case sensitive and will return either a result, or evaluate to true or false.

Record Operations and the Select Node

Clementine contains a number of data manipulation techniques that perform operations on records. These include sorting, selecting, merging, appending and balancing, and are found in the Record Ops palette.

Now that we have introduced the CLEM language, we are able to examine in detail one of the most useful of these operations, the Select node, which allows you to either select or eliminate a group of records based on a specified condition.

The Select node can be manually created or automatically derived. Here we manually create a Select records node. In this example we will work on the stream opened in Chapter 4. If you still have your work from Chapter 4 in the Stream Canvas, we will clear it since we will not need most of the nodes.

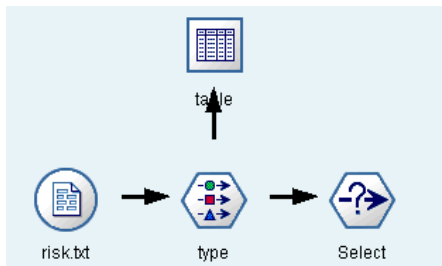
If the Stream Canvas is not empty, choose **File..Close Stream** (and click No if asked to save)

Click **File..Open Stream**, move to the **c:\Train\ClemIntro** directory and double-click **Riskdef.str**

Place a **Select** node from the Record Ops palette to the right of the **Type** node

Connect the **Type** node to the **Select** node

Figure 5.1 Select Node Added to Stream



To insert the condition for selection or deletion we need to first edit the Select node.

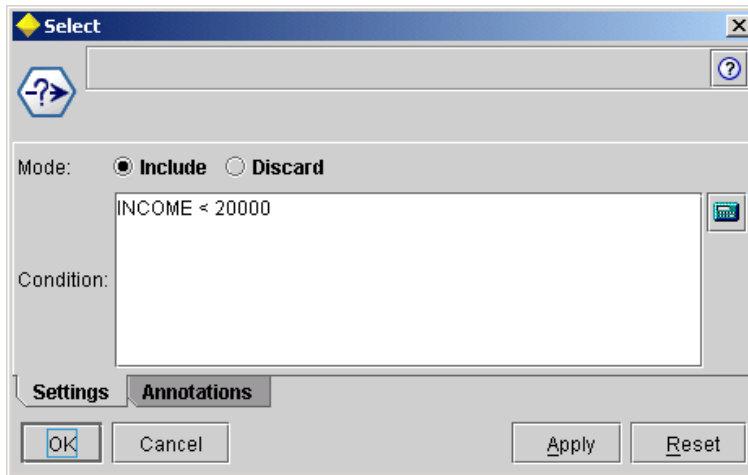
Right-click the **Select** node, then click **Edit**

The CLEM expression that depicts the required condition is entered in the Condition text box. The Mode option allows the user to choose whether to select (*Include*) or delete (*Discard*) records that satisfy the condition.

In this example we are interested in selecting records for which INCOME is below 20,000 (in British pounds), so that we can then examine the distribution of the RISK field to see if the proportions are different for this subgroup of individuals.

Type **INCOME < 20000** in the **Condition** text box (remember that CLEM is case sensitive!)
Check that **Mode:** is set to **Include**

Figure 5.2 Select Dialog to Select Records with INCOME < 20000



Click **OK** to return to the Stream Canvas

At this point, nodes downstream of the Select node will analyze only records for which income is below 20,000.

To ensure that the select statement is working, it is a good idea to connect a Table node to the Select node and execute the stream. Only those records that meet the condition will appear in the table. In this instance the Data Audit or Statistics node, which would display the minimum value, could be used as well.

Place a **Table** node from the Output palette to the right of the **Select** node
Connect the **Select** node to the new **Table** node
Right-click the new **Table** node, then click **Execute**

Figure 5.3 Table Report of Records Where Income is Less than 20,000

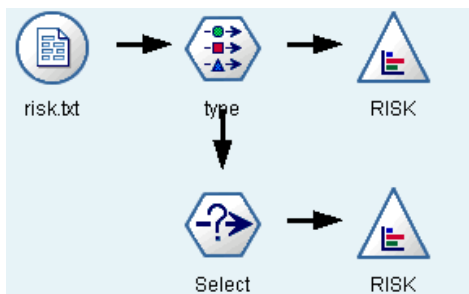
	ID	AGE	INCOME	GENDER	MARITAL	NUMKIDS	NUMCARDS	HOWPAID
1	102428	23	19990	f	single	0	1	monthly
2	103352	29	19950	m	divsepwid	4	6	weekly
3	102870	40	19945	f	divsepwid	2	6	weekly
4	102129	21	19876	f	single	0	2	weekly
5	103601	49	19863	f	divsepwid	2	5	weekly
6	100808	18	19847	f	married	2	0	weekly
7	102589	36	19811	f	single	1	3	monthly
8	102279	22	19809	m	single	0	3	monthly
9	103020	42	19779	f	divsepwid	2	5	weekly
10	103680	34	19767	m	divsepwid	4	6	weekly
11	103775	41	19705	m	married	2	4	monthly
12	103027	42	19697	f	divsepwid	4	5	weekly
13	103838	43	19696	f	married	3	3	weekly
14	103351	46	19688	m	divsepwid	4	6	weekly
15	101714	28	19683	m	married	2	2	monthly
16	101276	21	19640	f	married	1	1	weekly
17	103191	27	19624	m	divsepwid	4	6	weekly
18	101572	23	19617	f	married	1	2	monthly
19	103675	50	19606	m	divsepwid	2	6	weekly
20	103836	43	19599	f	married	2	2	monthly

The resulting table contains only 869 records— those with income below 20,000, so the Select node appears to have been successful.

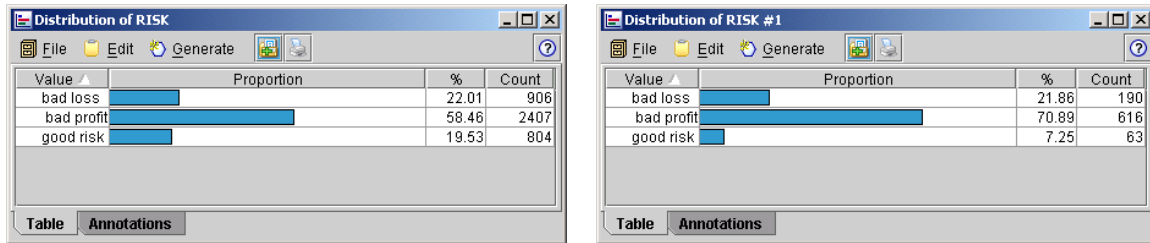
Next, using a Distribution node, we will compare the distribution of risk for the entire sample to the subgroup with income under 20,000.

- Click **File..Close** to close the Table window
- Delete the **Table** node(s)
- Drag the **Select** node below the **Type** node
- Place a **Distribution** node from the Graphs palette to the right of the **Type** node
- Connect the **Type** node to the **Distribution** node
- Double-click the **Distribution** node
- Click the field list button and select **RISK**
- Click **OK**
- Copy** the **Distribution** node and **Paste** it to the right of the **Select** node
- Connect the **Select** node to the pasted **Distribution** node

Figure 5.4 Comparing Credit Risk for the Complete Sample to a Selected Group



Execute the two **Distribution** nodes (right-click on each node, then click **Execute**)

Figure 5.5 Distribution of Risk for Complete Sample and for those Earning under 20,000

The distribution plots indicate that in comparison to the complete sample, the subgroup of those who earn below 20,000 contains a smaller proportion of good credit risks. This may lead us to try to predict credit risk using income, since there appears to be an association between the two fields.

We will now go on to introduce an operation that has a similar function to the Select node but works on fields and not records: the Filter node.

- Close the two **Distribution** graph windows
- Delete the **Distribution** nodes and the **Select** node from the stream canvas

Field Operations and the Filter Node

As mentioned earlier, Clementine has a number of nodes that allow you to manipulate fields within the data set. In this section we introduce the Filter node that can rename fields and remove unwanted fields from the data stream. If these functions need to be performed when the data are first read, the filter tab of any source node can be used (see Chapter 3).

In the last chapter we checked the distribution of a few of the fields in our data set. When data mining, two potential problems may occur within a field:

- A large proportion of missing records
- All records having the same value (invariant)

The Filter node (or Filter tab of a source node) allows data to pass through it and has two main functions:

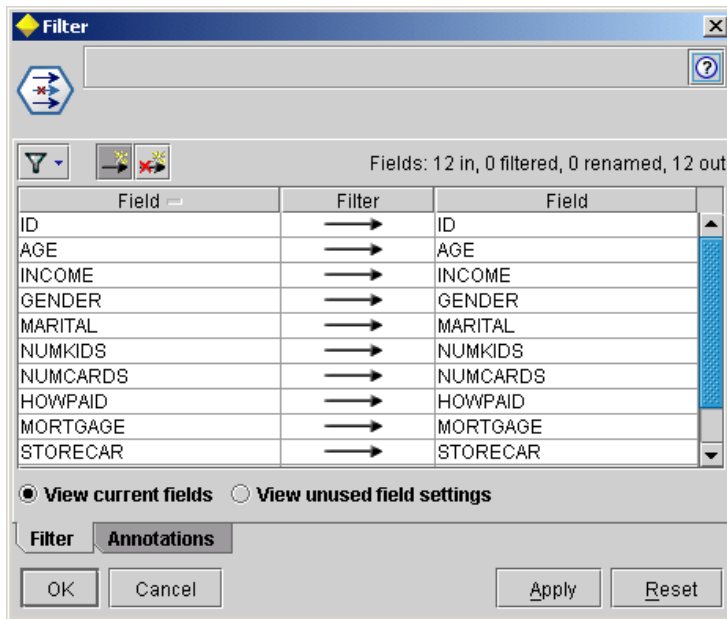
- To filter out (discard) unwanted fields
- To rename fields

- Place a **Filter** node from the **Field Ops** palette to the right of the **Type** node
- Connect the **Type** node to the **Filter** node

Figure 5.6 Stream with a Filter Node

- Right-click on the **Filter** node, and then click **Edit**

Figure 5.7 Filter Node Dialog



Text at the top of the dialog indicates the number of fields entering the Filter node, the number of fields filtered, the number of fields renamed, and the number of fields leaving it.

The left column lists the field names as the data stream enters the Filter node. The right column shows the field names as the data stream leaves the Filter node. By default the lists are the same.

To Change the Name of a Field

To demonstrate changing a field name, we will change the name STORECAR to STORECARDS (the number of store credit cards).

Click the text **STORECAR** in the right column (right of the arrow)

Type the new name **STORECARDS** in the text box (replace the original name, or simply append DS to it)

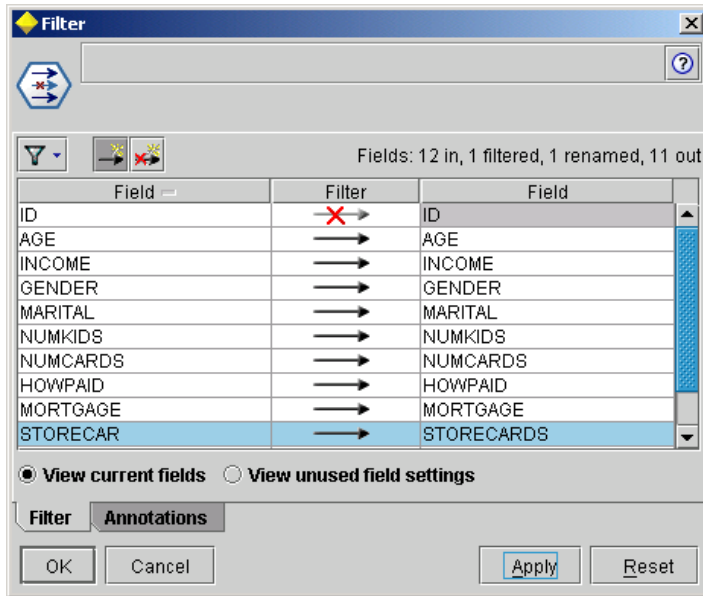
The new name should appear in the right column (not shown).

To Filter Out Fields

To demonstrate how to remove fields from the data stream, we will filter out the ID field. This involves clicking on the arrow connecting the input (left column) to the output (right column) in the Filter node dialog.

Click on the arrow next to **ID**

Figure 5.8 ID Removed from Stream and STORECAR Renamed



To reinstate a previously filtered field, click on the crossed arrow. The original arrow will be displayed and output field name will be reinstated.

Multiple fields can be removed. Simply click and drag from the arrow for the first field to be omitted to the arrow for the last field, highlighting the fields to be omitted and then click anywhere on the highlighted area (or right-click, then click Remove the selected fields). To reinstate multiple omitted fields, simply repeat the process.

The Filter options menu provides a set of options that are useful when working with a large number of fields.

Click the **Filter options** button 

Figure 5.9 Filter Options Menu



Filter options include removing or including all fields, toggling the remove/include settings for all fields, removing or renaming duplicate field names. These latter options are useful when working with database files with many fields, since they provide an easy way of setting the filter options for all fields.

Press the **Esc** key to close the Filter Options menu

The quickest way to check that the Filter node is doing its job is to connect it to a Table node and view the output. We will view this table shortly.

Click **OK** to close the **Filter** dialog box

Field Reordering

Another useful field operation is to reorder the fields, which would effect their ordering in dialog boxes and data streams. For example, you might want a specific field ordering in a table to better compare the outcome with predictions from different models, or it might be easier to locate field names in dialogs if they were alphabetically ordered. The Field Reorder node will reorder fields downstream of the node and has several options for field reordering, including custom ordering. To illustrate:


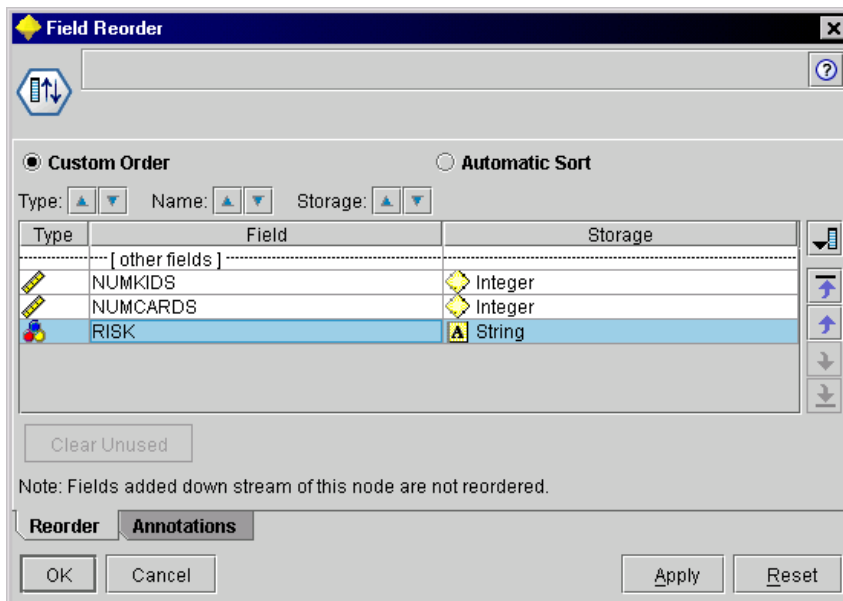
- Place a **Field Reorder** node from the Field Ops. Palette to the right of the **Filter** node
- Connect the **Filter** node to the **Field Reorder** node
- Right-click the **Field Reorder** node, and then click **Edit**
- Click the Field list  button
- Select (Ctrl-click) **NUMKIDS**, **NUMCARDS**, and **RISK** in the Select Fields dialog
- Click **OK**
- Click **RISK** in the Field Reorder dialog

Figure 5.10 Field Reorder Node Dialog



The field order shown in the Field Reorder dialog controls field ordering downstream. The [other fields] item represents fields not explicitly listed in the Field Reorder dialog. The current ordering would have NUMKIDS, NUMCARDS and RISK appearing as the last three fields, preceded by the other fields in their original order.

You can change the order of selected fields using the buttons. Selected fields or the [other fields] item can be moved up or down one position or moved to the top or bottom of the list. In addition, when *Custom Order* is selected, the fields in the list can be sorted in ascending or descending order by *Type*, *Name*, or *Storage*. When any of these sorts are performed, the [other fields] item moves to the bottom of the list.

If the *Automatic Sort* option is chosen, then all fields can be sorted by *Type*, *Name*, or *Storage*.

- Click the Move selected fields to the top button 

This will reorder the fields so that RISK appears first, followed by all fields in their original order except NUMKIDS and NUMCARDS, which appear last.

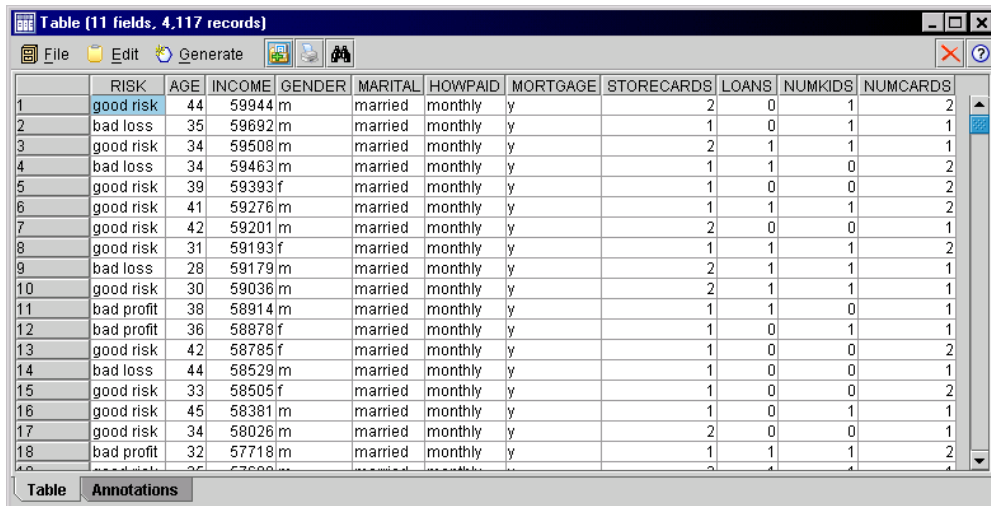
Click **OK**

Place a **Table** node from the Output palette to the right of the **Field Reorder** node

Connect the **Field Reorder** node to the **Table** node

Right-click the new **Table** node, then click **Execute**

Figure 5.11 Table Following the Filter and Field Reorder Operations



	RISK	AGE	INCOME	GENDER	MARITAL	HOWPAID	MORTGAGE	STORECARDS	LOANS	NUMKIDS	NUMCARDS	
1	good risk	44	59944	m	married	monthly	y		2	0	1	2
2	bad loss	35	59692	m	married	monthly	y		1	0	1	1
3	good risk	34	59508	m	married	monthly	y		2	1	1	1
4	bad loss	34	59463	m	married	monthly	y		1	1	0	2
5	good risk	39	59393	f	married	monthly	y		1	0	0	2
6	good risk	41	59276	m	married	monthly	y		1	1	1	2
7	good risk	42	59201	m	married	monthly	y		2	0	0	1
8	good risk	31	59193	f	married	monthly	y		1	1	1	2
9	bad loss	28	59179	m	married	monthly	y		2	1	1	1
10	good risk	30	59036	m	married	monthly	y		2	1	1	1
11	bad profit	38	58914	m	married	monthly	y		1	1	0	1
12	bad profit	36	58878	f	married	monthly	y		1	0	1	1
13	good risk	42	58795	f	married	monthly	y		1	0	0	2
14	bad loss	44	58529	m	married	monthly	y		1	0	0	1
15	good risk	33	58505	f	married	monthly	y		1	0	0	2
16	good risk	45	58381	m	married	monthly	y		1	0	1	1
17	good risk	34	58026	m	married	monthly	y		2	0	0	1
18	bad profit	32	57718	m	married	monthly	y		1	1	1	2

The ID field is no longer present and the STORECAR field has been renamed. RISK is now the first field while NUMKIDS and NUMCARDS are in the last two positions.

We have examined the Filter node as a method of renaming and discarding fields within the data and have seen how to use the Field Reorder node. It is often the case, however, that the values themselves within the fields need to be altered, or new fields created as combinations of existing fields. In the next section we will introduce the Derive node as a method of performing such data manipulations.

Click **File..Close** to close the Table report window

Right-click the **Field Reorder** node, then click **Delete**

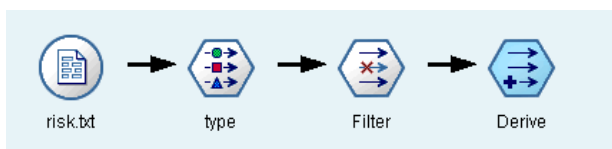
Right-click the new **Table** node, then click **Delete**

The Derive Node

In order to make full use of the modeling techniques available in Clementine, it may be necessary to modify data values or create new fields as functions of others. The Derive node calculates a new value, based on a CLEM expression for every record passed through it. To enter the CLEM expression and the new field name (a name is automatically assigned) you need to edit the Derive node.

Click **Insert..Field Ops..Derive**

Figure 5.12 Adding a Derive Node



You can use the Insert menu to add a node to the Stream Canvas. Note that the node is automatically connected to the stream.

Right-click the **Derive** node, then click **Edit**

Figure 5.13 Derive Node Dialog



The new field name is typed in the *Derive field* text box. Remember that Clementine is case sensitive with respect to field names.

The Derive node offers six different manipulations for a new field. Clicking the *Derive as* drop-down list will reveal these options:

Formula	The new field is the result of an arbitrary CLEM expression.
Flag	The resulting field will have a True or False response (flag), reflecting a specified expression.
Set	The new field will have values assigned from members of a specified set.
Count	The new field is based on the number of times a specified condition is true.
State	The new field's value represents one of two states. Switching between these states is triggered by specified conditions.
Conditional	The new field is the result of one of two expressions, depending on the value of a condition.

The Count and State derive types are most often used with time series or sequence data and are discussed in the *Data Manipulation with Clementine* training course.

Once the type of manipulation is chosen, the dialog box changes appropriately. The type of the field to be derived can explicitly be set using the Field type option. For the moment, we will leave it to its default value.

Single Versus Multiple Derive Mode

A Derive node is usually used to calculate a single new field, which is why the *Single* Derive Mode option button is selected by default. For instances in which you need to calculate multiple new fields using the

same operations applied to a series of fields, the *Multiple* Derive Mode option is available. In this mode you select the fields to which the operations will be applied, indicate how the new fields are to be named (by adding a user-specified prefix or suffix to the original field names), and specify the data transformation operations. To accommodate this, the Derive node dialog will expand when the *Multiple* Mode option is selected. In this course, we demonstrate the Single Derive mode. Some examples of multiple Derive mode are: applying a natural log transformation to a set of fields that will be used in a neural net; creating a series of flag fields coded as F (0 or negative balance) or T (positive balance) based on a set of financial account fields.

Derive Type Formula

For this example we will calculate a composite measure of potential debt, which is equal to the sum of the number of credit cards (NUMCARDS), number of store cards (STORECARDS— after renaming within the Filter node) and number of loans (LOANS) for each record.

- Select **Formula** from the **Derive as:** drop-down list
- Type **SUM DEBT** in the **Derive field:** text box (replacing the current name)

You can type the equation into the Formula text box, but it is easier to invoke the Expression Builder in which you can create an expression by clicking the operation buttons and selecting fields and functions from list boxes. We will demonstrate use of the Expression Builder.


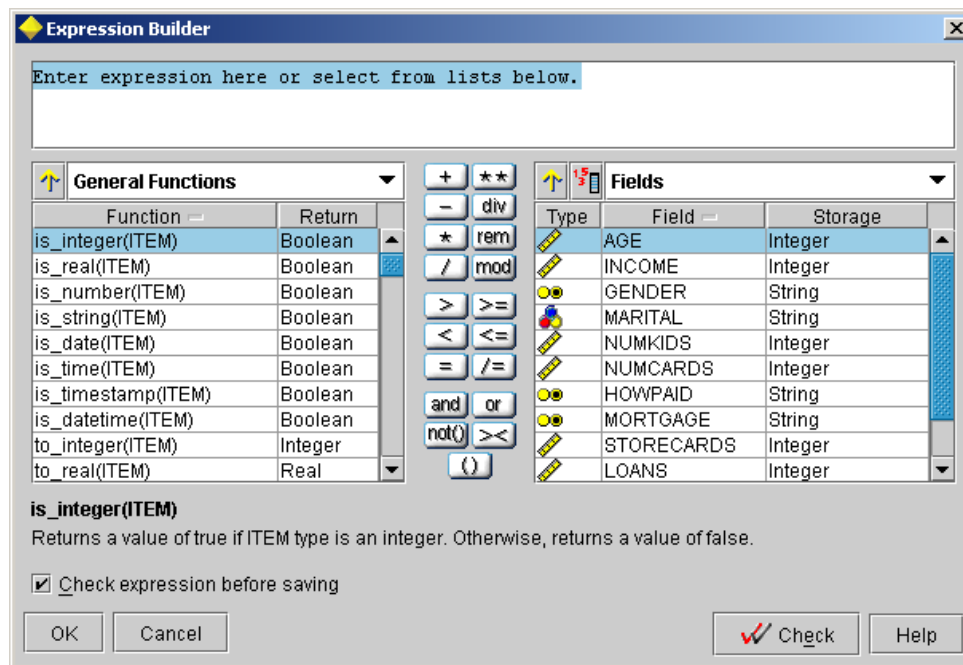

Click the Expression Builder  button

Figure 5.14 Expression Builder Dialog



The expression is built in the large text box. Operations (addition, subtraction, etc.) can be pasted into the expression text box by clicking the corresponding buttons. The Function list contains functions available in Clementine.

By default, a general list of functions appears, but you can use the drop-down list above the Function list box to display functions of a certain type (for example, Date and Time functions, Numeric functions,

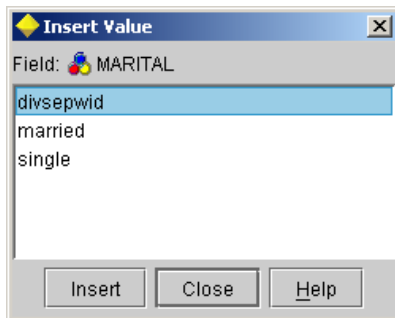
Logical functions). Similarly, by default, all fields in the stream are listed in the Fields list box and can be pasted into the expression by clicking the Insert button  after the field is selected. The Fields list box can display all fields, recently used fields, parameters, or globals. The latter two categories are discussed in the *Clementine User's Guide*.

In addition, this dialog can display field information for a selected field, which, in turn, can be pasted into the expression. To illustrate:

Click **MARITAL** in the **Fields** list box

Click the Select from existing field values button 


Figure 5.15 Field Values for MARITAL in the Insert Value Dialog




Since MARITAL is a field of type set, the Insert Value dialog contains a list of its set values and these can be pasted into the expression using the Insert button. Depending on the field's type, different information will display— similar to what we saw when examining the Types tab. In addition to allowing you to paste expressions, the Expression Builder will check the validity of the expression. Such features provide a powerful tool with which to construct expressions.


Now to build the equation:

Click **Close** button to close the Insert Value dialog

Click **NUMCARDS** in the Fields list box, then click the **Insert** button 

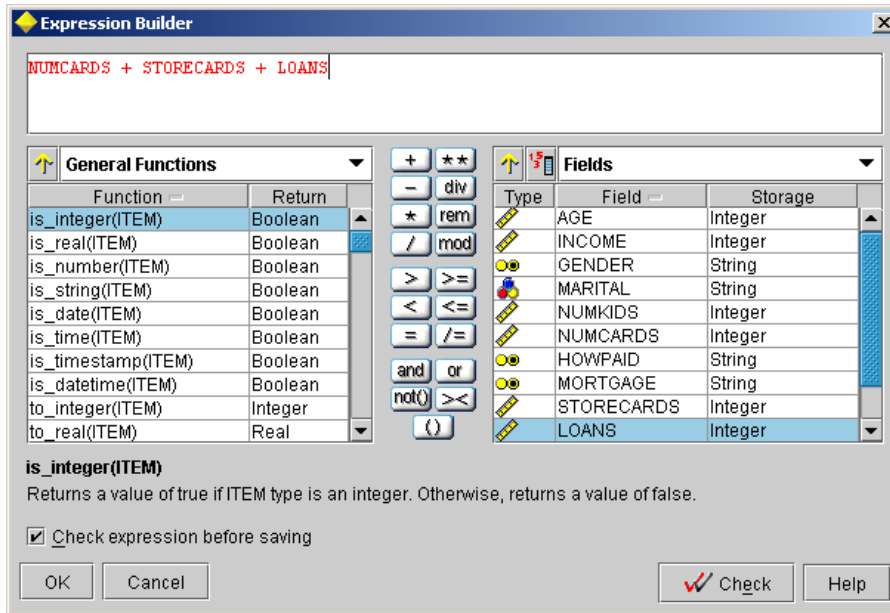
Click the plus button 

Click **STORECARDS** in the Fields list box, then click the **Insert** button 

Click the plus button 

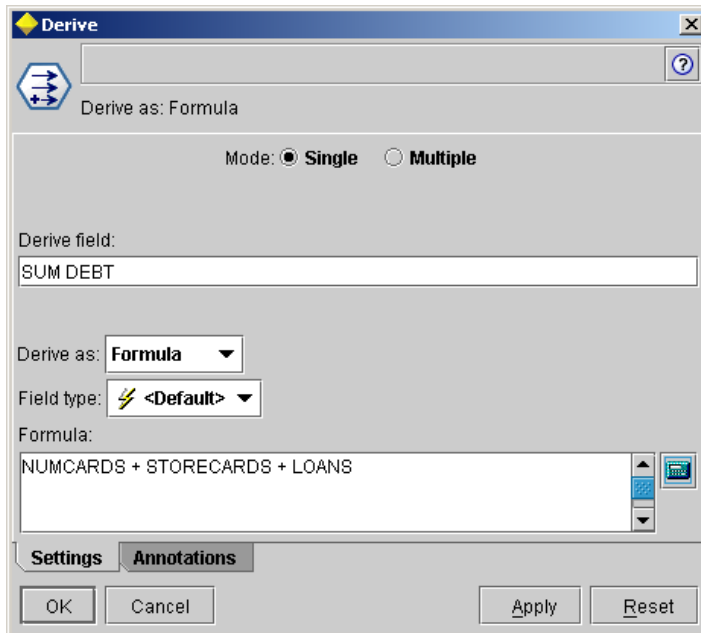
Click **LOANS** in the Fields list box, then click the **Insert** button 

Figure 5.16 Completed Expression in the Expression Builder



Click **OK**

Figure 5.17 Completed Derive Node Dialog Box for a Formula



Click **OK**

On clicking the OK button, we return to the Stream Canvas, where the Derive node is labeled with the name of the new field.

Derive Type Flag

For this example we will create a new field called CHILDREN, which is True if the number of children field (NUMKIDS) is greater than zero, and False if not.

Place a **Derive** node from the **Field ops** palette to the right of the **Derive** node named **SUM DEBT**

Connect the **Derive** node named **SUM DEBT** to the new **Derive** node

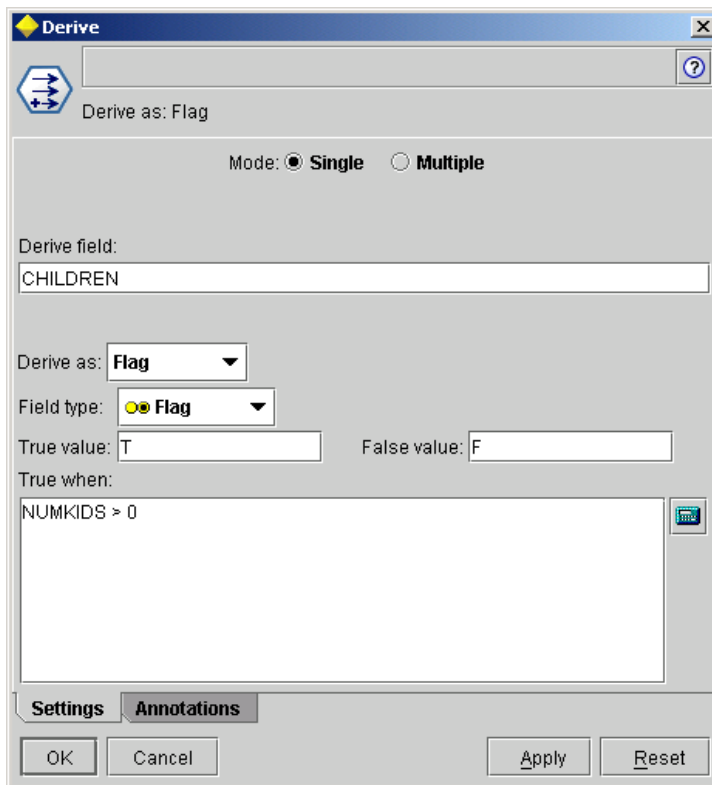
Right-click the new **Derive** node, then click **Edit**

Click **Flag** on the **Derive as:** drop-down list

Type **CHILDREN** in the **Derive field:** text box (replace the original name)

Type **NUMKIDS > 0** in the **True when:** text box

Figure 5.18 Derive Node Dialog Box for Type Flag



If the number of children (NUMKIDS) is greater than 0 for a record, the CHILDREN field will be assigned the value “T”. Otherwise, it will be assigned the value “F”.

Click **OK**

Derive Type Set

For this example we will create a new field called INCGROUP, which is the INCOME field banded into 3 bands:

- Under 20,000
- 20,000 to 35,000
- Over 35,000

If we wanted to create income categories that were equal width, equal sized, or were based on standard deviation width, we would instead use the Binning node from the Field Ops palette.

Place a **Derive** node from the **Field Ops** palette to the right of the **Derive** node named **CHILDREN**

Connect the **Derive** node named **CHILDREN** to the new **Derive** node

Right-click the new **Derive** node, then click **Edit**

Click **Set** on the **Derive as:** drop-down list

The dialog box contains two options to be completed for each member of the set:

- *Set field to* indicating the member of the set
- *If this condition is true* indicating the test condition for the member

Clementine will test the conditions and assign the value stored in *Set field to* for the first condition that applies. The *Default value* is assigned if no condition test succeeds.

To enter each member of the set, type its value in the *Set field to* text box and the test condition in the *If this condition is true* text box. If you have a required value for the set when none of the conditions apply, enter this in the *Default value* text box. The Expression Builder can be used to construct the If condition.

Type **INCGROUP** in the **Derive field:** text box

Type **No income** in the **Default:** text box

Type **Low** in the first **Set field to:** text box

Type **INCOME < 20000** in the **If this condition is true:** text box

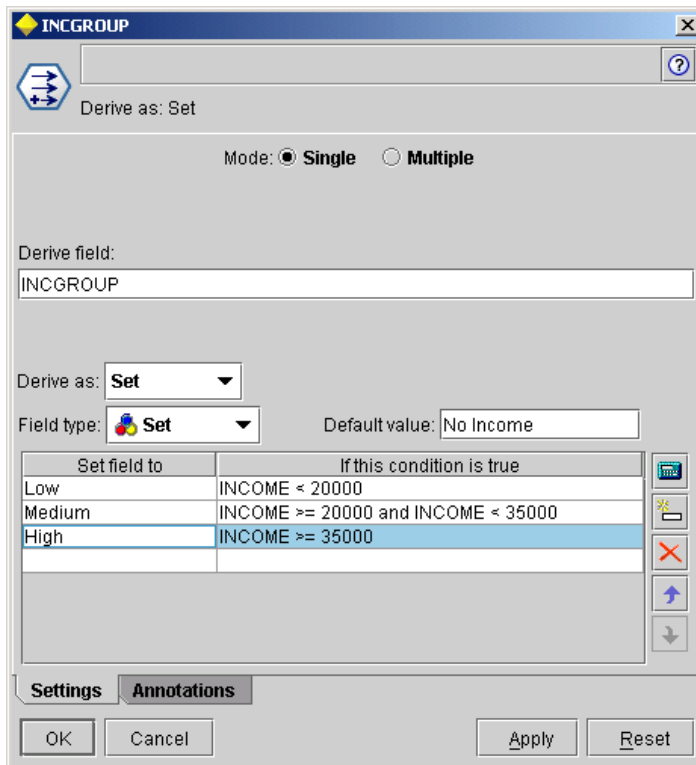
Type **Medium** in the next **Set field to:** text box

Type **INCOME >= 20000 and INCOME < 35000** in the **If this condition is true:** text box

Type **High** in the **Set field to:** text box (replace old value)

Type **INCOME >= 35000** in the **If this condition is true:** text box

Figure 5.19 Derive Node Dialog for Set Derive



Click **OK**

Derive Type Conditional

For this example we will create a new field called INCCARDS, which is the ratio of income to number of store cards (STORECARDS).

Place a **Derive** node from the **Field Ops** palette to the right of the **Derive** node named **INCGROUP** in the Stream Canvas

Connect the **Derive** node named **INCGROUP** to the new **Derive** node

Right-click the new **Derive** node, then click **Edit**

Click **Conditional** on the **Derive as** drop-down list

Type **INCCARDS** in the **Derive field:** text box

Type **STORECARDS > 0** in the **If:** text box

Type **INCOME / STORECARDS** in the **Then:** text box

Type **INCOME** in the **Else:** text box

Figure 5.20 Derive Node Dialog for Conditional Derive



The conditional type will only allow two conditions. If the If: expression applies then the Then: expression will be calculated, otherwise the Else: expression will be calculated.

Click **OK**

Reclassify Node

The Reclassify node allows you to recode or reclassify the data values for fields of set or flag type. For example, a field that stores a customer's specific job position may be more useful for prediction models if it is reclassified into broader job categories. The reclassified values can replace the original values for a field, although a safer approach is to create a new field, retaining the original. We will demonstrate by reclassifying the three values of the RISK field (bad loss, bad profit, good risk) into two values (bad and good).

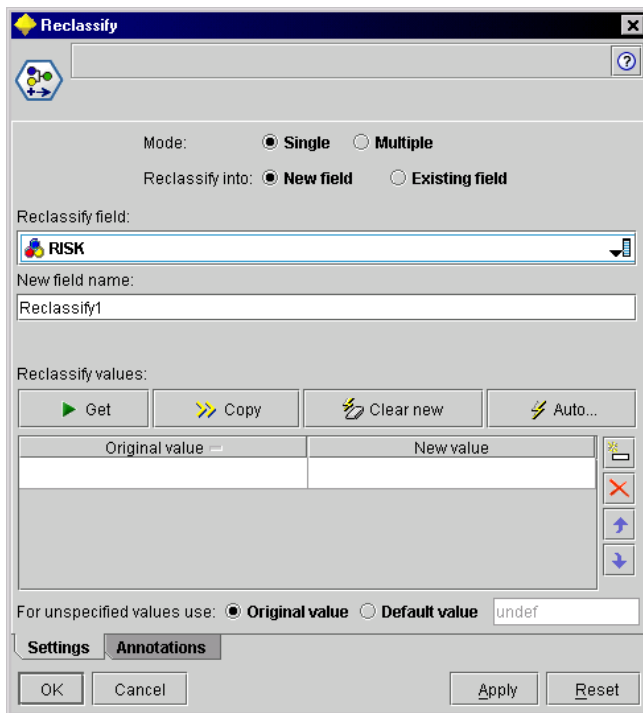
Place a **Reclassify** node from the **Field Ops** palette to the right of the **Derive** node named **INCCARDS** in the Stream Canvas

Connect the **Derive** node named **INCCARDS** to the **Reclassify** node

Right-click the **Reclassify** node, then click **Edit**

Click the field list button  for **Reclassify field** and select **RISK**

Figure 5.21 Reclassify Dialog



As we saw for the Derive node, the Reclassify node supports Single and Multiple modes. Multiple mode would be useful if the same reclassification rules were to be applied to a number of fields. By default, the new values will be placed in a new field, although the *Reclassify into Existing field* option permits you to modify the values in an existing field.

Within the *Reclassify values* group, the Get button will populate the *Original value* column with values from the upstream Type node or Types tab. Alternatively you can enter the original values directly. The Copy button will copy the values currently in the *Original value* column into the *New Value* column. This is useful if you want to retain most of the original values, reclassifying only a few. The Clear new button will clear values from the *New value* column (in case of errors), and the Auto button will assign a unique integer code to each value in the *Original value* column. This option is useful for replacing sensitive information (customer IDs, customer names, product names) with alternative identifiers.

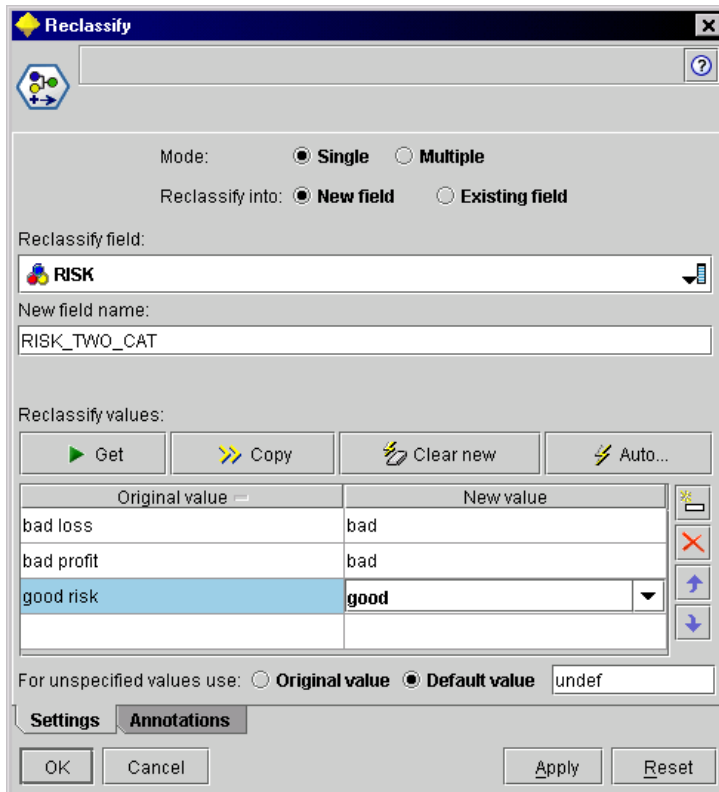
You have options to use the *Original value* or a *Default value* when a value not specified in the Original value column is encountered in the data stream.

- Type **RISK_TWO_CAT** in the **New field name** box
- Click the **Get** button
- Click the For unspecified values use: **Default value** option button
- Type **bad** in the **New value** box for the **bad loss** row
- Click in the **right side** of the **New value** box for the **bad profit** row, then click the **drop-down arrow**, and select **bad** from the drop-down value list

The New value drop-down list contains the original values and the new values entered thus far (not shown).

Type **good** in the **New value** box for the **good risk** row

Figure 5.22 Completed Reclassify Dialog



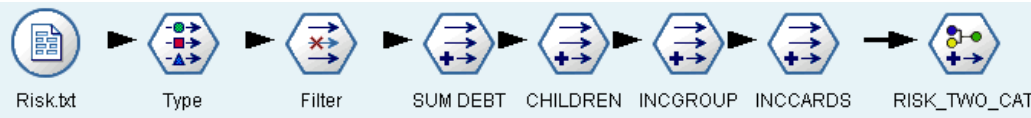
The Reclassify dialog will create a field named RISK_TWO_CAT that will have values bad, good, or undef. To verify that the reclassification works the way you expect, use a Matrix node (discussed in Chapter 6) to compare the original and new fields.

Click **OK**

Executing Field Operation Nodes Simultaneously

In the previous examples we attached each of the field operations nodes (except the first) to the previous one. Because of this, all the new fields will be added to the same stream, which is what we want. If we wished to create a separate stream for each field operation node, we could have attached each directly to the Type node.

Figure 5.23 Field Operation Nodes Placed in One Stream



The new fields can be used simultaneously in downstream analyses. To demonstrate, we will add a new Table node to the stream.

Place a **Table** node from the Output palette above the **Reclassify** node (named **RISK_TWO_CAT**)

Connect the **Reclassify** node to the new **Table** node

Right-click the new **Table** node, then click **Execute**

Figure 5.24 Table Showing Fields Created by Field Operations Nodes

	IS	LOANS	RISK	SUM DEBT	CHILDREN	INCGROUP	INCCARDS	RISK_TWO_CAT
1	2	0	good risk	4 T	High	29972.000	good	
2	1	0	bad loss	2 T	High	59692.000	bad	
3	2	1	good risk	4 T	High	29754.000	good	
4	1	1	bad loss	4 F	High	59463.000	bad	
5	1	0	good risk	3 F	High	59393.000	good	
6	1	1	good risk	4 T	High	59276.000	good	
7	2	0	good risk	3 F	High	29600.500	good	
8	1	1	good risk	4 T	High	59193.000	good	
9	2	1	bad loss	4 T	High	29589.500	bad	
10	2	1	good risk	4 T	High	29518.000	good	
11	1	1	bad profit	3 F	High	58914.000	bad	
12	1	0	bad profit	2 T	High	58878.000	bad	
13	1	0	good risk	3 F	High	58785.000	good	
14	1	0	bad loss	2 F	High	58529.000	bad	
15	1	0	good risk	3 F	High	58505.000	good	
16	1	0	good risk	2 T	High	58381.000	good	
17	2	0	good risk	3 F	High	29013.000	good	
18	1	1	bad profit	4 T	High	57718.000	bad	
19	2	1	good risk	4 T	High	28844.500	good	
20	2	1	bad loss	4 T	High	28841.500	bad	

Click **File..Close** to close the Table window

Although not shown here, another useful field operation node is the Binning node, which converts numeric fields into set fields – for example, income into income groups.

Automatically Generating Operational Nodes

In this chapter we have introduced ways of manually adding operational nodes. Clementine also allows automatic generation of many of the nodes we have manually created. In the previous chapters, output windows often contained a Generate menu. This menu frequently allows automatic generation of Filter, Select, Derive, and other nodes. For example, the Quality node can generate Select and Filter nodes to remove records and fields with missing data and the Distribution node output node can generate Select, Derive, Balance (see Help system for more information), and Reclassify nodes. We will now demonstrate a few examples of automatically generating Derive and Select nodes.

Automatically Generating Derive Nodes

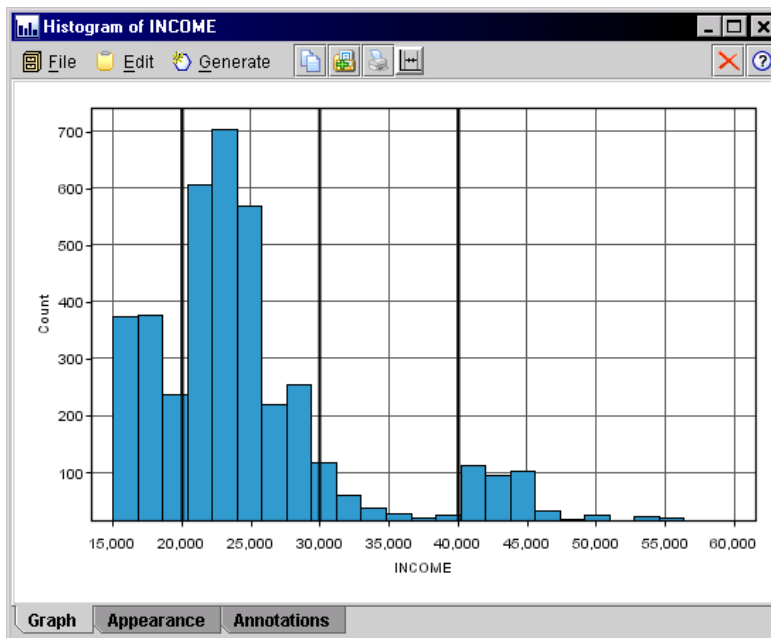
In this section we will generate possibly the most difficult of the Derive nodes to manually create: the Set type (although in many cases the Binning node will accomplish the task easily).

- Place a **Histogram** node from the Graphs palette to the right of the **last Derive** node (named **INCCARDS**)
- Connect the **Derive** node (named **INCCARDS**) to the **Histogram** node
- Double-click the **Histogram** node
- Click the field list button and select **INCOME**
- Click the **Execute** button

After the Histogram graph window appears:

- Click on the **Histogram** graph at value **20,000** (a black vertical line should appear)
- Click on the **Histogram** graph at (approximately) **30,000**
- Click on the **Histogram** graph at (approximately) **40,000**

Figure 5.25 Histogram with Interactive Lines Dividing Income into Four Groups

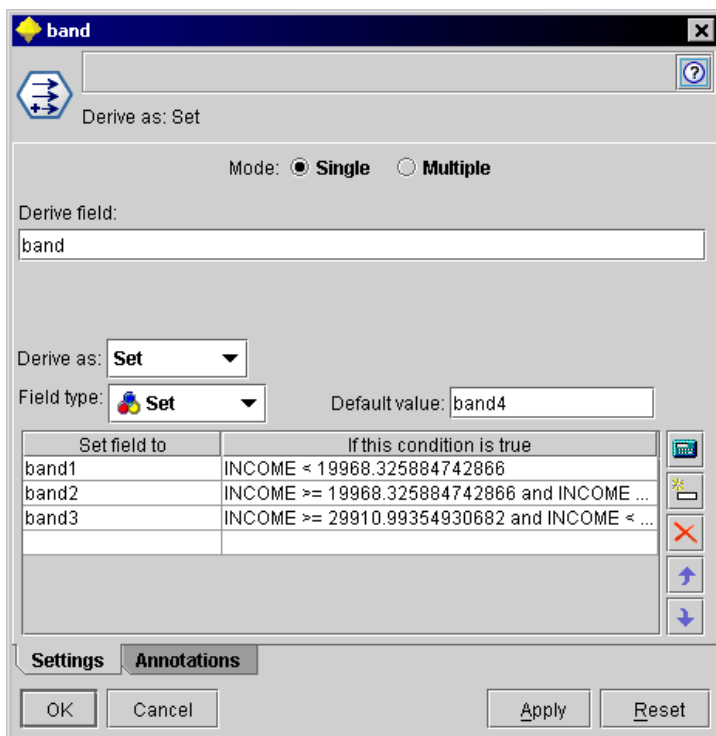


- Click **Generate..Derive Node**
- Click **File..Close** to close the Histogram window

A Derive node named band should appear in the top left corner of the Stream Canvas.

Double-click on the new **Derive** node (named **band**)

Figure 5.26 Automatically Generated Derive Node Dialog



A Derive node of type Set is generated and we see the condition for a record to be classified in *band1* (roughly, income under 20,000). The default values for the set members (*band1*, *band2*, etc.) can be replaced (replace original value in the Set field to text boxes, etc.) The conditions can be edited (for example, to make the cutoff exactly 20,000). The field name can also be changed by typing a new name in the Derive field text box (for example, INCOME CATEGORY in place of *band*).

Click **Cancel** to close the generated **Derive** node dialog

These data manipulation nodes typically precede the Modeling section of a stream.

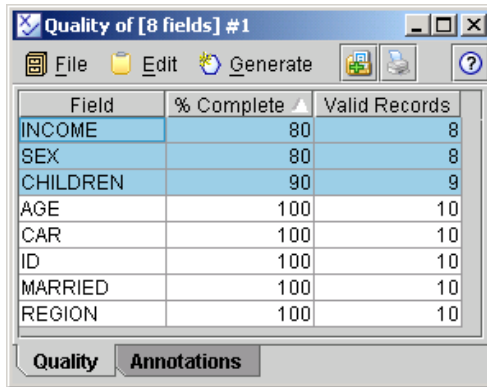
At this point you may want save these data manipulation nodes in a Clementine Stream file named *Data Preparation.str*.

Automatically Select Records or Fields with No Missing Values

You can use the output from the Quality node to generate a Filter node that removes fields with blank (missing values) or a Select node that removes all records with blank (missing) information. To demonstrate we will open a stream that checks for data quality (see Chapter 4 for details on how the stream was created).

- Click **File..Open Stream**, move to the **c:\Train\ClemIntro** directory
- Double-click on the **DataQuality.str** stream file
- Execute the **Quality** node
- Click on the **% Complete** header (to sort table by % Complete)
- Click and drag to select **INCOME**, **SEX** and **CHILDREN** in the Quality output window

Figure 5.27 Quality Output Window (Fields with Missing Values Selected)

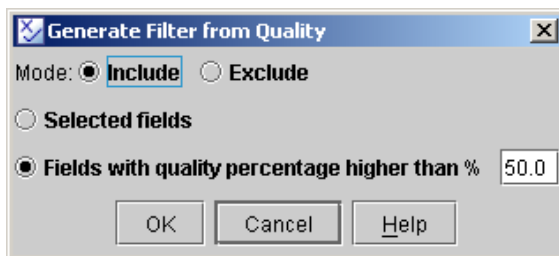


Field	% Complete	Valid Records
INCOME	80	8
SEX	80	8
CHILDREN	90	9
AGE	100	10
CAR	100	10
ID	100	10
MARRIED	100	10
REGION	100	10

We selected all fields with missing values; in practice, you might focus on fields with a substantial proportion of missing data.

Click **Generate..Filter Node**

Figure 5.28 Generate Filter Node from Quality Dialog



Mode: Include Exclude

Selected fields

Fields with quality percentage higher than %

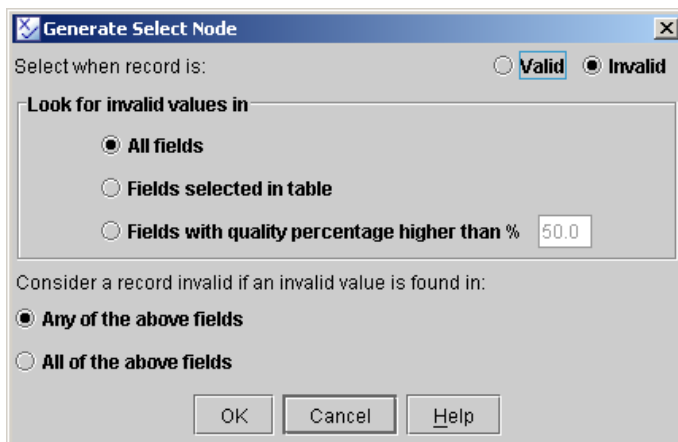
OK Cancel Help

This dialog will generate a Filter node that can include or exclude the fields selected in the Quality output window. Field selection can also be based on the percentage value in the *% Complete* column of the Quality output window. In this way, you can easily exclude fields with missing data from analyses, when needed, even if there are many fields in the stream.

Click **Cancel**

Click **Generate..Select Node**

Figure 5.26 Generate Select Node from Quality Dialog



Select when record is: Valid Invalid

Look for invalid values in

All fields

Fields selected in table

Fields with quality percentage higher than %

Consider a record invalid if an invalid value is found in:

Any of the above fields

All of the above fields

OK Cancel Help

This dialog will generate a Select node that can delete or retain records in the stream based on whether they have missing values. There are options to check *All fields*, the fields selected in the table, or fields based on the percentage value in the *% Complete* column of the Quality output window. In addition, there is an option to consider a record invalid if there is a missing value in any of the indicated fields or in all of the indicated fields.

These generated nodes provide a means of excluding (or retaining to examine more carefully) missing values on either a field or record basis.

Click **Cancel**

Summary

In this chapter you have been given an introduction to a number of methods of manipulating your data.

You should now be able to:

- Enter simple expressions using CLEM
- Create a Select, Filter or Field Reorder node
- Create different types of Derive nodes
- Use the Reclassify node
- Use Clementine to automatically generate Derive, Select and Filter nodes

Chapter 6

Looking for Relationships in Data

Overview

- Introduce the Web and Matrix nodes to investigate relationships between symbolic fields
- Introduce the use of correlation within the Statistics node to investigate relationships between numeric fields

Objectives

In this session we will explore ways in which Clementine can be used to study relationships between fields.

Data

In this chapter we will work with the credit risk data (*Risk.txt*) used in previous chapters. The data file contains information concerning the credit rating and financial position of 4117 individuals, along with basic demographic information, such as marital status and gender. In this section we are primarily interested whether any of the fields in the data can be related to credit risk.

Introduction

In most data mining projects, you will be interested in investigating whether there are relationships between data fields. With respect to our current data set, simple questions may include:

- Is credit risk directly related to income?
- Do the males differ from females with respect to credit risk?
- If an individual has a large number of current credit cards, loans, and store cards, does this mean that he is more likely, or less likely, to be a bad credit risk?

Although many of the machine learning and modeling techniques available in Clementine will answer these questions, it is often a better approach to try to first understand the basic relationships among a small number of fields. By eliminating fields that are not at all related to the field you wish to predict, or combining related fields into a composite measure, models can be built more quickly and possibly perform with greater accuracy and efficiency.

The methods used for examining relationships between fields depend on the type of fields in question. In the following sections we will introduce several techniques, some for studying relationships between symbolic fields and one for investigating relationships between numeric fields. In addition, we discuss but do not run plots that involve a mixture of symbolic and numeric fields (these plots are presented in later chapters)

Studying Relationships between Symbolic Fields

In this section we will introduce two methods for examining whether symbolic fields are related. The first is the Matrix node used to display the relation between a pair of symbolic fields. We will extend this to more than two symbolic fields with the graphical Web node.

Matrix Node: Relating Two Symbolic Fields

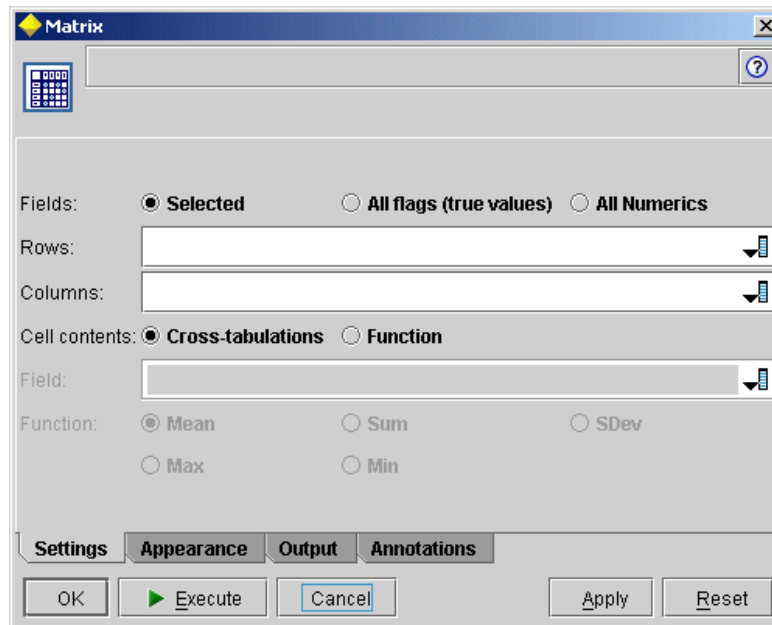
The Matrix node performs crosstabulations of two symbolic fields within the data, and shows how values of one field relate to those of a second field. A third numeric field can be included as an overlay field to see how it varies across the symbolic pair relationship. The Matrix node is located in the Output palette and is thus a terminal node.

In this example we will use the Matrix node to see whether there are any relationships between the field we are going to try to predict, credit risk (RISK), and some of the other symbolic fields within the data. We begin with investigating whether there is a difference between the two gender groups with respect to their credit risk.

We will build our stream starting with the data source and Type nodes saved in the Riskdef.str stream file.

- Click **File..Open Stream**, move to the **c:\Train\ClemIntro** directory and double-click on **Riskdef.str**
- Place a **Matrix** node from the **Output** palette to the right of the **Type** node
- Connect the **Type** node to the **Matrix** node
- Double-click on the **Matrix** node

Figure 6.1 Matrix Node Dialog



The default setting of the *Fields:* option (*Selected*) will display the fields selected in the *Rows* and *Columns* boxes (one field in each), which is what we want. Note that only one field for the *Rows* and one field for the *Columns* can be selected. Thus the Matrix node will produce one matrix at a time.

Less often used, the *All flags* option will create a symmetric matrix in which one row and one column appear for each Flag field and the cell values contain the co-occurrence counts of True values for the flag

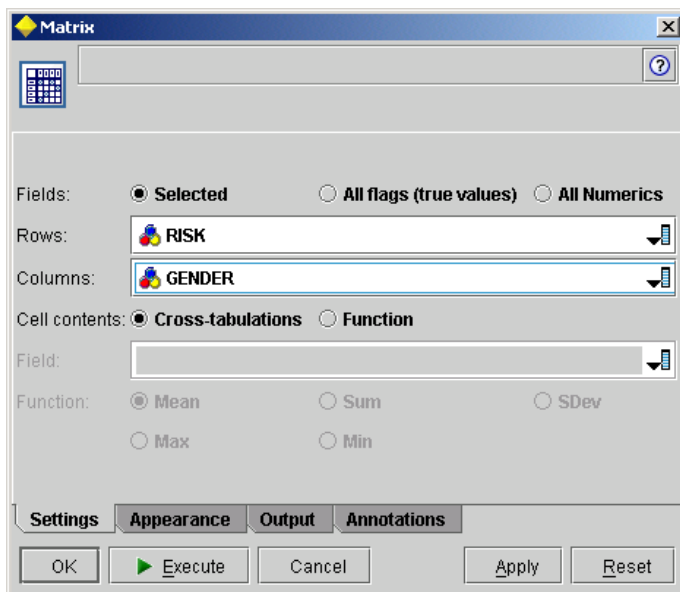
fields. Finally, the rarely used *All Numerics* option will produce a table with one row and one column for each numeric field, and the cells contain the sum of the products of each pair of fields (a cross-products table).

The default option for *Cell contents* is *Cross-tabulations*. The alternative is a function applied to a selected numeric field and this will have the effect of calculating the *Sum*, *Average*, *Minimum* or *Maximum* value of this field for each of the cells in the matrix.

Within the Matrix dialog box:

- Click the Fields list button in the **Rows:** list and select **RISK**
- Click the Fields list button in the **Columns:** list and select **GENDER**

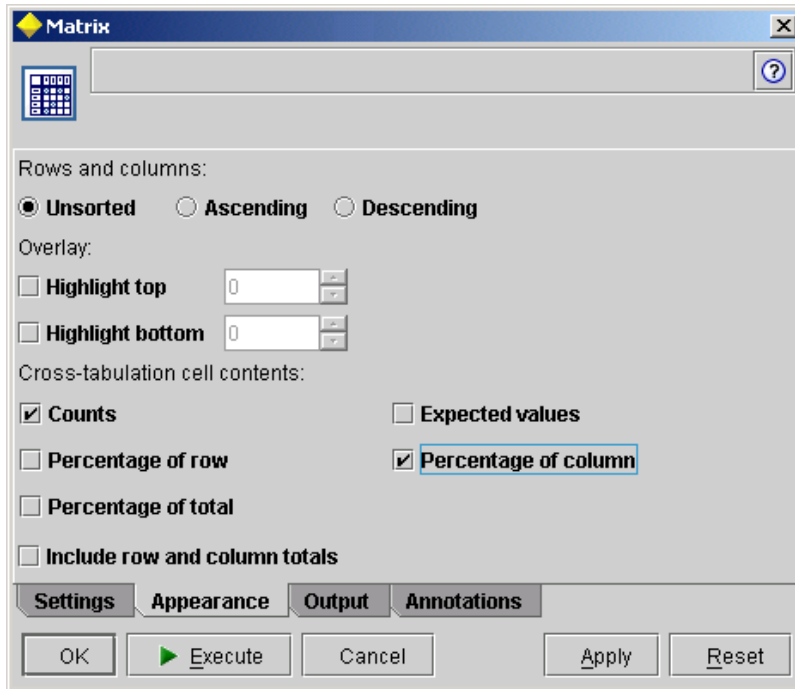
Figure 6.2 Matrix Node Dialog Box



The default table contains counts in the cells of the matrix. Alternatively, different percentages can be requested by using the Appearance tab. We will ask for column percentages in order to compare men and women with respect to their credit risk.

- Click the **Appearance** tab
- Select **Counts** (if not already checked)
- Select **Percentage of column**

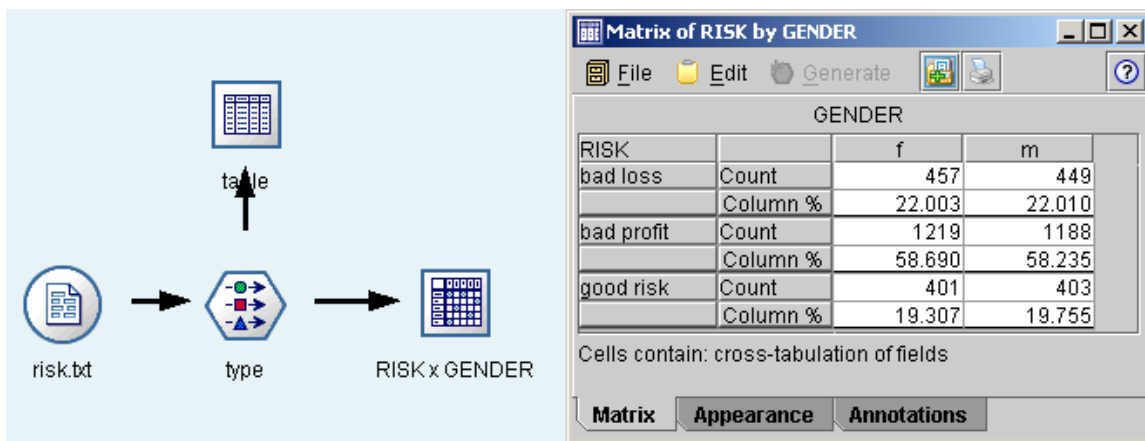
Figure 6.3 Matrix Node: Appearance Tab



Cells with the highest or lowest values in the table can be highlighted by entering the number of cells in the *Highlight top / bottom* options. This feature can be useful when percentages are displayed. As with most output nodes, the Output tab can be selected for setting the output destination; by default, output is sent to the screen.

Click the **Execute** button

Figure 6.4 Matrix Table of Gender and Credit Risk



Examining the matrix table, there appear to be no differences between the two gender groups in their distribution across credit risk categories. For instance, 22.003% of the women are categorized as bad loss, while 22.010% of the men belong to this category, so there is essentially no difference between the two groups.

To repeat this exercise for a second symbolic field in relation to credit risk we can either edit the existing Matrix node, or create a new one. We next investigate the relationship between credit risk and how often an individual is paid (monthly or weekly).

- Click **File..Close** to close the Matrix output window
- Double-click on the **Matrix** node, and then click the **Settings** tab
- Click the Fields list button in the **Columns:** list and select **HOWPAID**
- Click **Execute**

Figure 6.5 Matrix Table of Salary Payment Schedule and Credit Risk

		HOWPAID	
RISK		monthly	weekly
bad loss	Count	359	547
	Column %	17.720	26.160
bad profit	Count	1090	1317
	Column %	53.801	62.984
good risk	Count	577	227
	Column %	28.480	10.856

Cells contain: cross-tabulation of fields

Matrix Appearance Annotations

The matrix table suggests that individuals paid monthly are more likely to be good credit risks than those paid weekly (28.5% against 10.9%).

A restriction of the Matrix node is that only one pair of symbolic fields can be investigated at a time.

Note

One useful feature is that the summaries requested in the Appearance tab can be changed after the matrix output is generated. Different summaries can be obtained directly from the Matrix node output, without requiring re-execution of the Matrix node. Thus, you could easily view row percentage within the matrix output displayed in the last two figures.

We next introduce a graphical method of visualizing relationships between two or more symbolic fields: the Web node.

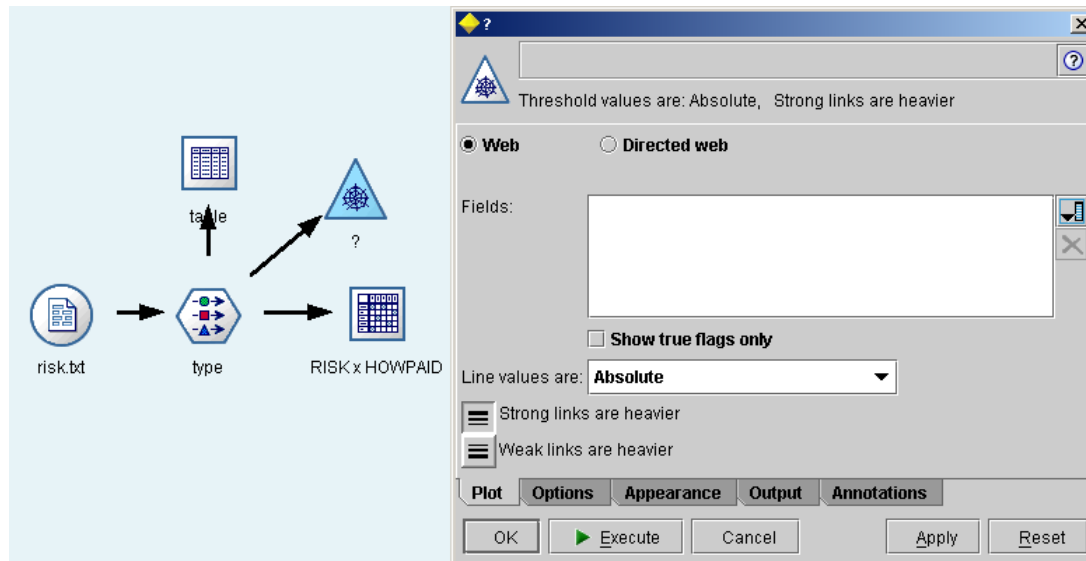
- Click **File..Close** to close the Matrix output window

The Web Node

The Web node, located in the Graphs palette, can be used to show the strength of connection between values of two or more symbolic fields. A web plot consists of points representing the values of the selected symbolic fields. These points are connected with lines, whose thickness depicts the strength of the connections between the values. Thin lines represent weak connections while heavy, solid lines represent strong connections. Intermediate strength connections are drawn as normal lines. Web displays are interactive and it is possible to vary the threshold settings (what defines a weak or strong connection), hide irrelevant fields, modify the layout, and generate nodes.

Place a **Web** node from the **Graphs** palette near the **Type** node
 Connect the **Type** node to the **Web** node
 Double-click the **Web** node

Figure 6.6 Web Node Dialog

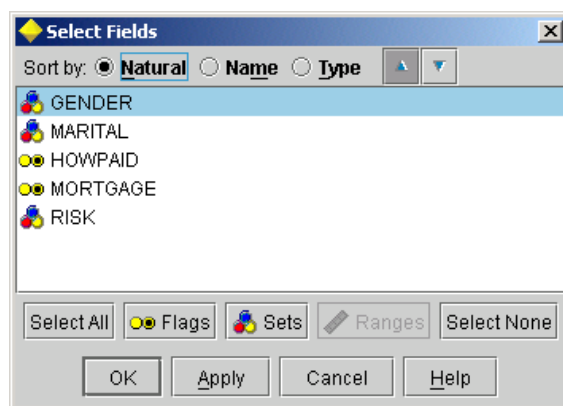


Two types of plots are available. In a web plot the relations between all selected symbolic fields are shown, while in a directed web plot only relations involving a specified target field are displayed.

The *Show true flags only* check box allows only the True response for flag fields (as defined in the Type node or Types tab of a source node) to be shown in a web plot. This is a very useful feature in displaying the relationships among many products (bought yes or no), as we will see in a later chapter.

Click the Field List button 

Figure 6.7 Select Fields Dialog



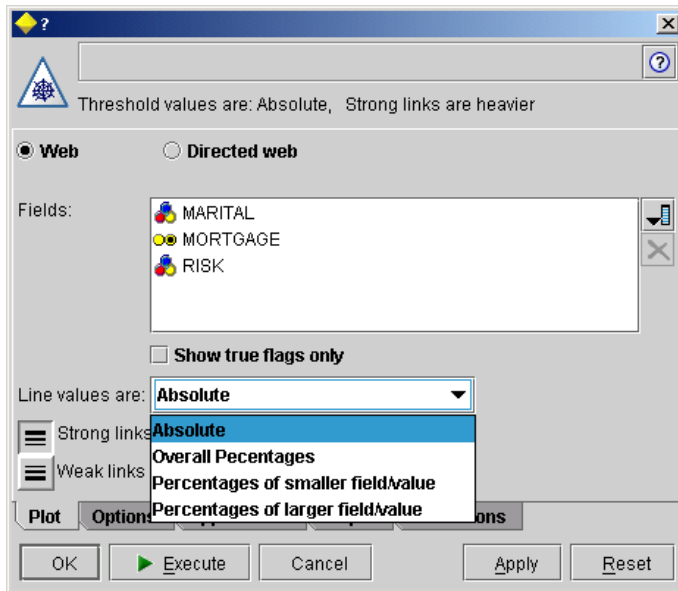
Only symbolic fields (sets and flags) are eligible for a web plot. All fields, all flag fields, or all set fields can be chosen at once by clicking the respective button. In addition Select None can be chosen to deselect fields. The *Ranges* button is inactive, since fields of range type are not appropriate for web plots. In this example we will investigate the relationship between credit risk and the two other symbolic fields in the data: marital status and whether the individual has a mortgage or not.

Select **MARITAL**, **MORTGAGE** and **RISK**
Click **OK** to return to the Web dialog box

The web plot will show strong, normal, and weak links. What constitutes a strong or weak link is defined by the threshold value. Several types of thresholds are available, as shown in below.

Click the **Line values are** drop-down list

Figure 6.8 Threshold Types

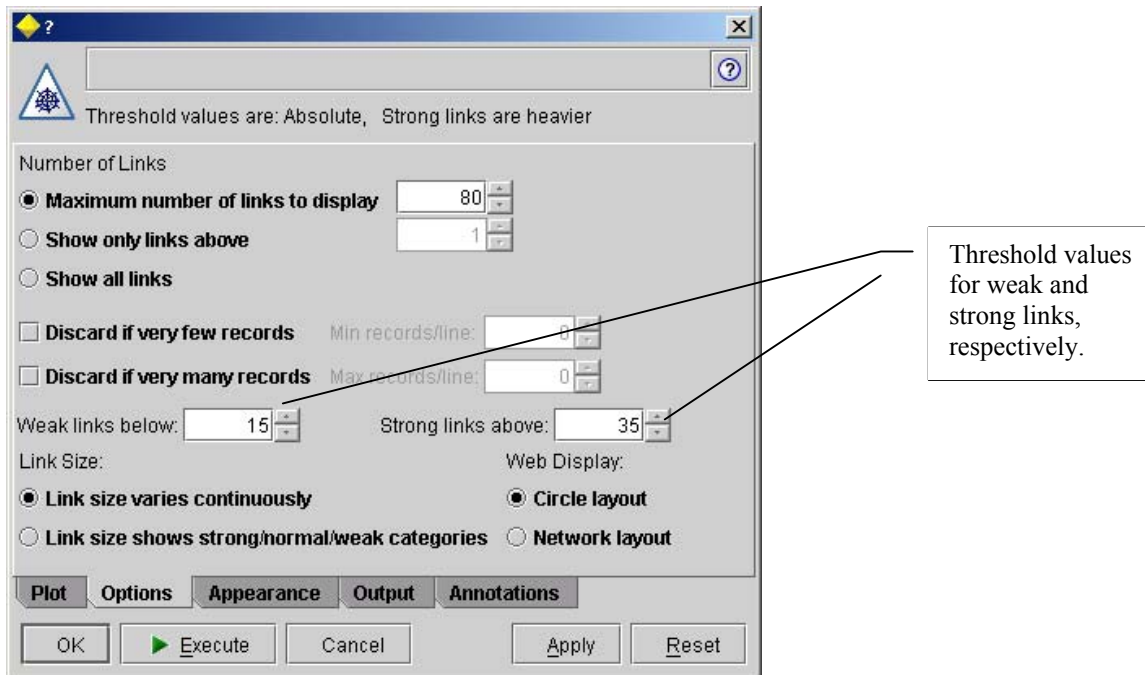


Line values can represent counts (*Absolute*), percentages of the overall data (*Overall Percentages*), or percentages of either the smaller or larger field/value. For example, if 100 records have value X for one field, 10 records have value Y for a second field, and there are 7 records containing both values, the connection involves 70% of the smaller field and 7% of the larger field.

The threshold values themselves are set under the Options tab.

Click the **Options** tab

Figure 6.9 Setting Threshold Values



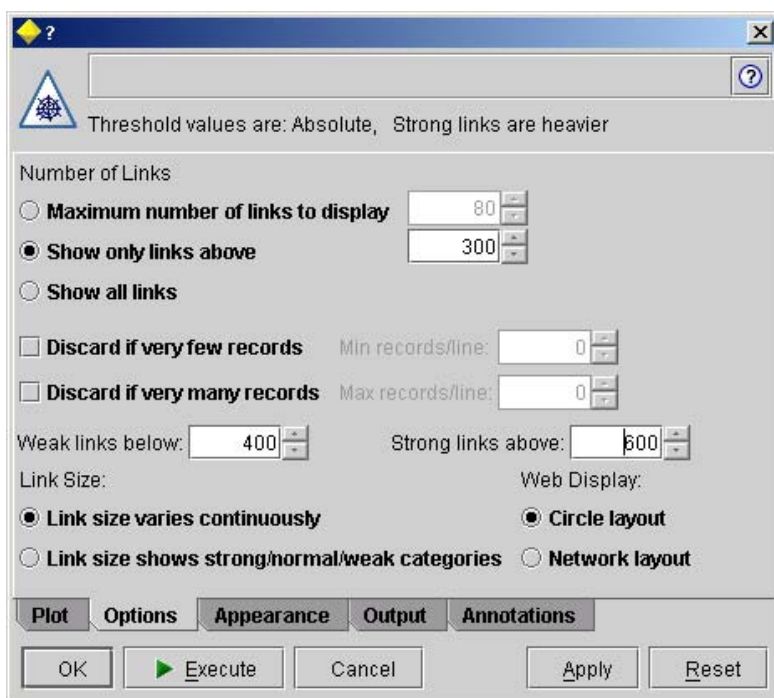
The number of links in the web plot is controlled by: (1) choosing a maximum number of links, (2) displaying links only above a specified value, or (3) displaying all links. The *Discard* options allow you to ignore connections supported by too few records (only when thresholds are percentages) or too many records.

The *Link Size* options control the size of links. The *Link size varies continuously* option will display a range of link sizes reflecting the variation in connection strengths based on actual data values. The alternative, *Link size shows strong/normal/weak categories*, will display three strengths of connections--strong, normal, and weak. The cut-off points for these categories can be specified above as well as in the final graph.

We have over 4000 records and we will set the thresholds initially at 300, 400 and 600 records, respectively. We will see how this can be adjusted once the plot has been produced.

- Click **Show only links above** option button
- Type **300** in the **Show only links above** box
- Type **400** in the **Weak links below:** box
- Type **600** in the **Strong links above:** box

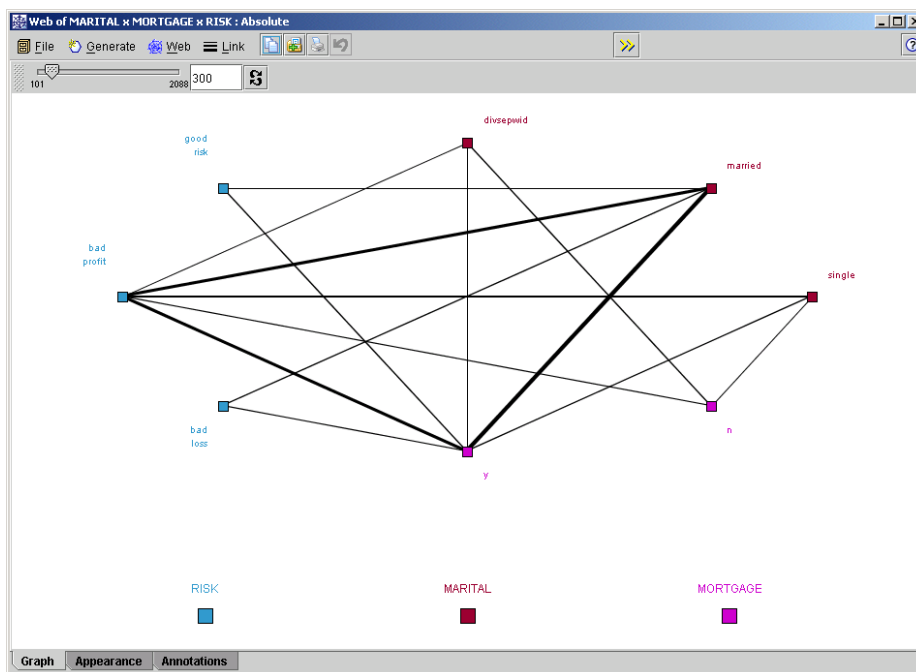
Figure 6.10 Options for the Web Plot



Click **Execute**

Drag the lower corner of the resulting Web graph window to enlarge it


Figure 6.11 Web Plot of Marital Status, Having a Mortgage, and Credit Risk



Remember, the thickness of the lines varies continuously, reflecting how strong a link is. At first glance, the link between *married* and MORTGAGE *y* is the strongest. Looking at the different categories of RISK, *bad profit* is strongly connected to MORTGAGE *y*: and (although somewhat less strongly) to *married*.

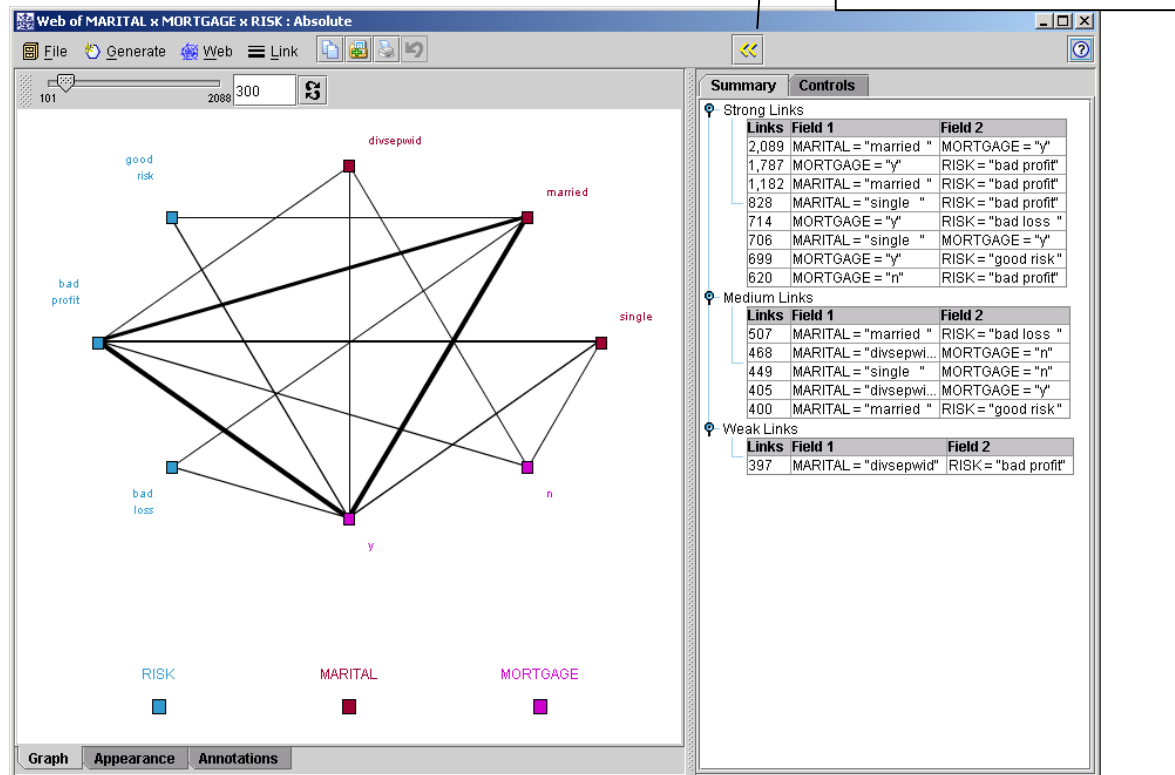
Apparently, the bad profit group is characterized by married persons having a mortgage (based on the pairwise counts).

Besides a visual inspection of the thickness of the link to look for weak and strong links, we can ask for the associated counts in several ways. One method is to position the cursor over a link and a pop-up will display the number of records having that particular combination of values (not shown). The disadvantage of this method is that we have to check the links one-by-one.

An alternative is to ask for a web output summary, using the web summary button  in the Toolbar.

Click 

Figure 6.12 Web Summary

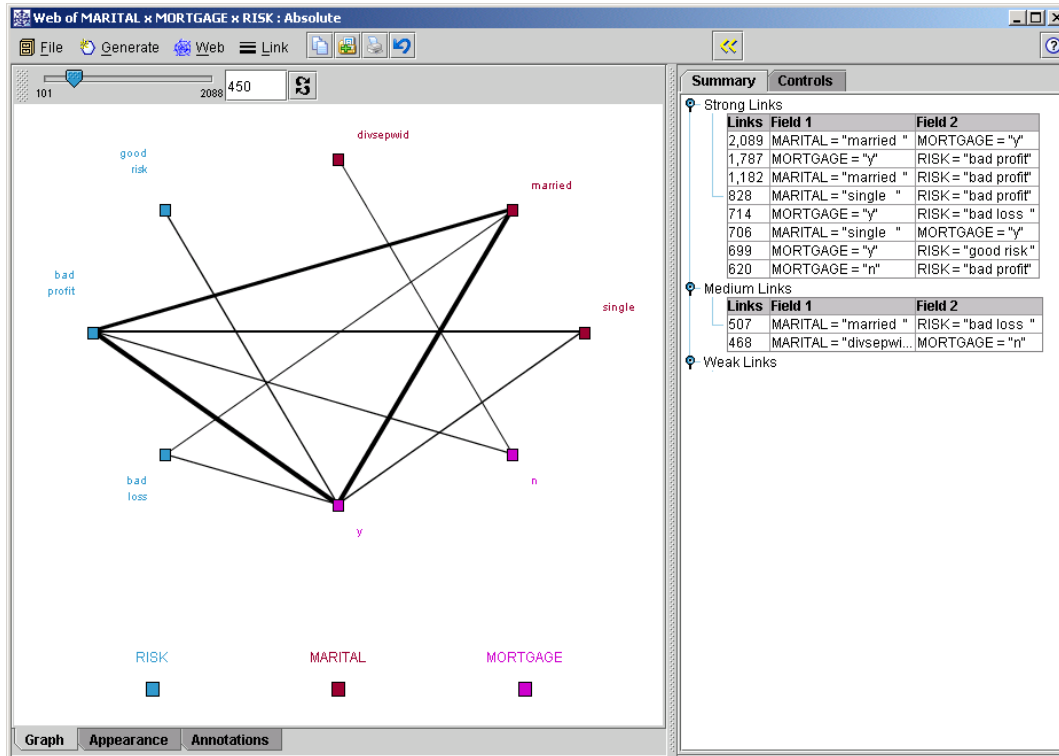


Having seen the counts, we will change the web plot in order to see the strong links more clearly. Clementine allows you to do this in several ways.

First, we might want to use the slider control, located in the upper left corner of the window. In the figure above the slider starts at value 101 and ends at 2088 (recall, that the strongest link has 2,089 records). On the right of the slider we see a text box for specifying the minimum link size, here with value 300. You can either use the slider control or the textbox to set the minimum link size. For instance:

Use **slider control** to discard links weaker than **450** (alternatively, type 450 in the slider control box and then press Enter key)

Figure 6.13 Web Plot: Setting Minimum Link Size Interactively



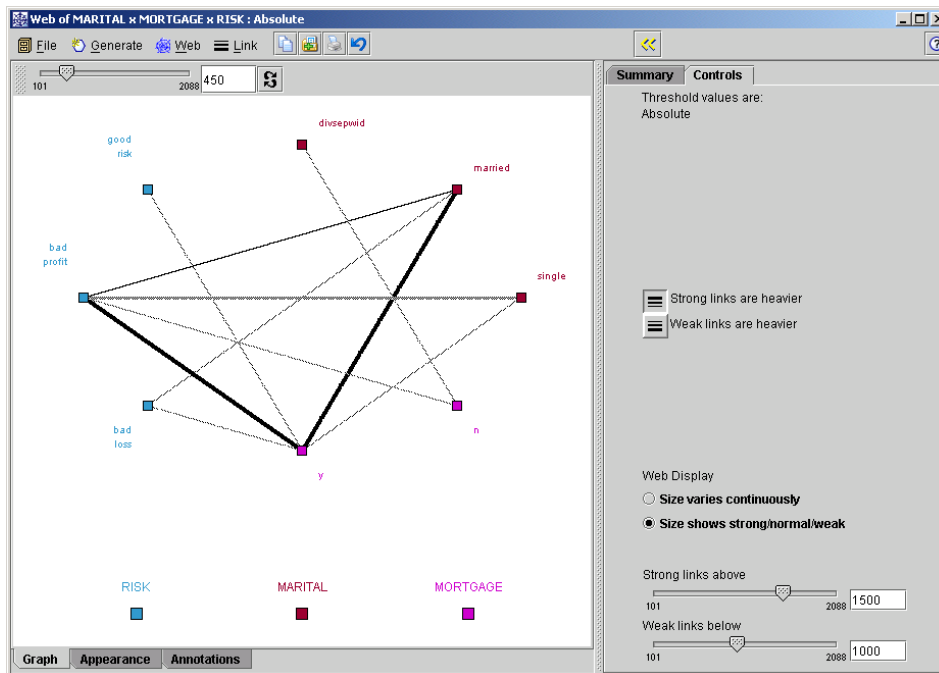
Besides setting the minimum link size interactively, you can also re-specify what constitute weak and strong links:

Click the **Controls** tab in the Web Summary output

Set the **Web Display** option to **Size shows strong/normal/weak**

Use slider control or text box to set the value for **strong links above to 1500** and the value for **weak links below to 1000**

Figure 6.14 Web plot: Setting Strong and Weak Link Size Interactively



To facilitate a better interpretation of the web plot, we finally should mention the options of hiding categories or moving categories. For instance, in our plot *divsepwid* is not connected to any of the three RISK categories, so we can hide this category. To do so:

- Right-click the point representing **divsepwid**
- Click **Hide** from the context menu (result not shown)

To move a category, simply drag it to a new location in the plot (not shown).

As in other output windows we have the opportunity to generate a Select or Derive node. For example, suppose that we want to create a flag field indicating whether an individual is married, has a mortgage and belongs to the bad profit category:

- Click the link connecting **married** and **y** (the link will turn red if selected)
- Click the link connecting **bad profit** and **y** (the link will turn red if selected)
- Click **Generate..Derive Node("And")**

As before, a Derive node will be generated and will appear in the Stream Canvas. This node can be edited or included in the stream as is.

We will now go on to investigate relationships between numeric fields in the data.

- Click **File..Close** to close the Web graph window

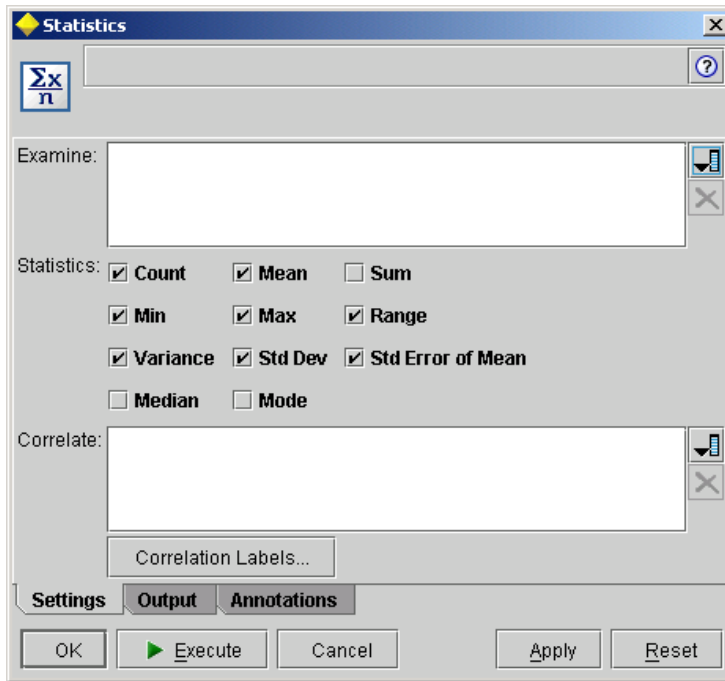
Correlations between Numeric Fields

When investigating relationships between numeric variables, a linear correlation is commonly used. It measures the extent to which two numeric fields are linearly associated. The correlation value ranges from -1 to $+1$, where $+1$ represents a perfect positive linear relationship (as one field increases the other field also increases at a constant rate) and -1 represents a perfect negative relationship (as one field increases the other decreases at a constant rate). A value of zero represents no linear relationship between the two fields.

Earlier we used the Data Audit node to produce basic descriptive statistics for numeric fields. The Statistics node, which produces summary statistics but no graphs, can provide correlations between fields.

- Place a **Statistics** node from the Output palette near the **Type** node in the Stream Canvas
- Connect the **Type** node to the **Statistics** node
- Double-click on the **Statistics** node

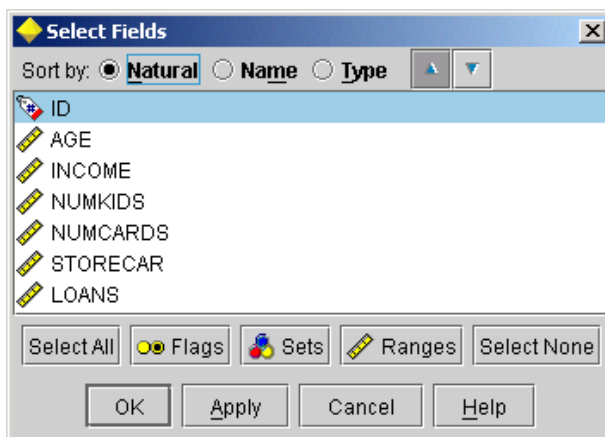
Figure 6.15 Statistics Node



In this example we will ask for correlations between all of the numeric fields, excluding ID.

- Click the **Examine:** field list button 

Figure 6.16 Selecting Fields to Examine



- Click the **Ranges** button to select all numeric fields


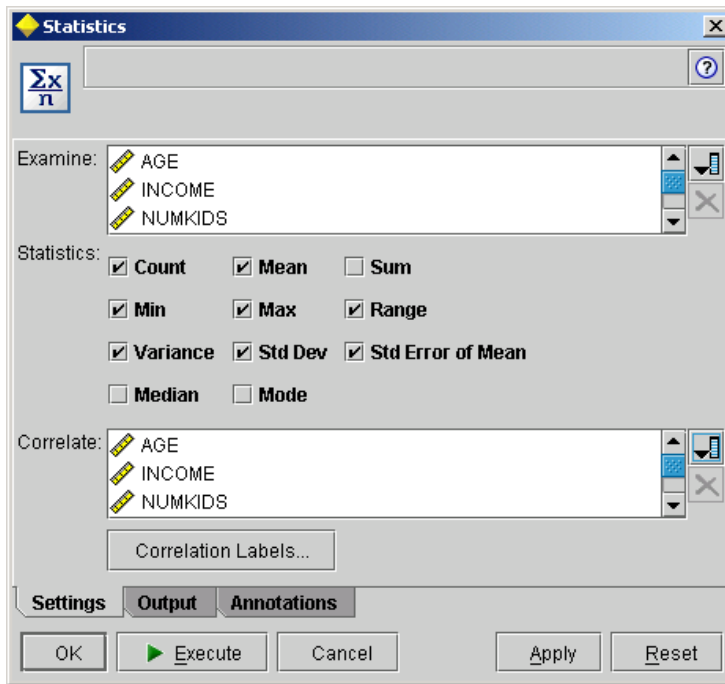
- Ctrl-click **ID** to deselect it, and then click **OK**
- Click the **Correlate:** field list button 
- Click the **Ranges** button to select all numeric fields
- Ctrl-click **ID** to deselect it, and then click **OK**

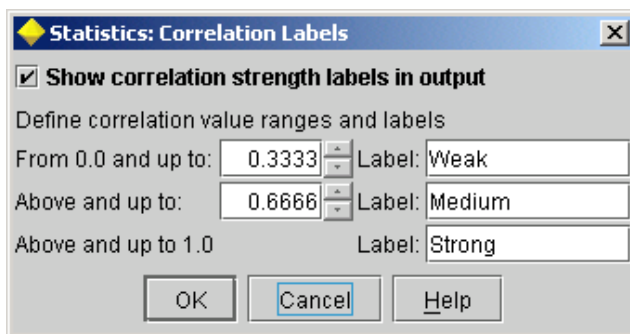
Figure 6.17 Statistics Dialog: Requesting Correlations



Correlations between the fields selected in the *Correlate* list and those selected in the *Examine* list will be calculated. Similar to the web plot, labels will be attached to weak and strong relationships. With the *Correlation Labels* button you define what weak and strong relationships are:

- Click the **Correlation Labels** button

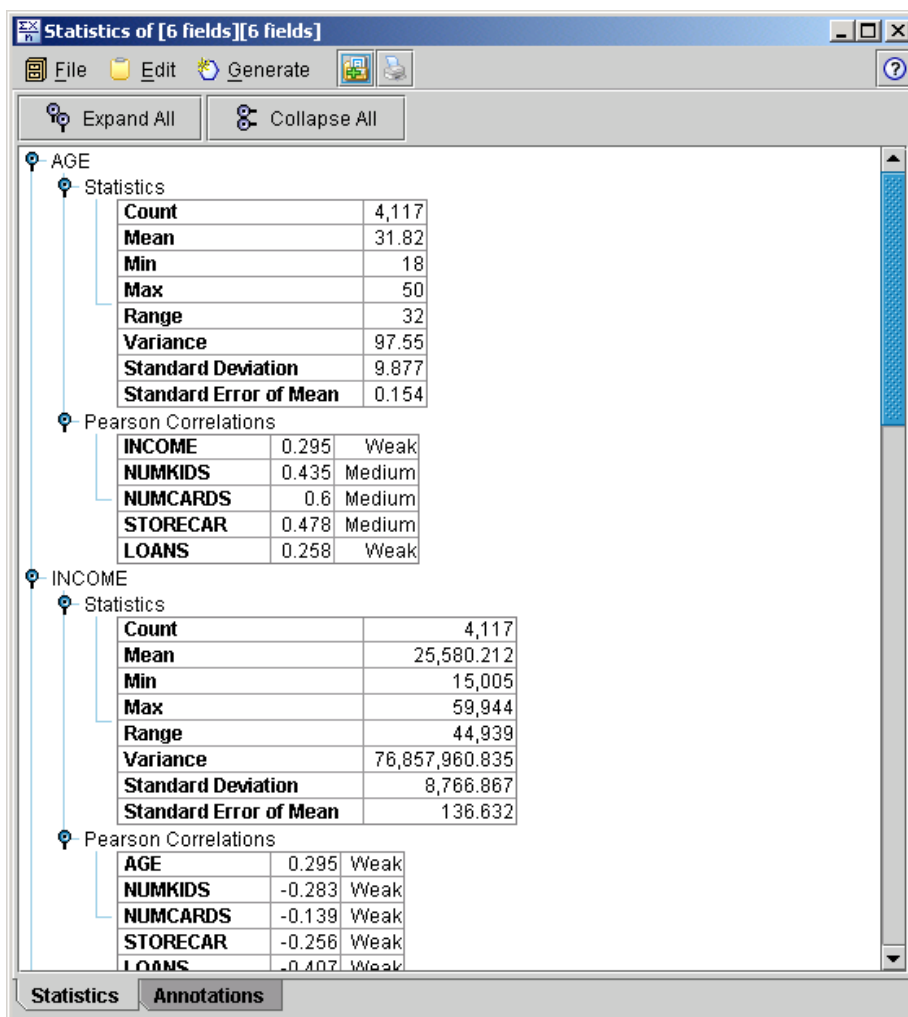
Figure 6.18 Defining Correlation Strength Labels



Here correlations up to .33 (in absolute value) are defined as weak, between .33 and .66 as medium and above .66 as strong. These default values can be changed in the respective text boxes. We will accept these values and ask for the correlation table. Note that these labels are not based on statistical tests, but rather the magnitude of the estimated correlations.

- Click **OK**
- Click **Execute**

Figure 6.19 Statistics Output with Correlations



Along with the descriptive statistics, the report displays the correlation between each pair of selected fields (each field selected in the Correlate field list with each field selected in the Examine field list). The report also suggests how to interpret the strength of the linear association between the fields, according to the definitions set with the Correlation Labels button.

Scrolling through the output, we find strong positive linear relationships between number of loans, children, store cards and credit cards.

One limiting aspect of using correlations is that they only give an indication of linear relationships between the numeric fields. There may be no linear relationship between two fields but there still may be a relationship of another functional form. For example, the correlation between age and income is very weak but experience suggests that these two fields are related to one another in some way.

Click **File..Close** to close the Statistics output window

Extensions

Relations between two symbolic fields can also be displayed using a Distribution node if an overlay field is included, while relations between a symbolic field and a numeric field can be displayed by using the symbolic field as an overlay within the Histogram node. Also, recall that the Data Audit node allows you to

specify an overlay field, which would appear in all graphs produced by that node. Some of these plots are used in later chapters as aids in interpreting models.

Summary

In this chapter you have been introduced to a number of methods to explore relationships in data. You should now be able to:

- Produce a data matrix to investigate a relationship between two symbolic fields
- View a Web node to visualize associations between symbolic fields
- Use correlation to quantify the linear relationship between two numeric fields.

Chapter 7

Modeling Techniques in Clementine

Overview

- Give a brief introduction to the modeling techniques available in Clementine
- Which technique, When and Why?

Objectives

In this session we will briefly introduce the different modeling techniques available in Clementine. We will also discuss when it is appropriate to use each technique and why.

Introduction

Clementine includes a number of machine learning and statistical modeling techniques. Although there are different taxonomies, these techniques can be classified into three main approaches to modeling:

- Predictive
- Clustering
- Associations

In predictive modeling, sometimes referred to as supervised learning, inputs are used to predict the values for an output. Clementine has six predictive modeling nodes available; neural networks, two different rule induction methods, regression and logistic regression analysis, and a sequence detection method.

The different clustering methods, sometimes referred to as unsupervised learning methods, have no concept of an output field. The aim of clustering techniques is to try to segment the data into groups of individuals who have similar patterns of input fields. Clementine has three clustering techniques available: Kohonen networks, k-means clustering, and two-step clustering.

Association techniques can be thought of generalized predictive modeling. Here the fields within the data can act as both inputs and outputs. Association rules try to associate a particular conclusion with a set of conditions. There are two association techniques available within Clementine: Apriori and GRI. In addition, the sequence detection node (mentioned in the predictive modeling section) and an algorithm option to Clementine, called CaprI, will search for common sequences in data; for example, stages in processing customer service problems, or web pages visited during a visit to a website that led to a product inquiry.

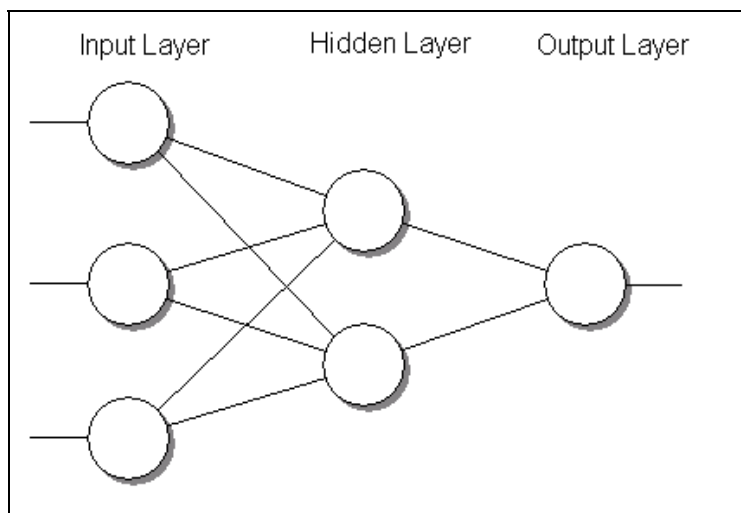
In the following sections we will briefly introduce some of these techniques. More detail will be given to machine learning techniques than to statistical techniques, since the latter methods are more likely to be familiar to you. It should be stressed at this stage that the power of Clementine is that models can be built and results obtained without having to deeply understand the various techniques. We will, therefore, not be describing in great detail how each of the different methods works – just a brief overview of what they are capable of and when to use them.

Neural Networks

Historically, neural networks attempted to solve problems in a way modeled on how the brain operates. Today they are generally viewed as powerful modeling techniques.

A typical neural network consists of several neurons arranged in layers to create a network. Each neuron can be thought of as a processing element that is given a simple part of a task. The connections between the neurons provide the network with the ability to learn patterns and interrelationships in data. The figure below gives a simple representation of a neural network (a multi-layer perceptron).

Figure 7.1 Simple Representation of a Common Neural Network



When using neural networks to perform predictive modeling, the input layer contains all of the fields used to predict the outcome. The output layer contains an output field: the target of the prediction. The input and output fields can be numeric or symbolic (in Clementine, symbolic fields are transformed into a numeric form (dummy or binary set encoding) before processing by the network). The hidden layer contains a number of neurons at which outputs from the previous layer combine. A network can have any number of hidden layers, although these are usually kept to a minimum. All neurons in one layer of the network are connected to all neurons within the next layer.

While the neural network is learning the relationships between the data and results, it is said to be training. Once fully trained, the network can be given new, unseen data and can make a decision or prediction based upon its experience.

When trying to understand how a neural network learns, let us think of how a parent teaches a child how to read. Patterns of letters are presented to the child and the child makes an attempt at the word. If the child is correct she is rewarded and the next time she sees the same combination of letters she is likely to remember the correct response. However, if she is incorrect, then she is told the correct response and tries to adjust her response based on this feedback. Neural networks work in the same way.

Clementine provides two different classes of supervised neural networks, the Multi-Layer Perceptron (MLP) and the Radial Basis Function Network (RBFN). In this course we will concentrate on the MLP type network and the reader is referred to the *Clementine User's Guide* and the *Advanced Modeling with Clementine* training course for more details on the RBFN approach to neural networks.

Within a MLP, each hidden layer neuron receives an input based on a weighted combination of the outputs of the neurons in the previous layer. The neurons within the final hidden layer are, in turn, combined to

produce an output. This predicted value is then compared to the correct output and the difference between the two values (the error) is fed back into the network, which in turn is updated. This feeding of the error back through the network is referred to as back-propagation.

To illustrate this process we will take the simple example of a child learning the difference between an apple and a pear. The child may decide in making a decision that the most useful factors are the shape, the color and the size of the fruit – these are the inputs. When shown the first example of a fruit she may look at the fruit and decide that it is round, red in color and of a particular size. Not knowing what an apple or a pear actually looks like, the child may decide to place equal importance on each of these factors – the importance is what a network refers to as weights. At this stage the child is most likely to randomly choose either an apple or a pear for her prediction.

On being told the correct response, the child will increase or decrease the relative importance of each of the factors to improve their decision (reduce the error). In a similar fashion a MLP begins with random weights placed on each of the inputs. On being told the correct response, the network adjusts these internal weights. In time, the child and the network will hopefully make correct predictions. Neural networks are examined in Chapter 8.

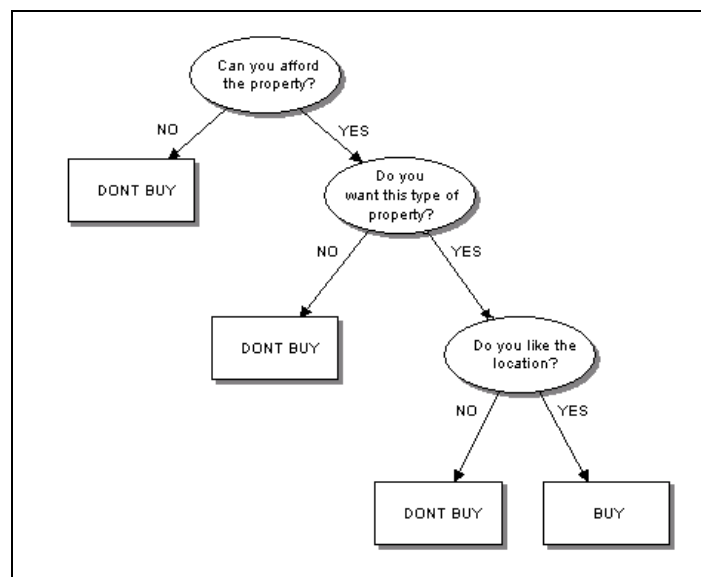
Rule Induction

A common complaint with neural networks is that they are “black box” techniques; that is, it is very difficult to work out the reasoning behind their predictions. Rule Induction is a complementary technique in the sense that it does not suffer this problem.

Clementine contains two different rule induction (also called decision tree) algorithms: C5.0 and C&R Tree (classification and regression tree). Both derive a decision tree of rules that try to describe distinct segments within the data in relation to an outcome or output field. The tree’s structure openly shows the rule’s reasoning and can therefore be used to understand the decision-making process that drives a particular outcome. Some differences between the two algorithms will be given in Chapter 9.

To help understand rule induction, let us think about making a decision to buy a house. The most important factor may be cost – can you afford the property? The second may be what type of property are you looking for – a house or a condo? The next consideration may be the location of the property... etc.

Figure 7.2 Graphical Representation of a Decision Tree



Another advantage of rule induction methods over neural networks is that the process automatically eliminates any fields that are not important in making decisions, while most neural networks include all inputs. This provides you with useful information and can even be used to reduce the number of fields entering a neural net.

The C5.0 rule induction method in Clementine allows you to view the rules in two different formats: the decision tree presentation, which is useful if the user wants to visualize how the predictor fields split the data into subsets, and the rule set presentation which breaks the tree into collections of “IF – THEN” rules, organized by outcome. The latter is useful if we wish to see how particular groups of input values relate to one value of the outcome. The two available rule induction algorithms are discussed, and C5.0 is demonstrated, in Chapter 9.

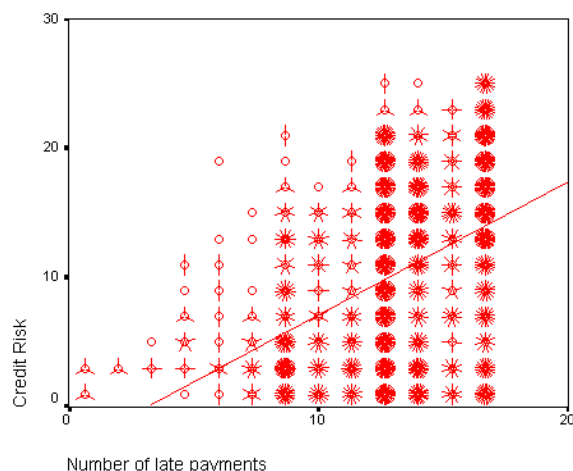
Statistical Prediction Models

Linear regression and logistic regression, the statistical modeling procedures within Clementine, make stronger data assumptions (linear model, normality of errors for regression; linear model in log-odds form, binomial or multinomial distribution of outcome) than do machine learning techniques. Models can be expressed using simple equations, aiding interpretation, and statistical tests can guide field selection in the model. In Clementine, both procedures have stepwise options that can automate input field selection when building models. They are not as capable, at least in standard form, as neural networks in capturing complex interactions among inputs and nonlinear relations.

Linear Regression

Linear regression is a method familiar to just about everyone these days. It is the classic general linear model technique and is used to predict a numeric outcome field with a set of predictors that are also numeric. However, symbolic input fields can be included by creating dummy-coded forms of these fields. Linear regression assumes that the data can be modeled with a linear relationship. The figure below presents a scatter plot depicting the relationship between the number of previous late payments for bills and the credit risk of defaulting on a new loan. Superimposed on the plot is the best-fit regression line.

Figure 7.3 Linear Regression Line Superimposed on Plot



Although there is a lot of variation around the regression line, it is clear that there is a trend in the data such that more late payments are associated with a greater credit risk. Of course, linear regression is normally used with several predictors; this makes it impossible to display the complete solution with all predictors in convenient graphical form. Thus most users of linear regression focus on the statistical output.

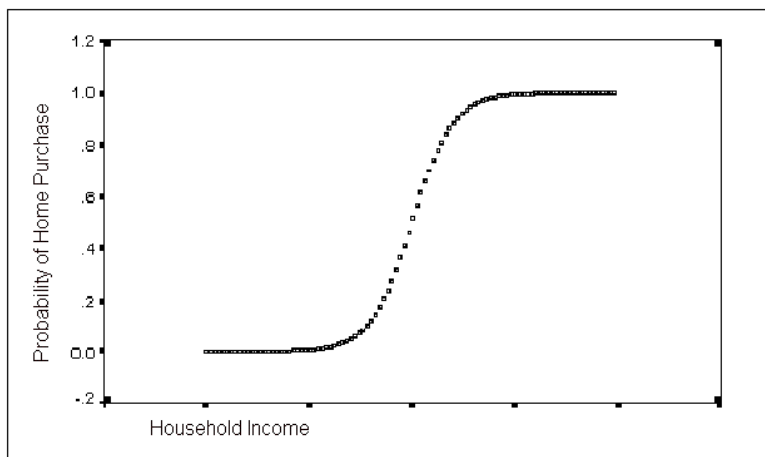
Linear regression modeling runs relatively quickly (single pass through the data). It is supported by statistical tests and goodness-of-fit measures. Since the final model is in the form of a single linear equation, model interpretation is straightforward.

Logistic Regression

Logistic or multinomial regression attempts to predict a symbolic outcome field. It is similar to linear regression in that it uses the general linear model as its theoretical underpinning, and so calculates regression coefficients and tries to fit cases to a line, although not a straight one. A common application would be predicting whether or not someone renews an insurance policy.

Logistic regression actually predicts a continuous function that represents the probability associated with being in a particular outcome category. This is shown in the figure below, which presents the two-category outcome case. It displays the predicted relationship between household income and the probability of purchase of a home. The S-shaped curve is the logistic curve, hence the name for this technique. The idea is that at low income, the probability of purchasing a home is small and rises only slightly with increasing income. But at a certain point, the chance of buying a home begins to increase in almost a linear fashion, until eventually most people with substantial incomes have bought homes, at which point the function levels off again. Thus the outcome variable varies from 0 to 1 because it is measured in probability.

Figure 7.4 Logistic Function



After the procedure calculates the predicted outcome probability, it simply assigns a record to a predicted outcome category based on whether its probability is above .50 or not. An extension of this approach is used when the outcome field has three or more values.

As with linear regression, logistic regression produces regression coefficients and associated statistics. Input fields can be dropped from the model based on statistical tests. The logistic regression coefficients can be related to the predicted odds of the target outcome category. This type of information is very powerful for decision-making. Although interaction effects (effects involving combinations of input fields) can be explicitly built into logistic regression models, they are not usually included, so logistic regression procedures, like linear regression procedures, are less likely to fit complex data sets than neural network or rule induction techniques.

Principal Components

Principal components analysis and factor analysis are data reduction techniques that can be used prior to predictive modeling and, less so, to clustering. Principal components can replace sets of highly correlated numeric fields with a smaller number of uncorrelated fields that are linear combinations of the original fields. It is more likely to accompany statistical than machine learning methods, and is often used in analyses involving survey data with many rating scale fields. For more information, see the Clementine manuals or the *Advanced Modeling with Clementine* training course.

Clustering

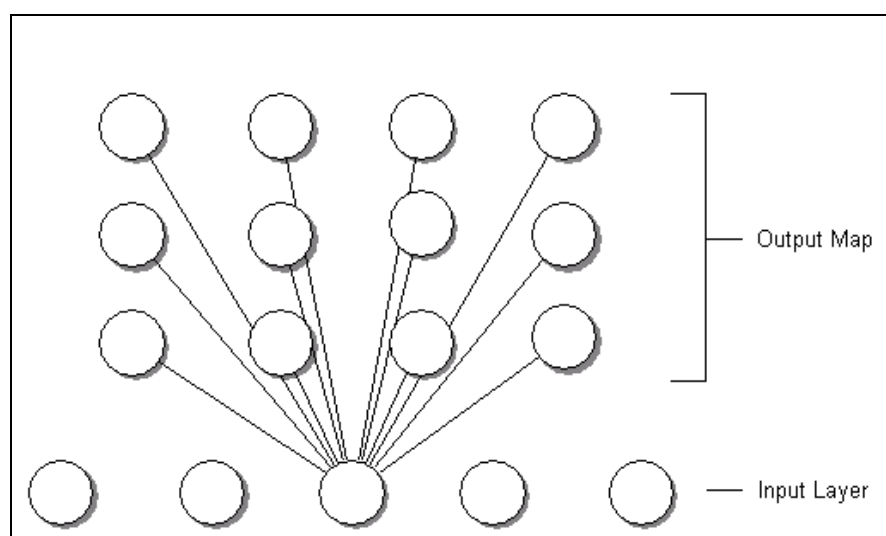
Clustering methods help discover groups of data records with similar values or patterns. These techniques are used in marketing (customer segmentation) and other business applications (records that fall into single-record clusters may contain errors or be instances of fraud). Clustering is sometimes performed prior to predictive modeling. In such instances, the customer groups might be modeled individually (taking the approach that each cluster is unique) or the cluster group might be an additional input to the model. Clementine offers three clustering methods: Kohonen networks, k-means clustering, and two-step clustering.

Kohonen Networks

A Kohonen network is a type of neural network that performs unsupervised learning: that is, it has no output or outcome to predict. Such networks are used to cluster or segment data, based on the patterns of input fields. Kohonen networks make the basic assumption that clusters are formed from patterns that share similar features and will therefore group similar patterns together.

Kohonen networks are usually one- or two-dimensional grids or arrays of artificial neurons. Each neuron is connected to each of the inputs (input fields), and again weights are placed on each of these connections. The weights for a neuron represent a profile for that cluster on the fields used in the analysis. There is no actual output layer in Kohonen networks, although the Kohonen map containing the neurons can be thought of as an output. The Figure below shows a simple representation of an output grid or Kohonen map.

Figure 7.5 Basic Representation of a Kohonen Network



Note that the connections from the input neuron layer are shown for only one neuron.

When a record is presented to the grid, its pattern of inputs is compared with those of the artificial neurons within the grid. The artificial neuron with the pattern most like that of the input “wins” the input. This causes the weights of the artificial neuron to change to make it appear even more like the input pattern. The Kohonen network also slightly adjusts the weights of those artificial neurons surrounding the one with the pattern that wins the input.

This has the effect of moving the most similar neuron and the surrounding nodes, to a lesser degree, to the position of the record in the input data space. The result, after the data have passed through the network a number of times, will be a map containing clusters of records corresponding to different types of patterns within the data. Kohonen networks will be examined in Chapter 11.

K-Means Clustering

K-means clustering is a relatively quick method for exploring clusters in data. The user sets the number of clusters (*k*) to be fit and the procedure selects *k* well-spaced data records as starting clusters. Each data record is then assigned to the nearest of the *k* clusters. The cluster centers (means on the fields used in the clustering) are updated to accommodate the new members. Additional data passes are made as needed; as the cluster centers shift, a data record may need to be moved to its now nearest cluster.

Since the user must set the number of clusters, this procedure is typically run several times, assessing the results (mean profiles, number of records in each cluster, cluster separation) for different numbers of clusters (values of *k*).

Two-Step Clustering

Unlike the previous cluster methods discussed, two-step clustering will automatically select the number of clusters. The user specifies a range (Minimum (*Min*) and Maximum (*Max*) for the number of clusters. In the first step, all records are classified into pre-clusters. These pre-clusters are designed to be well separated. In the second step, a hierarchical agglomerative cluster method (meaning that once records are joined together to create clusters, they are never split apart) is used to successively combine the pre-clusters. This produces a set of cluster solutions containing from *Max* clusters down to *Min* clusters. A criterion (likelihood-based) is then used to decide which of these solutions is best.

The two-step clustering method thus has the advantage of automatically selecting the number of clusters (within the range specified) and does not require enormous machine resources (since only the *Max* clusters, not all records, are used in the second step).

Association Rules

Association rule procedures search for things (events, purchases, attributes) that typically occur together in the data. Association rule algorithms automatically find the patterns in data that you could manually find using visualization techniques such as the web node, but with much greater speed and they can explore more complex patterns.

The rules found associate a particular outcome category (called a conclusion) with a set of conditions. The outcome fields may vary from rule to rule and as a result the user does not often focus on one particular output field. In fact, the advantage of these algorithms over rule induction is that associations can exist between any of the fields. One disadvantage to rule associations is that they attempt to find patterns in what is potentially a very large search space, and can be slow in running.

The two algorithms, provided by Clementine, to generate association rules are called Apriori and GRI, and the differences between these two methods will be discussed in Chapter 12.

The algorithms begin by generating a set of extremely simple rules. These rules are then specialized by adding more refined conditions to them (making the rules more complex) and the most interesting rules are stored.

Clementine allows certain restrictions on the algorithms to help speed up the process such as limiting the number of possible conjuncts within a rule. The result is a set of rules that can be viewed but cannot be used directly for predicting. Association rules will be demonstrated in Chapter 12.

Sequence Detection

Sequence detection methods search for sequential patterns in time-structured data. Their focus on time-ordered sequences, rather than general association, is what distinguishes them from the association rule methods discussed earlier. In such analyses there may be interest in identifying common sequence patterns or in finding sequential patterns that often lead to a particular conclusion (for example, a purchase on a web-site, or a request for additional information).

Application areas of sequence detection include retail shopping, web log analysis, and process improvement (for example, finding common sequences in the steps taken to resolve problems with electronic devices).

Sequence detection is applied to symbolic fields (categories) and if numeric fields are input, their values will be treated as symbolic. That is, a field that takes the values from 1 to 100 would be treated as having one hundred categories.

The Sequence node in Clementine uses the CARMA algorithm, which makes only two passes through the data. It can also generate nodes that make predictions based on specific sequences. The CAPRI algorithm add-on, which uses a different algorithm, contains pruning options that give you greater flexibility in specifying the types of sequences of interest (for example, subsequences that appear within reported sequences can be suppressed (pruned); only sequences that end in a specified event can be reported; should repeating patterns be allowed within a larger sequence?). While CaprI displays the common sequences, it does not generate nodes that produce predictions. For more details, see the *CaprI User Guide*. The Sequence node will be explored in Chapter 13.

Which Technique, When?

Apart from the basics, this is a very difficult question to answer. Obviously if you have a clear field in the data you want to predict, then any of the supervised learning techniques or one of the statistical modeling methods (depending on the output field's type) will perform the task, with varying degrees of success. If you want to find groups of individuals that behave similarly on a number of fields in the data, then any of the clustering methods is appropriate. Association rules are not going to directly give you the ability to predict, but are extremely useful as a tool for understanding the various patterns within the data. If there is interest in sequential patterns in data, then sequence detection methods are the techniques of choice and some of them can be used to generate predictions.

But if you want to go further and decide which particular predicting technique will work better, then unfortunately the answer is that it depends on the particular data you are working on. In fact, more accurately, it depends on the particular fields you want to predict and how they are related to the various inputs. There are suggested guidelines as to when one technique may work better than another, and we will mention these in the following chapters, but these are only suggestions and not rules. They will be broken on many occasions!

The advantage of Clementine is the simplicity of building the models. Neural networks, rule induction (decision trees) and regression models can be built with great ease and speed, and their results compared. You must remember that data mining is an iterative process; models will be built, broken down, and often even combined before the business user is happy with the results.

One final yet important point to keep in mind when building models is that Clementine will only find rules or patterns in data if they exist. You cannot extract a model with high predicting accuracy if no associations between the input fields and output field exist.

Summary

In this chapter you have been introduced to a number of the machine learning and statistical modeling capabilities of Clementine. You should now have an understanding of the different types of analyses you can perform and the different algorithms that can help you achieve your desired outcome. In the next chapter we will describe how to build a neural network within Clementine.

Chapter 8

Neural Networks

Overview

- Introduce the Neural Net node
- Build a neural network
- Introduce the generated Models palette
- Browse and interpret the results
- Evaluate the model

Objectives

In this chapter we introduce how to build a neural network with Clementine. The resulting model will be browsed and the output interpreted.

Data

Throughout the session we will continue using the credit risk data introduced in the previous chapters with the aim to build a model that predicts the credit risk field. Following recommended practice, the data file has been split into two (text) files—*RiskTrain.txt*, which will be used to build the model, and *RiskValidate.txt*, a holdout sample, which will be used later on in the course.

In fact, this idea of dividing the data into two sections, one to build the model and one to test the model (often called a holdout or validation sample), is a common practice in data mining. With a holdout sample, you are able to check the resulting model performance on data not used fitting the model. This holdout data sample has the known outcome field and therefore can be used to check model performance.

Introduction

In this section we will introduce the Neural Net node that builds a neural network. In the main, the default algorithm and settings will be used. The *Clementine User's Guide* contains details on alternative algorithms and expert options, and these topics are covered in the *Advanced Modeling with Clementine* training course.

The Neural Network Node

The Neural Net node is used to create a neural network and can be found in the Modeling palette. Once trained, a Generated Net node labeled with the name of the predicted field will be appear in the Generated Models palette. This node represents the trained neural network. Its properties can be browsed and new data can be passed through this node to generate predictions. We will investigate the properties of the trained network node later in this chapter.

Before a data stream can be used by the Neural Net, or any node in the Modeling palette, field types must be defined (either in the source node or a Type node). This is because it is within these nodes that the type and direction of each field is set and this information is used by all modeling nodes. As a reminder, the table below shows the four available direction definitions.

Table 8.1 Direction Settings

IN	The field acts as an input or predictor within the modeling
OUT	The field is the output or target for the modeling
BOTH	Allows the field to be act as both an input and an output in modeling. Direction suitable for the association rule and sequence detection algorithms only, all other modeling techniques will ignore the field.
NONE	The field will not be used in machine learning or statistical modeling. Default if the field is defined as Typeless.

Direction can be set by clicking in the Direction column for a field within the Type node or the Type tab of a source node and selecting the direction from the drop-down menu. Alternatively, this can be done from the Fields tab of a modeling node.

If the Stream Canvas is not empty, click **File..New Stream**
 Place a **Var. File** node from the Sources palette
 Double-click the **Var. File** node
 Move to the **c:\Train\ClemIntro** directory and double-click on the **RiskTrain.txt** file
 Click, if not already checked, the **Read field names from file** check box
 As delimiter, check the **Tab** option
 Set the **Strip lead and trail spaces** option to **Both**
 Click **OK** to return to the Stream Canvas
 Place a **Type** node from the Field Ops palette to the right of the Var. File node named **RiskTrain.txt**
 Connect the Var. File node named **RiskTrain.txt** to the **Type** node

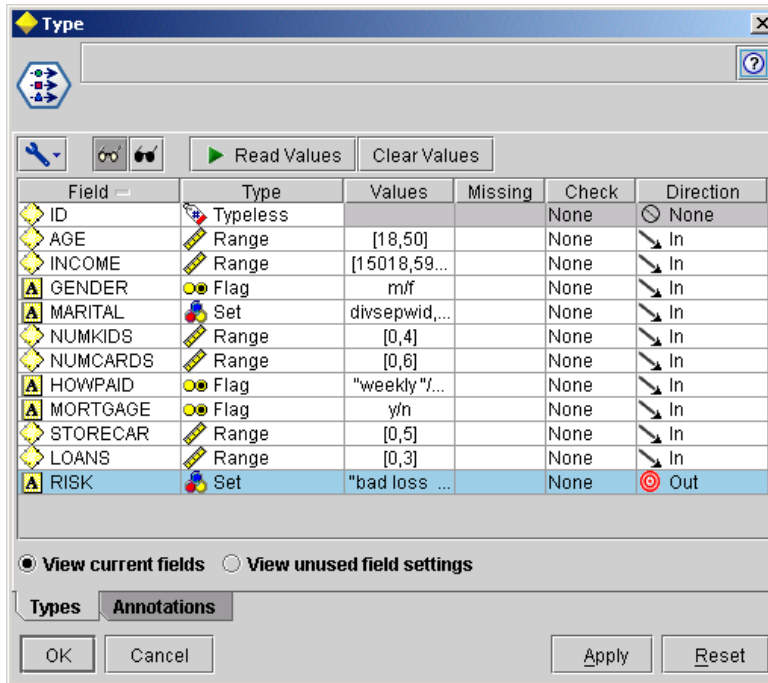
Next we will add a Table node to the stream. This not only will force Clementine to autotype the data but also will act as a check to ensure that the data file is being correctly read.

Place a **Table** node from the Output palette above the **Type** node in the Stream Canvas
 Connect the **Type** node to the **Table** node
 Right-click the **Table** node
Execute the **Table** node

The values in the data table look reasonable (not shown).

Click **File..Close** to close the Table window
 Double-click the **Type** node
 Click in the cell located in the **Type** column for **ID** (current value is **Range**), and select **Typeless** from the list
 Click in the cell located in the **Direction** column for **Risk** (current value is **In**) and select **Out** from the list

Figure 8.1 Type Node Ready for Modeling

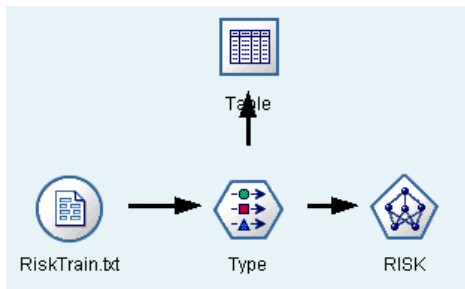


Notice, that ID will be excluded from any modeling as the direction is automatically set to *None* for a Typeless field. The RISK field will be the output field for any predictive model and all fields but ID will be used as predictors.

Click **OK**

Place a **Neural Net** node from the **Modeling** palette to the right of the **Type** node
Connect the **Type** node to the **Neural Net** node

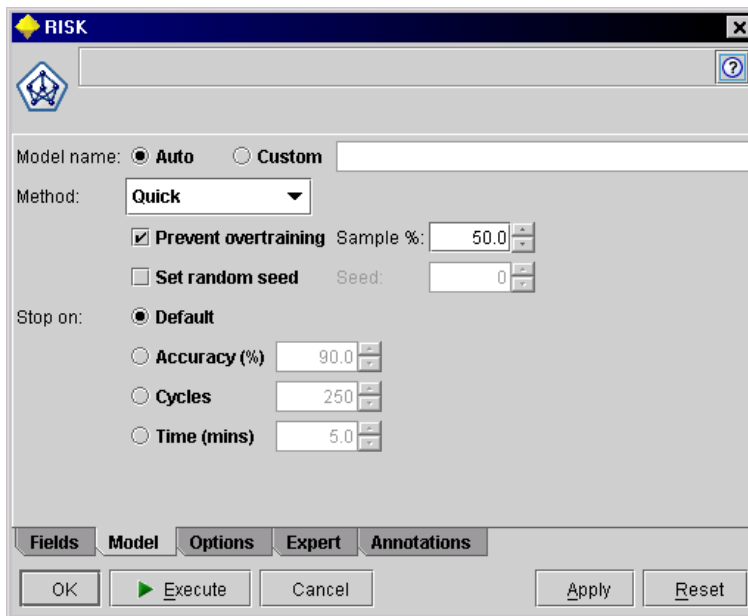
Figure 8.2 Neural Net Node (RISK) Added to Data Stream



Notice that once the Neural Net node is added to the data stream, its name becomes RISK, the field we wish to predict. The name can be changed (among other things), by editing the Neural Net node.

Double-click the **Neural Net** node

Figure 8.3 Neural Net Dialog



The name of the Network, which, by default, will also be used as the name for the Neural Net and the generated model node, can be entered in the Model name Custom text box.

There are five different algorithms available within the Neural Net node. Within this course we will stay with the default *Quick* method. The *Quick* method will use a feed-forward back-propagation network whose topology (number and configuration of nodes in the hidden layer) is based on the number and types of the input and output fields. For details on the other neural network methods, the reader is referred to the *Clementine User's Guide*, or the *Advanced Modeling with Clementine* training courses.

Over-training is one of the problems that can occur within neural networks. As the data pass repeatedly through the network, it is possible for the network to learn patterns that exist in the sample only and thus over-train. That is, it will become too specific to the training sample data and lose its ability to generalize. By selecting the *Prevent overtraining* option, only a randomly selected proportion of the training data is used to train the network. Once this proportion of data has made a complete pass through the network, the rest is reserved as a test set to evaluate the performance of the current network. By default, this information determines when to stop training and provides feedback information. *We advise you to leave this option turned on.*

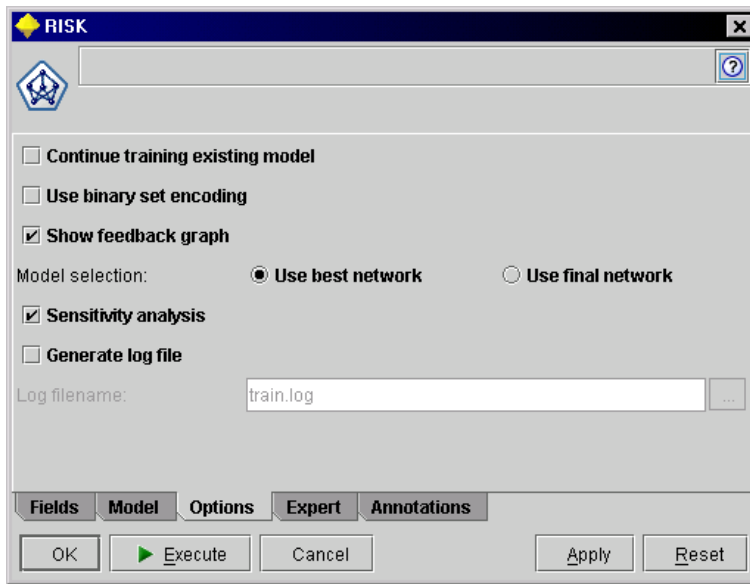
You can control how Clementine decides to stop training a network. By default, Clementine stops when it appears to have reached its optimally trained state; that is, when accuracy in the test data set seems to no longer improve. Alternatively, you can set a required accuracy value, a limit to the number of cycles through the data, or a time limit in minutes. In this chapter we use the default option.

Since the neural network initiates itself with random weights, the behavior of the network can be reproduced using the *Set random seed* option and entering the same seed value. Although we do it here to reproduce the results in the guide, setting the random seed is not a normal practice and it is advisable to run several trials on a neural network to ensure that you obtain similar results using different random starting points.

The Options tab allows you to customize some settings:

Click the **Options** tab

Figure 8.4 Neural Net: Options



The *Use binary set encoding* option uses an alternative method of coding fields of set type when they are used in the Neural Net node. It is more efficient and thus can have benefits when set type fields included in the model have a large number of values.

A feedback graph appears while the network is training and gives information on the current accuracy of the network. We will describe the feedback graph in more detail later.

By default, a model will be generated from the best network found (based on the test data), but there is an option to generate a model from the final network trained. This can be used if you wish to stop the network at different points to examine intermediate results, and then pick up network training from where it left off.

Sensitivity analysis provides a measure of relative importance for each of the fields used as inputs to the network and is helpful in evaluating the predictors. We will retain this option as well.

The Expert tab allows you to refine the properties (for example, the network topology and training parameters) of the training method. Expert options are detailed in the *Clementine User's Guide* and reviewed in the *Advanced Modeling with Clementine* training course.

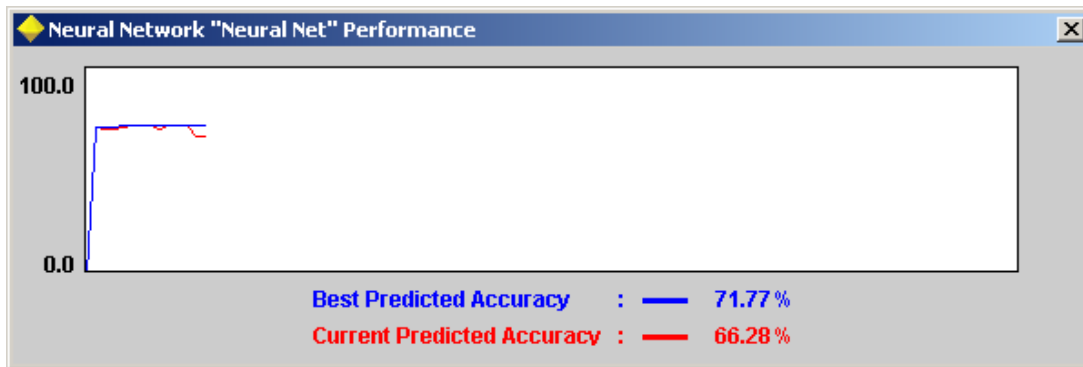
In this chapter we will stay with the default settings on the majority of the above options.

To reproduce the results in this training guide:


- Click the **Model** tab
- Click the **Set random seed** check box
- Type **233** into the **Seed:** text box
- Click **Execute**

Note that if different models are built from the same stream using different inputs, it may be advisable to change the Neural Net node names for clarity.

Figure 8.5 Feedback Graph During Network Training

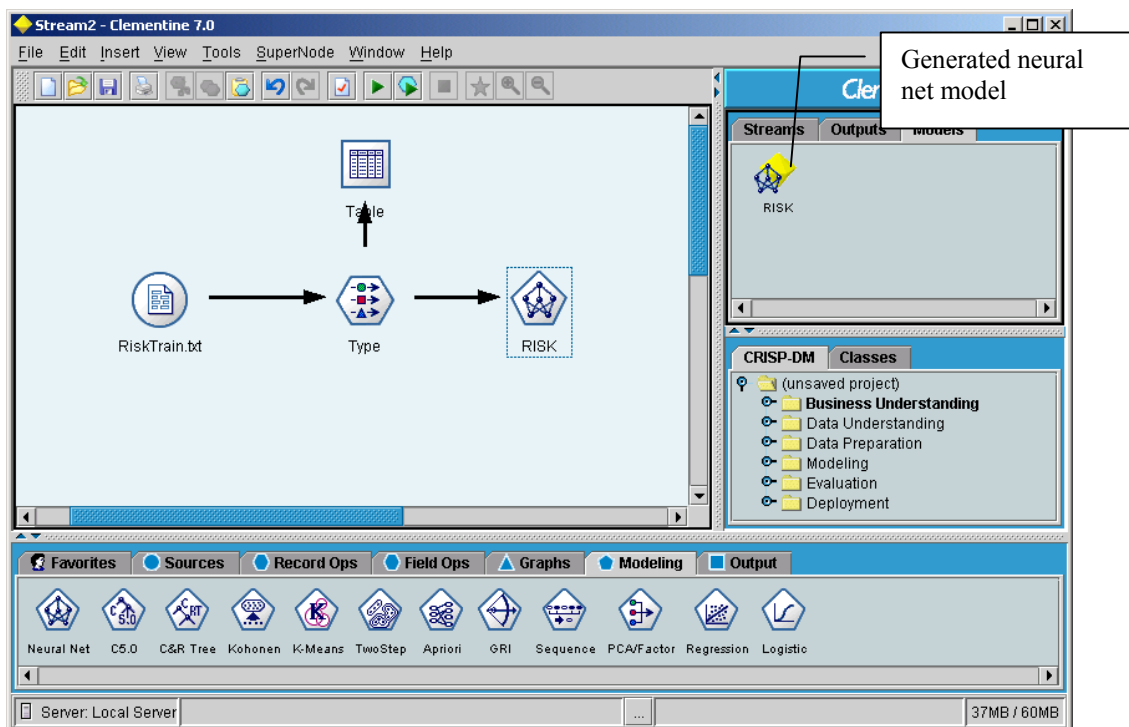


Clementine passes the data stream to the Neural Net node and begins to train the network. A feedback graph similar to the one shown above appears on the screen. The graph contains two lines. The red, more irregular line labeled *Current Predicted Accuracy*, presents the accuracy of the current network in predicting the test data set. The blue, smoother line, labeled *Best Predicted Accuracy*, displays the best accuracy so far on the test data.

Training can be paused by clicking the *Stop execution* button  in the Toolbar (this button can be found next to the Execute buttons).

Once trained the network performs, if requested, the sensitivity analysis and a diamond-shaped node appears in the Models palette. This represents the trained network and is labeled with the network name.

Figure 8.6 Generated Neural Net Node Appearing in Models Palette

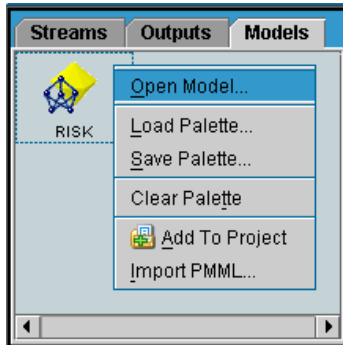


Models Palette

The Models tab in the Manager holds and manages the results of the machine learning and statistical modeling operations. There are two context menus available within the Models palette. The first menu applies to the entire model palette.

Right-click in the background (empty area) in the **Models** palette

Figure 8.7 Context Menu in the Models Palette

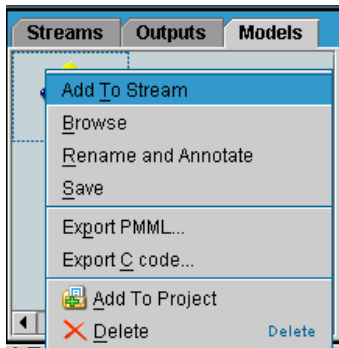


This menu allows you to save the models palette and its contents, open a previously saved models palette, clear the contents of the palette or to add the generated models to the Modeling section of the CRISP-DM project window.

The second menu is specific to the generated model nodes.

Right-click the generated **Neural Net** node named **RISK** in the Models palette

Figure 8.8 Context Menu for Nodes in the Models Palette



This menu allows you to rename, annotate, and browse the generated model node. A generated model node can be deleted, exported in either PMML (Predictive Model Markup Language) or C code, or saved for future use. The most important menu option is to browse the model:

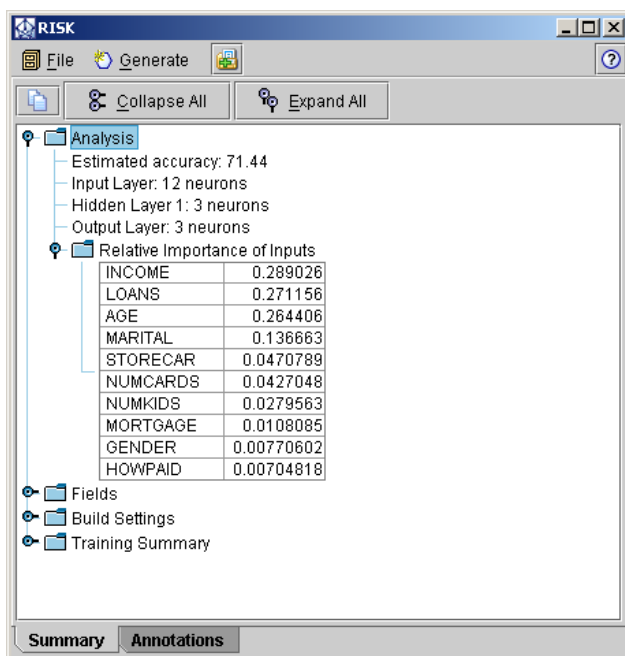
Click **Browse**

For more information on a section simply expand the section by double-clicking the section (or click the *Expand All* button to expand all sections at once).

To start with we will take a closer look at the *Analysis* section.

Expand the **Relative Importance of Inputs** folder

Figure 8.9 Browsing the Generated Net Node



The Analysis section displays information about the neural network. The predicted accuracy for this neural network is 71.44%, indicating the proportion of the test set correctly predicted. The input layer is made up of one neuron per numeric or flag type field. Set type fields will have one neuron per value within the set (unless binary encoding is used). In this example, there are nine numeric or flag fields and one set field with three values, totaling twelve neurons. In this network there is one hidden layer, containing three neurons, and the output layer contains three neurons corresponding to the three values of the output field, RISK. If the output field had been defined as numeric then the output layer would only contain one neuron.

The input fields are listed in descending order of relative importance. Importance values can range from 0.0 and 1.0, where 0.0 indicates unimportant and 1.0 indicates extremely important. In practice this figure rarely goes above 0.35. Here we see that INCOME is the most important field within this current network, closely followed by LOANS and AGE.

The sections *Fields*, *Build Settings* and *Training Summary* contain technical details and we will skip these sections.

Click **File..Close** to close the Neural Net output window

Understanding the Neural Network

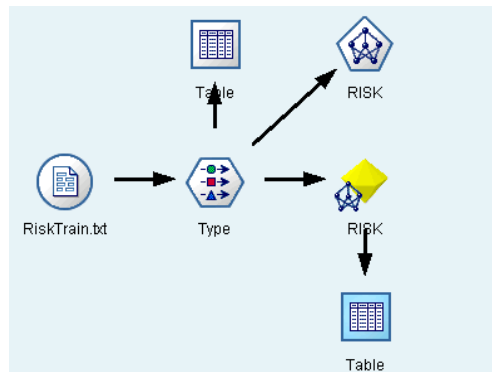
A common criticism of neural networks is that they are opaque: that is, once built, the reasoning behind their predictions is not clear. In the following sections we will use some of the techniques learned in earlier chapters to help you evaluate the network and understand it at a simplistic level.

Creating a Data Table Containing Predicted Values

Generated model nodes can be placed on the Stream Canvas and treated in the same way as the other operational nodes within Clementine; that is, the data can be passed through them and they perform an operation (adding model-based fields to the stream).

- Move the **Neural Net** node named **RISK** higher in the Stream Canvas
- Place the **generated Neural Net** node named **RISK** from the Models palette to the right of the **Type** node
- Connect the **Type** node to the **generated Neural Net** node named **RISK**
- Place a **Table** node below the **generated Neural Net** node named **RISK**
- Connect the **generated Neural Net** node named **RISK** to the **Table** node

Figure 8.10 Placing a Generated Model on the Stream Canvas



Execute the **Table** node

Figure 8.11 Table Showing the Two Fields Created by the Generated Net Node

	MORTGAGE	STORECAR	LOANS	RISK	\$N-RISK	\$NC-RISK
1	y	2	0	good risk	good risk	0.301
2	y	1	0	bad loss	good risk	0.302
3	y	1	1	bad loss	good risk	0.302
4	y	1	0	good risk	good risk	0.302
5	y	2	0	good risk	good risk	0.302
6	y	1	1	good risk	good risk	0.301
7	y	2	1	bad loss	good risk	0.301
8	y	2	1	good risk	good risk	0.301
9	y	1	1	bad profit	good risk	0.302
10	y	1	0	bad profit	good risk	0.302
11	y	1	0	bad loss	good risk	0.302
12	y	2	0	good risk	good risk	0.302

The generated Neural Net node calculates two new fields, \$N-RISK and \$NC-RISK, for every record in the data file. The first represents the predicted RISK value and the second a confidence value for the prediction. The latter is only appropriate for symbolic outputs and will be in the range of 0.0 to 1.0, with the more confident predictions having values closer to 1.0.

Close the **Table** window

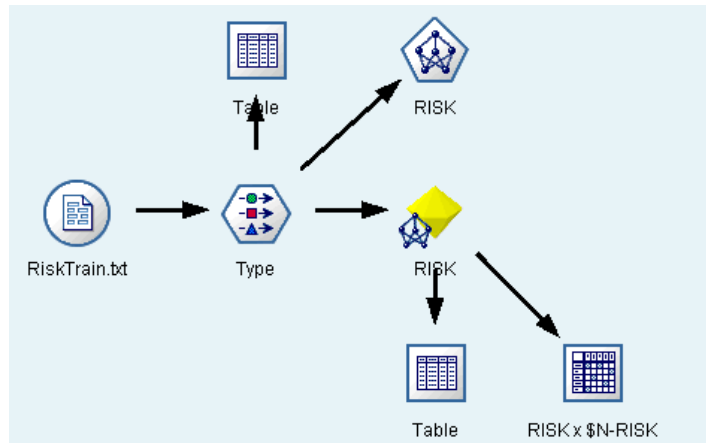
Comparing Predicted to Actual Values

When predicting a symbolic field, it is valuable to produce a data matrix of the predicted values (\$N-RISK) and the actual values (RISK) in order to study how they compare and where the differences are.

Place a **Matrix** node from the **Output** palette to the lower right of the **generated Neural Net** node named **RISK**

Connect the **generated Neural Net** node named **RISK** to the **Matrix** node

Figure 8.12 Comparing Predicted to Actual Values Using a Matrix Node



Double-click the **Matrix** node
 Put **RISK** in the **Rows:**
 Put **\$N-RISK** in the **Columns:**
 Click the **Appearance** tab
 Click the **Percentage of row** option

The *Percentage of row* choice will show us, for each actual risk category, the percentage of records predicted into each of the outcome categories.

Click **Execute**

Figure 8.13 Matrix of Actual (Rows) and Predicted (Columns) Credit Risk

		\$N-RISK		
RISK		bad loss	bad profit	good risk
bad loss	Count	174	291	94
	Row %	31.127	52.057	16.816
bad profit	Count	22	1330	123
	Row %	1.492	90.169	8.339
good risk	Count	5	152	264
	Row %	1.188	36.105	62.708

Cells contain: cross-tabulation of fields

Matrix Appearance Annotations

The model is predicting over 90% of the bad but profitable risk types correctly but only 31% of bad loss generating risk types, with the good risk types falling in the middle of these two (62%). If we wanted to correctly predict bad but profitable types at the expense of the other types this would appear to be a reasonable model. On the other hand, if we wanted to predict those credit risks that were going to cause the company a loss, this model would only predict correctly in about one third of the instances.

Close the **Matrix** window

Evaluation Chart Node

The Evaluation Chart node offers an easy way to evaluate and compare predictive models in order to choose the best model for your application. Evaluation charts show how models perform in predicting particular outcomes. They work by sorting records based on the predicted value and confidence of the prediction, splitting the records into groups of equal size (quantiles), and then plotting the value of the business criterion for each quantile, from highest to lowest.

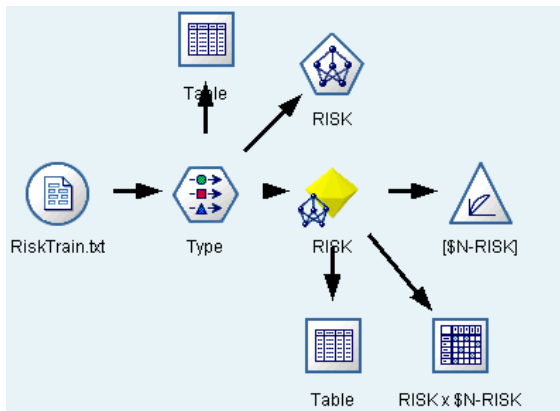
Outcomes are handled by defining a specific value or range of values as a hit. Hits usually indicate success of some sort (such as a sale to a customer) or an event of interest (such as someone given credit being a good credit risk). Flag output fields are straightforward; by default, hits correspond to *true* values. For Set output fields, by default the first value in the set defines a hit. For the credit risk data, the first value for the RISK field is bad loss. To specify a different value as the hit value, use the Options tab of the Evaluation node to specify the target value in the *User defined hit* group. There are five types of evaluation charts, each of which emphasizes a different evaluation criterion. Here we discuss Gains and Lift charts. For information about the others, which include Profit and ROI charts, see the *Clementine User's Guide*.

Gains are defined as the proportion of total hits that occurs in each quantile. We will examine the gains when the data are ordered from those most likely to those least likely to be in the bad loss category (based on the model predictions).

Place an **Evaluation** node from the Graphs palette near the **generated Neural Net** node named **RISK**

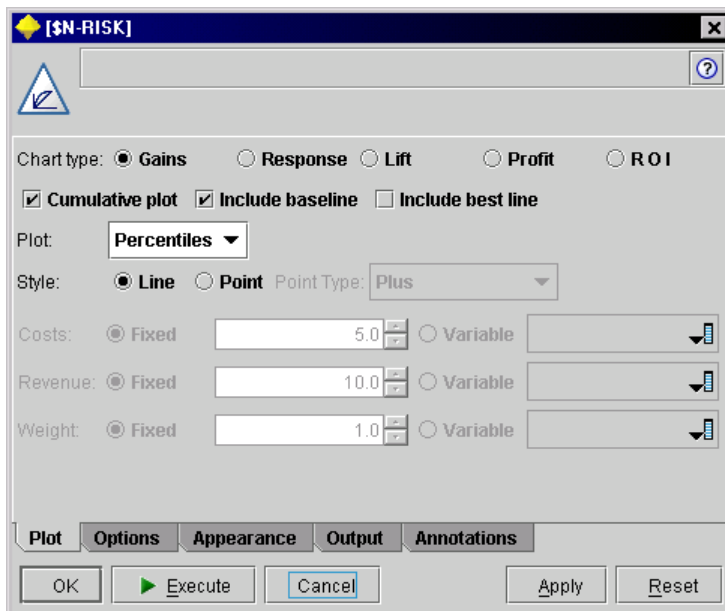
Connect the **generated Neural Net** node named **RISK** to the **Evaluation** node

Figure 8.14 Stream with Evaluation Node Connected to a Generated Model Node



Double-click the **Evaluation** node

Figure 8.15 Evaluation Node Dialog Box

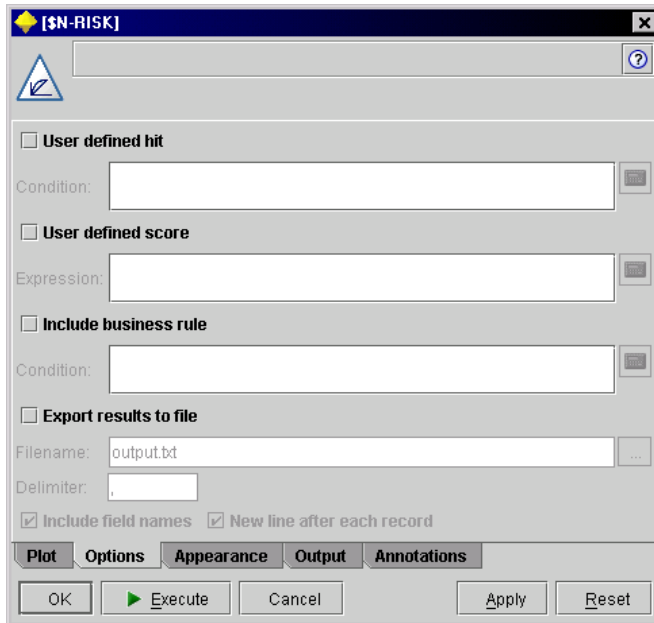


The *Chart Type* options support five chart types with *Gains* chart being the default. If *Profit* or *ROI* chart type is selected, then the appropriate options (cost, revenue and record weight values) become active so information can be entered. The charts are cumulative by default (see *Cumulative plot* check box), which is helpful in evaluating such business questions as “how will we do if we make the offer to the top X% of the prospects?” The granularity of the chart (number of points plotted) is controlled by the *Plot* drop-down list and the *Percentiles* choice will calculate 100 values (one for each percentile from 1 to 100). For small data files or business situations in which you can only contact customers in large blocks (say some number of groups, each representing 5% of customers, will be contacted through direct mail), the plot granularity might be decreased (to deciles (10 equal-sized groups) or vingtiles (20 equal-sized groups)).

A baseline is quite useful since it indicates what the business outcome value (here gains) would be if the model predicted at the chance level. The *Include best line* option will add a line corresponding to a perfect prediction model, representing the theoretically best possible result applied to the data.

Click the **Include best line** checkbox
Click **Options** tab

Figure 8.16 Evaluation Node Options Tab



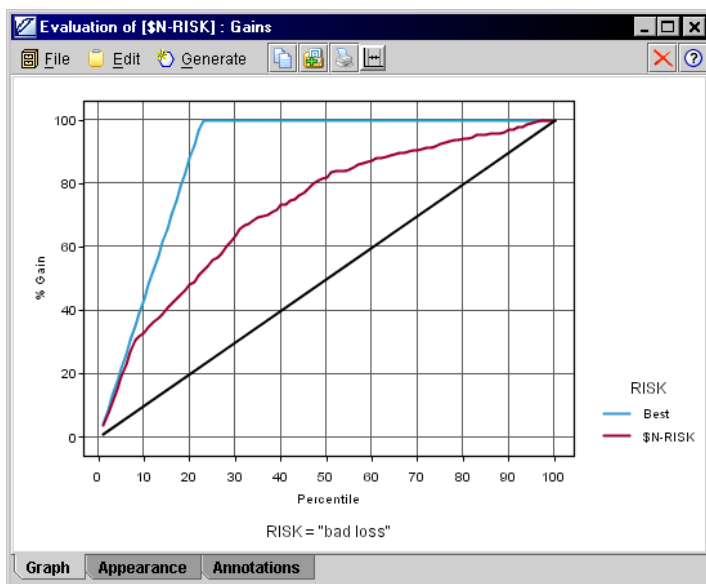
To change the definition of a hit, check the *User defined hit* check box and then enter the condition that defines a hit in the Condition box. For example, if we want the evaluation chart to be based on the good risks category, the condition would be `@TARGET = "good risk"`, where `@TARGET` represents the target fields from any models in the stream. The Expression Builder can be used to build the expression defining a hit. This tab also allows users to define how scores are calculated, which determines how the records are ordered in Evaluation charts. Typically scores are based on functions involving the predicted value and confidence.

The Include business rule option allows the Evaluation chart to be based only on records that conform to the business rule condition. So if you wanted to see how a model(s) performs for males in the southern part of the country, the business rule could be `REGION = "South" and SEX = "M"`.

The model evaluation results used to produce the evaluation chart can also be exported to a file (*Export results to file* option).

Click **Execute**

Figure 8.17 Gains Chart (Cumulative) with Bad Loss Credit Group as Target



The vertical axis of the gains chart is the cumulative percentage of the hits, while the horizontal axis represents the ordered (by model prediction and confidence) percentile groups. The diagonal line presents the base rate, that is, what we expect if the model is predicting the outcome at the chance level. The upper line (labeled *Best*) represents results if a perfect model were applied to the data, and the middle line (labeled *\$N-RISK*) displays the model results. The three lines connect at the extreme [(0, 0) and (100, 100)] points. This is because if either no records or all records are considered, the percentage of hits for the base rate, best model, and actual model are identical. The advantage of the model is reflected in the degree to which the model-based line exceeds the base-rate line for intermediate values in the plot and the area for model improvement is the discrepancy between the model line and the *Best* (perfect model) line. If the model line is steep for early percentiles, relative to the base rate, then the hits tend to concentrate in those percentile groups of data. At the practical level, this would mean for our data that many of the bad loss customers could be found within a small portion of the ordered sample. (You can create bands on an Evaluation chart [as we did earlier on a histogram] and generate a Select or Derive node for a band of business interest.)

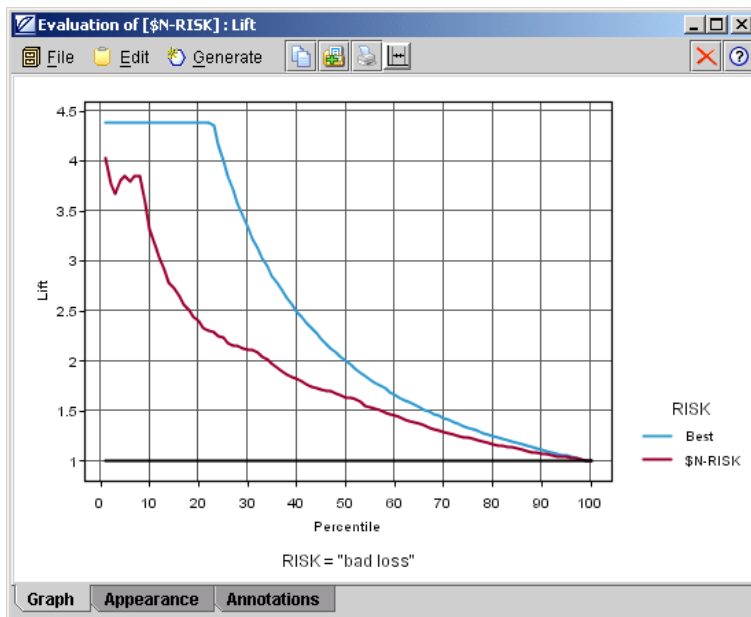
Examining the plot we see that across percentiles 1 through 20 or so, the distance between the model and baseline lines grows (indicating a concentration of bad loss individuals). If we look at the 20th percentile value (horizontal axis), we see that under the base rate we expect to find 20% of the hits (bad losses) in the first 20 percentiles (20%) of the sample, but the model produces over 40% of the hits in the first 20 percentiles of the model-ordered sample. The steeper the early part of the plot, the more successful is the model in predicting the target outcome. Notice that the line representing a perfect model (*Best*) continues with a steep increase between the 10% and 20% percentiles, while the results from the actual model flatten.

For the next forty percentiles (20 through 60), the distance between the model and base rate is fairly constant. This suggests that the hit percentage of bad losses for these model-based percentile groups does no better than the base-rate. Notice that the *Best* (perfect) model has already reached 100%. The gap between the model and base rate for the remaining percentiles (80 through 100) narrows, indicating that these last model-based percentile groups contain a relatively small (lower than the base rate) proportion of bad loss individuals. The Gains chart provides a way of visually evaluating how the model will do in predicting a specified outcome.

Another way of representing this information graphically, the lift chart plots a ratio of the percentage of records in each quantile that are hits divided by the overall percentage of hits in the training data. Thus the relative advantage of the model is expressed as a ratio to the base rate.

Close the **Evaluation** chart window
 Double-click the **Evaluation node** named **\$N-RISK**
 Click the **Lift** Chart Type option
 Click **Execute**

Figure 8.18 Lift Chart (Cumulative) with Bad Loss Credit Group as Target



The first 20 percentiles show lift values ranging from about 4 to 2.5, providing another measure of the relative advantage of the model over the base rate. Gains charts and lift charts are very helpful in marketing and direct mail applications, since they provide evaluations of how well the campaign would do if it were directed to the top X% of prospects, as scored by the model. Such charts based on the holdout data should also be examined.

We have established where the model is making incorrect predictions and evaluated the model graphically. But how is the model making its predictions? In the next section we will examine a couple of methods that may help us to begin to understand the reasoning behind the predictions.

Close the **Evaluation** chart window

Understanding the Reasoning Behind the Predictions

One method of trying to understand how a neural network is making its predictions is to apply an alternative machine learning technique, such as rule induction, to model the neural network predictions. We will introduce this approach in a later chapter.

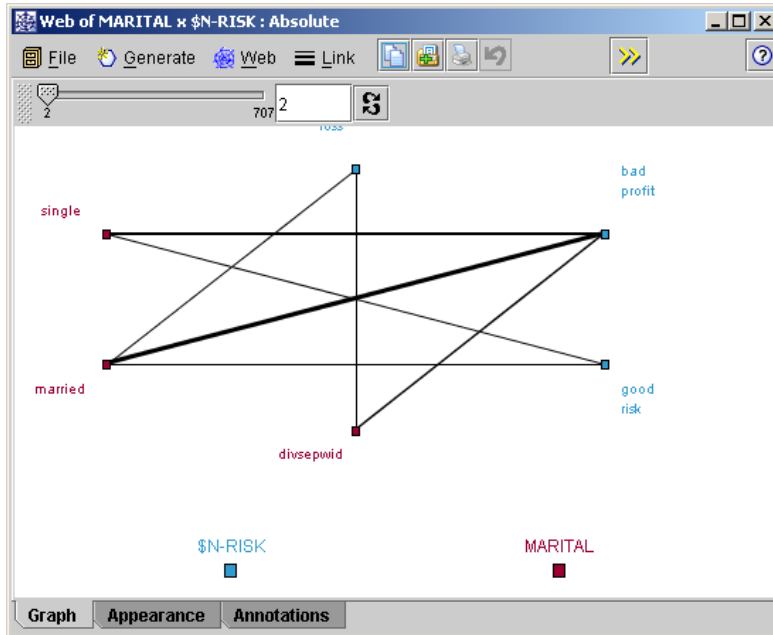
In the meantime we will use some of the methods shown before to understand the relationships between the predicted values and the fields used as inputs.

Symbolic Input with Symbolic Output

Based on the sensitivity analysis, a symbolic input of moderate importance for this network is marital status. Since it and the output field are symbolic we could use a web plot, or a distribution plot with a symbolic overlay, to understand how marital status relates to the credit risk predictions.

- Place a **Web** node from the Graphs palette near the **generated Neural Net** node named **RISK**
- Connect the **generated Neural Net** node named **RISK** to the **Web** node
- Double-click the **Web** node
- Select **MARITAL** and **\$N-RISK** in the Fields box
- Click **Execute**

Figure 8.19 Web Plot Relating Marital Status and Predicted Credit Risk

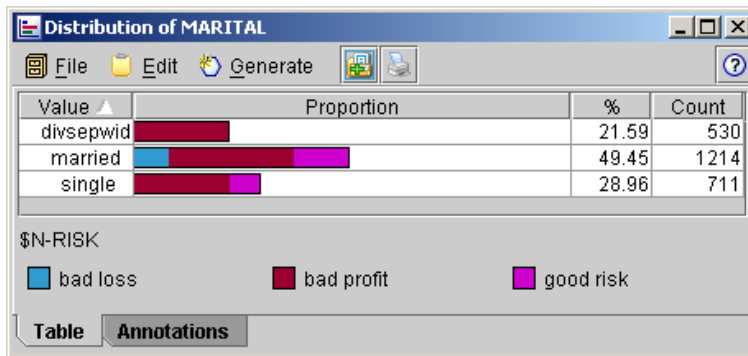


Although we have not fine-tuned the web plot, some associations between marital status and the model predictions are visible (for example, *divsepwid* and *bad profit* and no connection between *single* and *bad loss*).

We next look at a distribution plot with an overlay.

- Close the **Web** plot window
- Place a **Distribution** node from the Graphs palette near the **generated Neural Net** node named **RISK**
- Connect the **generated Neural Net** node named **RISK** to the **Distribution** node
- Double-click the **Distribution** node
- Click **MARITAL** in the **Field:** list
- Select **\$N-RISK** as the **Color Overlay** field
- Click **Execute**

Figure 8.20 Distribution of Marital Status with Predicted Credit Risk Overlay



This figure illustrates that the model is predicting the divorced, separated or widowed individuals into the *bad profit* category. The *single* individuals are associated with both *good risk* and *bad profit* categories. *Bad loss* predictions are concentrated in the *married* category.

Close the **Distribution** plot window

Numeric Input with Symbolic Output

The most important numeric input for this model is income. Since the output field is symbolic, we will use a histogram of INCOME with the predicted value as an overlay to try to understand how the network is associating income with RISK.

Place a **Histogram** node from the Graphs palette near the **generated Neural Net** node named **RISK** in the Stream Canvas

Connect the **generated Neural Net** node named **RISK** to the **Histogram** node

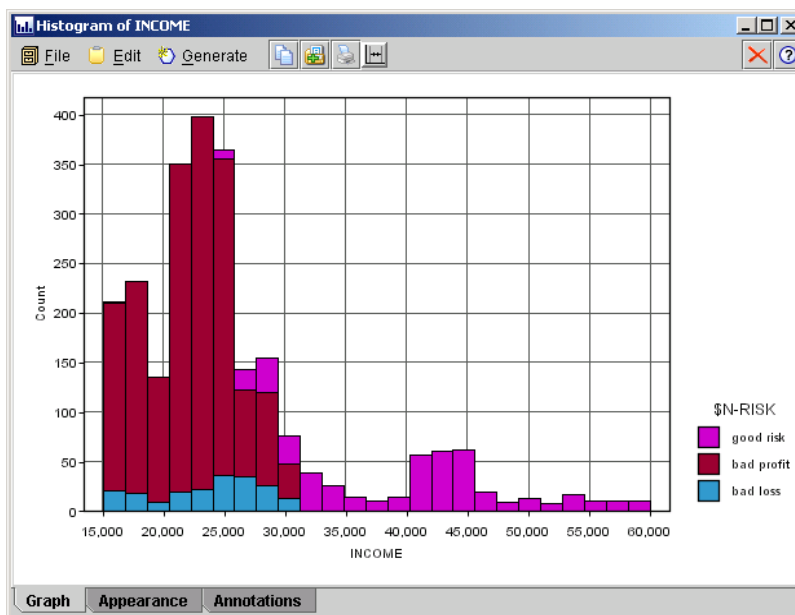
Double-click the **Histogram** node

Click **INCOME** in the **Field:** list

Select **\$N-RISK** in the **Overlay Color** field list

Click **Execute**

Figure 8.21 Histogram with Overlay of Predicted RISK [\$N-RISK]



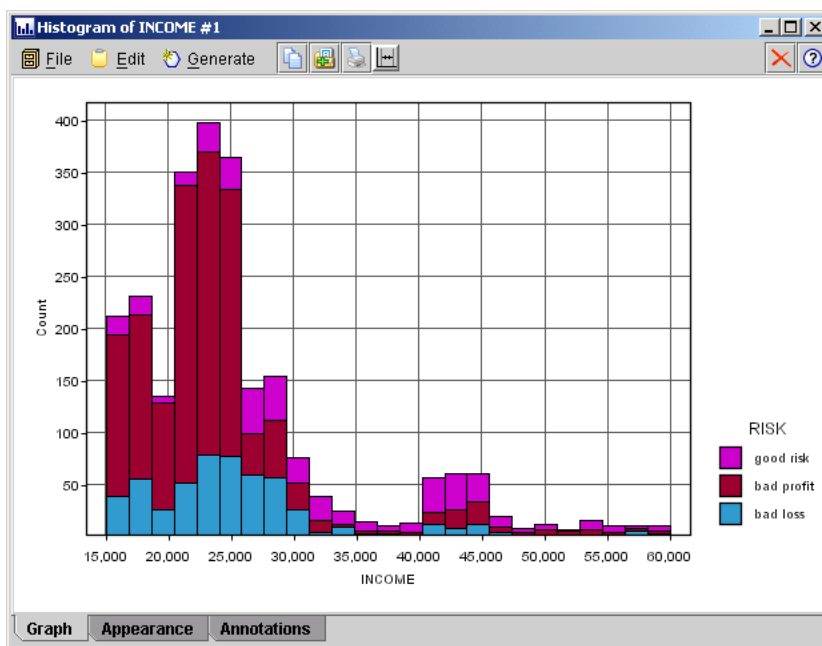
Here we see that the neural network is associating high income with good credit risk. The lower income ranges are split roughly proportionally between the two bad credit types. By comparing this to a histogram of income with the actual output field (RISK) as an overlay, we can assess where, in terms of income, the model is getting it wrong.

Double-click the **Histogram** node
 Select **RISK** in the **Overlay Color** field list
 Click **Execute**

Note

Both graphs can be viewed at once in separate windows.

Figure 8.22 Distribution of Income with Actual Credit Risk Overlay



It appears that there are some good credit risk individuals at the lower end of the salary scale. The model under-predicts this group. The model also seems to be under-predicting the bad loss credit types across the full range of income, but more noticeably at lower income levels.

Note: Use of Data Audit Node

We explored the relationship between just two input fields (MARITAL and INCOME) and the prediction from the neural net (\$N-RISK), and used the Distribution and Histogram nodes to create the plots. If more inputs were to be viewed in this way, a better approach would be to use the Data Audit node (see Data Quality chapter [Chapter 4]), because overlay plots could easily be produced for multiple input fields and a more detailed plot could be created by double-clicking on it in the Data Audit output window.

Saving the Stream

To save this stream for later work:

Click **File..Save Stream As**
 Move to the **c:\Train\ClemIntro** directory
 Type **NeuralNet** in the File Name: text box
 Click **Save**

Model Summary

In summary, we appear to have built a neural network that is pretty good at predicting the three different RISK groups. The most important factors in the making its predictions are INCOME, LOANS and AGE. The network appears to associate divorced, separated or widowed as those individuals likely to belong to the *bad profit* group.

The neural network associates high incomes with good credit types, but does not appear to be very successful in correctly identifying bad credit risks that create a loss for the company. It could be argued that this latter group of individuals is the most important to identify successfully and for this reason the network is not achieving great success.

Extensions

Ordinarily you would validate your neural network model by passing the validation data set through the generated Neural Net node and using the Matrix node, Evaluation node, and Analysis node (to be introduced later) to evaluate its performance (see Chapter 10). As an exercise you can edit the Var. File node in this stream to point to the *RiskValidate.txt* file, execute the Matrix and Evaluation nodes, and compare the results with those of the training data (*RiskTrain.txt*).

Summary

In this chapter you have been introduced, at a very basic level, to the capabilities of neural networks within Clementine.

You should now be able to:

- Build a neural network
- Browse the trained network and ascertain its predicted accuracy
- Assess the network and see where errors in predictions are made
- Attempt to understand, at a simple level, how the network is making its predictions

Chapter 9

Rule Induction

Overview

- Introduce the two rule induction nodes, C5.0 and C&R Tree
- Build a C5.0 rule
- Browse and interpret the results
- Build a Rule Set to view the rules in a different way

Objectives

In this session we introduce how to build a rule induction model within Clementine. The resulting model will be browsed and the output interpreted. We will also show how the form of the rules can be changed from a decision tree structure to a set of rules.

Data

Throughout the session we will continue using the credit risk training sample, *RiskTrain.txt*, with the aim to build a model that helps to explain the relationships between the credit risk field and the remaining fields.

Introduction

Rule induction or decision tree methods are capable of culling through a set of predictors by successively splitting a data set into subgroups on the basis of the relationships between predictors and the output field. In this section we will introduce the two algorithms in Clementine that build rule induction models. We will explain the differences between the two different algorithms and work through an example using C5.0. As in the previous chapter on neural networks, the default settings will be used throughout and the reader is referred to the *Clementine User's Guide* for more details on alternative settings and expert options. These topics are also covered in the *Advanced Modeling with Clementine* training course.

Rule Induction in Clementine

Clementine contains two different algorithms for performing rule induction: C5.0 and C&R Tree (classification and regression trees). They are similar in that they can both construct a decision tree by recursively splitting data into subgroups defined by the predictor fields as they relate to the outcome. They differ in several ways that are important to users.

- First, C5.0 only allows symbolic output fields while C&R Tree supports both symbolic and numeric outputs. Thus either could be used to build a credit risk model in which the outcome is composed of three categories of credit risk, but only C&R Tree could be used to build a model to predict second year spending (in dollars) for recently acquired customers.
- C5.0 can represent solutions as decision trees (we saw an example in Chapter 7) or in rule set form, while C&R Tree only produces decision trees. Since rule sets are arguably easier to interpret than complex decision trees, this might be a consideration when the results must be presented to

- clients. At the same time, a decision tree produces a unique classification for each data record, while more than one rule in a rule set may apply to a data record, which adds complexity, but still allows a prediction to be made (by voting).
- When the data set is recursively split into subgroups, C&R Tree supports only binary (two group) splits, while C5.0 supports splits with more subgroups for symbolic predictor fields (type Set).
 - The algorithms differ in the criterion used to drive the splitting. For C5.0 an information theory measure is used: information gain ratio. When C&R Tree predicts a symbolic output, a dispersion measure (the Gini coefficient by default) is used.
 - Both algorithms allow missing values for the predictor fields, although they use different methods. C5.0 uses a fractioning method, which passes a fractional part of a record down each branch of the tree from a node whose split is based on a field for which the record is missing; C&R Tree uses substitute prediction fields, where needed, to advance a record with missing values through the tree during training.
 - Both algorithms grow large tree trees and then prune them back: a method found to be effective. However, they differ in their pruning criteria.
 - For any data set, as decision trees grow large and bushy, the percentage of cases that pass through any given path in the tree decreases. Such bushy trees: (1) may not generalize as well to data, and (2) may have rules that apply to tiny groups of data. Both algorithms have pruning methods that trim back bushy decision trees. In addition, C5.0 contains options that favor accuracy (maximum accuracy on training sample) or generality (results that should better generalize to other data). Also, in Clementine both algorithms allow you to control the minimum subgroup size (their definitions differ slightly), which helps avoid branches with few data records.
 - C5.0 and C&R Tree will each produce a generated model node that can be browsed to examine the decision tree. C5.0 can, in addition or alternatively, create a rule set—a collection of “If..Then” rules grouped by outcome category.
 - C5.0 contains a field winnowing option, which attempts to reduce the number of fields needed to create the decision tree or rule set.

For these reasons, you should not expect the two algorithms to produce identical trees for the same data. You should expect that important predictors would be included in trees built by either algorithm.

Those interested in more detail concerning the algorithms can find additional discussion in the *Advanced Modeling with Clementine* training course. Also, you might consider *C4.5: Programs for Machine Learning* (Morgan Kauffman, 1993) by Ross Quinlan, which details the predecessor to C5.0, and *Classification and Regression Trees* (Wadsworth, 1984) by Breiman, Friedman, Olshen and Stone, who developed CART (Classification and Regression Tree) analysis.

Rule Induction Using C5.0

We will use the C5.0 node to create a rule induction model. It and the C&R Tree node are found in the Modeling Palette.

Once trained, the result is a C5.0 Rule node in the Models tab of the Manager. It contains the rule induction model in either decision tree or rule set format. By default, the C5.0 Rule node is labeled with the name of the output field. Like the generated Neural Net node, the C5.0 Rule node may be browsed and predictions can be made by passing new data through it in the Stream Canvas.

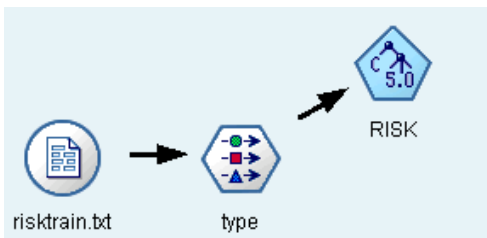
As with the Neural Net node, the C5.0 node must appear in a stream containing fully instantiated types (either in a Type node or the Types tab in a source node). Within the Type node or Types tab, the field to be predicted (or explained) must have direction OUT or it must be specified in the Fields tab of the modeling node. All fields to be used as predictors must have their direction set to IN (in Types tab or Type node). Any field not to be used in the modeling must have its direction set to NONE. Any field with direction BOTH will be ignored by C5.0.

Rather than rebuild the source and Type nodes, we use those created earlier and saved in a stream file. The C5.0 model node will use the same direction settings in the Type node as were used for modeling with neural networks, which are also appropriate for rule induction.

- Click **File..Open Stream**, and then move to the **c:\Train\ClemIntro** directory
- Double-click **NeuralNet.str** (alternatively, open Backup_NeuralNet.str)
- Delete all nodes **except** the **Var. File node** (named RiskTrain.txt) and the **Type** node (right-click on a node, then click Delete, or click on a node and press the Delete key)
- Place the **C5.0** node from the Modeling palette to the upper right of the **Type** node in the Stream Canvas
- Connect the **Type** node to the **C5.0** node

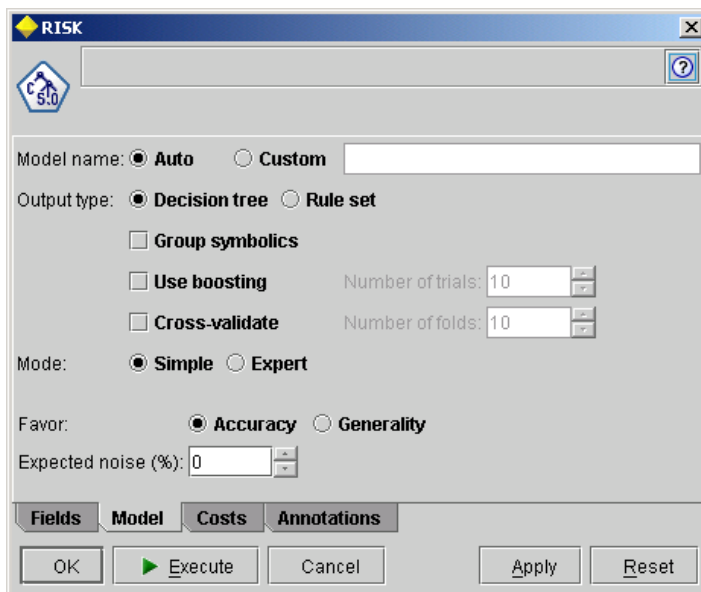
The name of the C5.0 node should immediately change to RISK.

Figure 9.1 C5.0 Modeling Node Added to Stream



Double-click the **C5.0** node

Figure 9.2 C5.0 Dialog



The *Model name* option allows you to set the name for both the C5.0 and resulting C5.0 rule nodes. The form (decision tree or rule set, both will be discussed) of the resulting model is selected using the *Output type*: option.

By default, a model is built using all data in the data stream for training. The *Cross-validate* option provides a way of validating the accuracy of C5.0 models when there are too few records in the data to permit a separate holdout sample. It does this by partitioning the data into N equal-sized subgroups and fits N models. Each model uses (N-1) of the subgroups for training, then applies the resulting model to the remaining subgroup and records the accuracy. Accuracy figures are pooled over the N holdout subgroups and this summary statistic estimates model accuracy applied to new data. Since N models are fit, N-fold validation is more resource intensive and reports the accuracy statistic, but does not present the N decision trees or rule sets. By default N, the number of folds, is set to 10.

For a predictor field that has been defined as type set, C5.0 will normally form one branch per value in the set. However, by checking the *Group symbolic values* check box, the algorithm can be set so that it finds sensible groupings of the values within the field, thus reducing the number of rules. This is often desirable. For example, instead of having one rule per region of the country, group symbolic values may produce a rule such as:

Region [South, Midwest] ...
Region [Northeast, West] ...

Once trained, C5.0 builds one decision tree or rule set that can be used for predictions. However, it can also be instructed to build a number of alternative models for the same data by selecting the *Boosting* option. Under this option, when it makes a prediction it consults each of the alternative models before making a decision. This can often provide more accurate prediction, but takes longer to train. Also the resulting model is a set of decision tree predictions and the outcome is determined by voting, which is not simple to interpret.

The algorithm can be set to favor either *Accuracy* on the training data (the default) or *Generality* to other data. In our example, we favor a model that is expected to better generalize to other data and so we select *Generality*.

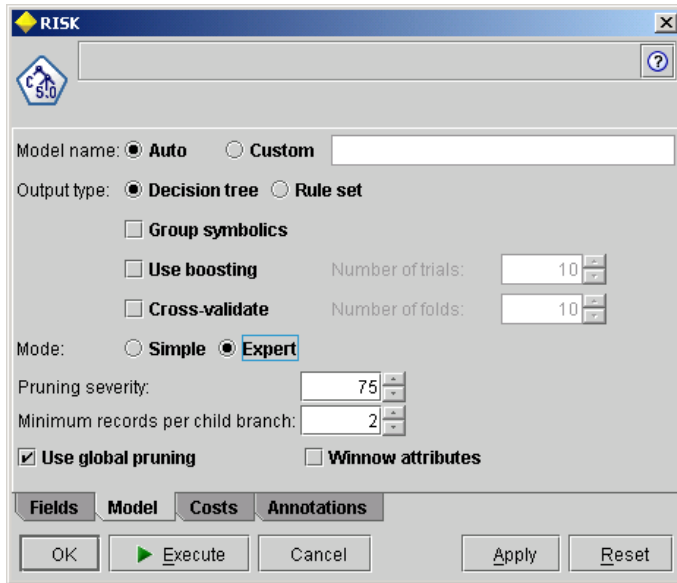
Click **Generality** option button

C5.0 will automatically handle noise within the data and, if known, you can inform Clementine of the expected proportion of noisy or erroneous data. This option is rarely used.

As with all of the modeling nodes, after selecting the Expert option or tab, more advanced settings are available. In this course, we will discuss the Expert options briefly. The reader is referred to the *Clementine User's Guide* or the *Advanced Modeling with Clementine* training course for more information on these settings.

Check the **Expert** option button

Figure 9.3 C5.0: Expert Options

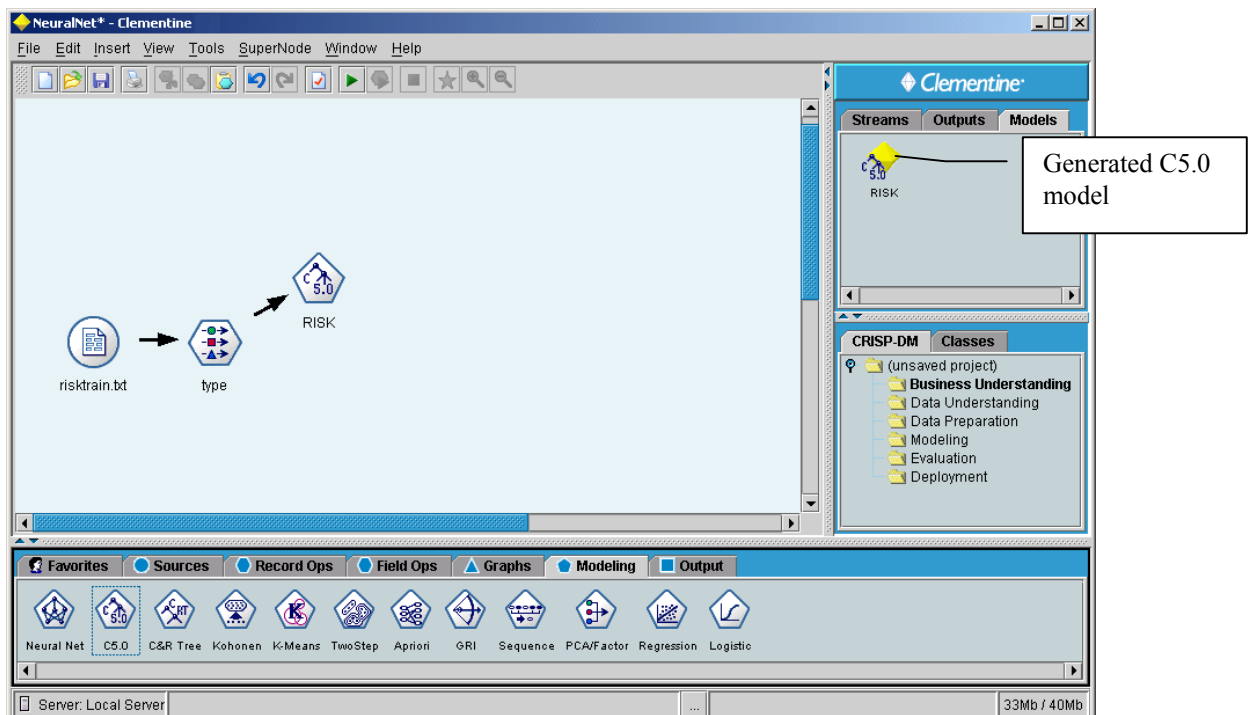


By default, C5.0 will produce splits if at least two of the resulting branches have at least two data records each. For large data sets you may want to increase this value to reduce the likelihood of rules that apply to very few records. To do so, just click the *Expert* option button and increase the value in the *Minimum records per child branch* box.

Click the **Simple** Mode option button, and then click **Execute**

A C5.0 Rule node, labeled with the predicted field (RISK), will appear in the Models palette of the Manager.

Figure 9.4 C5.0 Rule Node in the Generated Models Manager

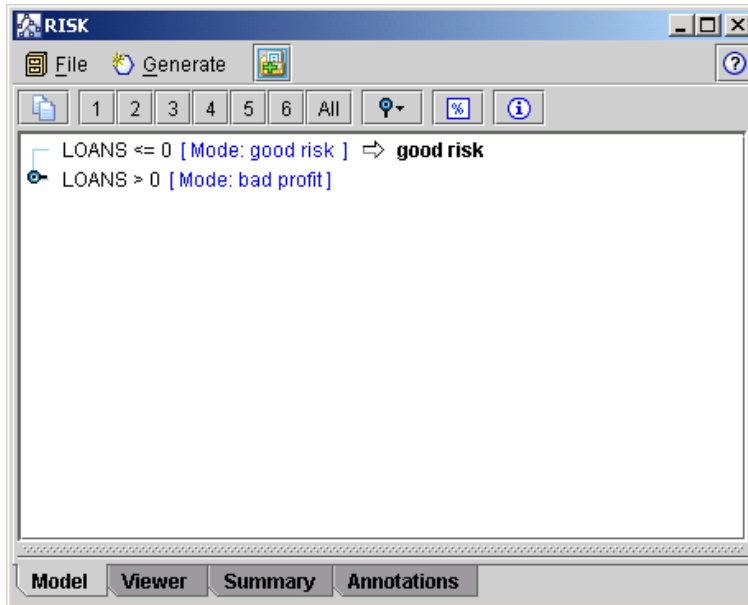


Browsing the Model

Once the C5.0 Rule node is in the Models palette, the model can be browsed.


Right-click the **C5.0** node named **RISK** in the Models palette, then click **Browse**

Figure 9.5 Browsing the C5.0 Rule Node



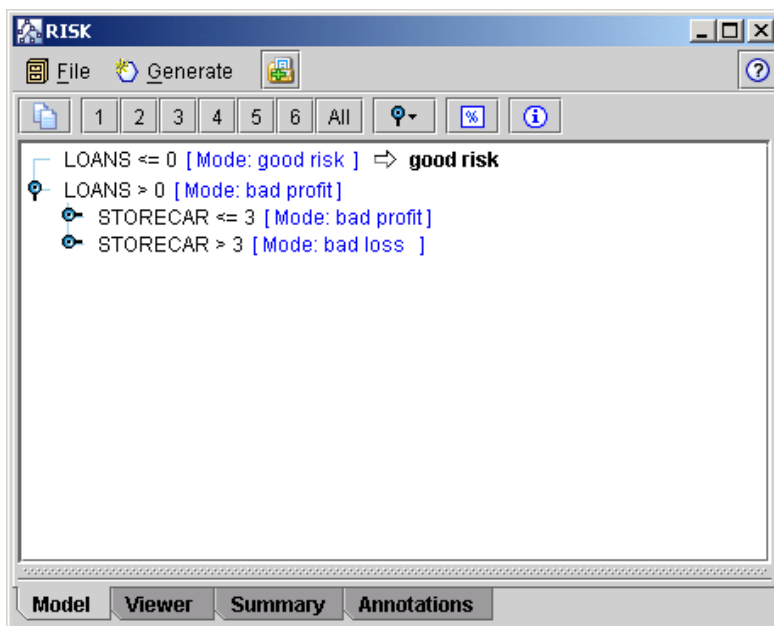
The results are in the form of a decision tree and not all branches are visible. Only the beginning of the tree is shown.

The first line indicates that LOANS is the first split field in the tree and that if LOANS ≤ 0 , then RISK is predicted to be *good risk*. The *Mode* area lists the modal (most frequent) output value for the branch. The mode will be the predicted value, unless other fields differentiate among the three risk groups. In that case we need to unfold the branch; the second line provides an example of this. In the group of records where LOANS > 0 , the category *bad profit* has the highest frequency of occurrence (Mode: *bad profit*). However, no prediction of RISK can be given because there are other fields within this group that differentiate among the three risk categories.

To unfold the branch LOANS > 0 , just click the expand button 

Click  to **unfold** the branch LOANS > 0

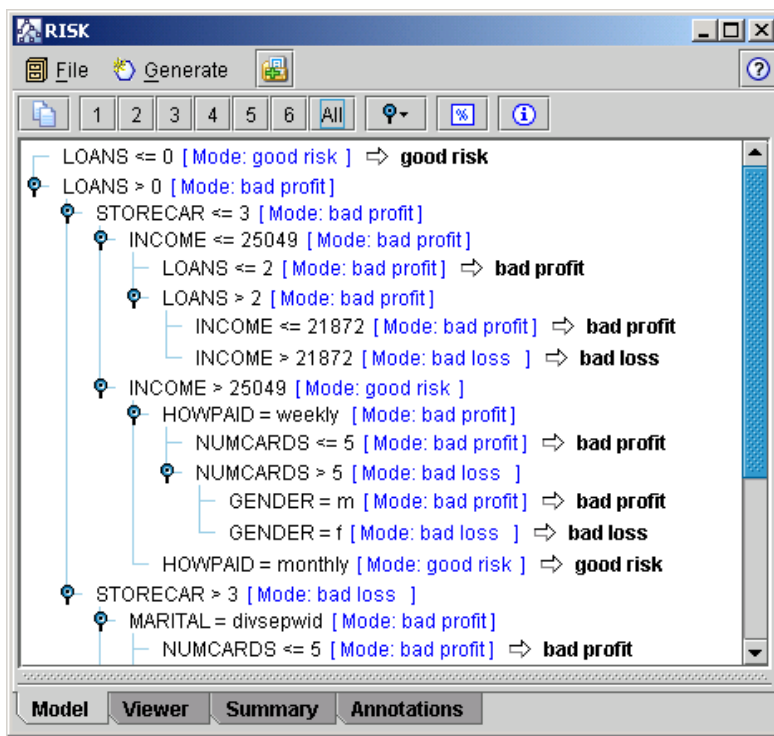
Figure 9.6 Unfolding a Branch



STORECAR appears to be the next split field. Again, the risk group cannot be predicted as there are other fields that should be taken into account for the prediction. We can once again unfold each separate branch to see the rest of the tree, but we will take a shortcut:

Click the **All** button in the Toolbar

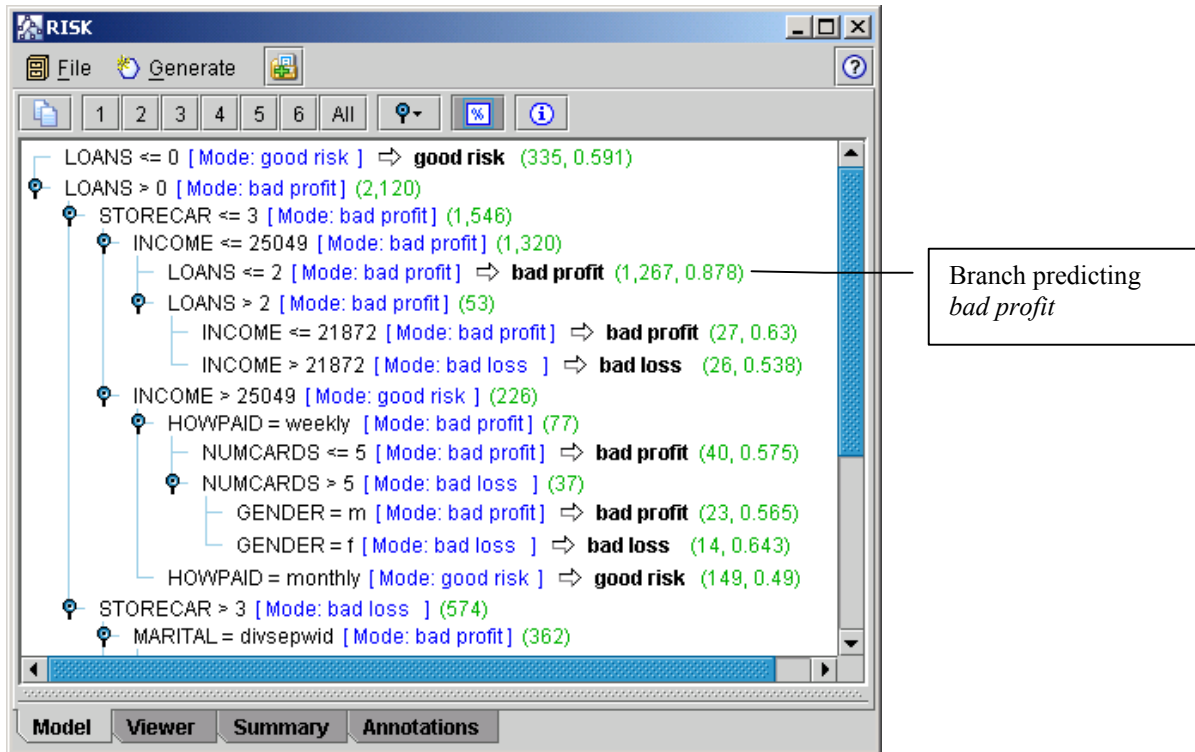
Figure 9.7 Fully Unfolded Tree



If we are interested in the *bad profit* risk group, one group are those where $LOANS > 0$, $STORECAR \leq 3$ and $INCOME \leq 25049$, and $LOANS \leq 2$. To get an idea about the number of records and the percentage of *bad profit* records within such branches we ask for more details.

Click **Show or hide instance and confidence figures**  in the toolbar

Figure 9.8 Instance and Confidence Figures Shown



The branch described in the fifth line informs us that 1,267 records had loans, three or fewer store credit cards, income less than or equal to 25,049, and two or fewer loans. The confidence figure for this set of individuals is 0.878 and represents the proportion of records within this set correctly classified (predicted to be *bad profit* and actually being *bad profit*).

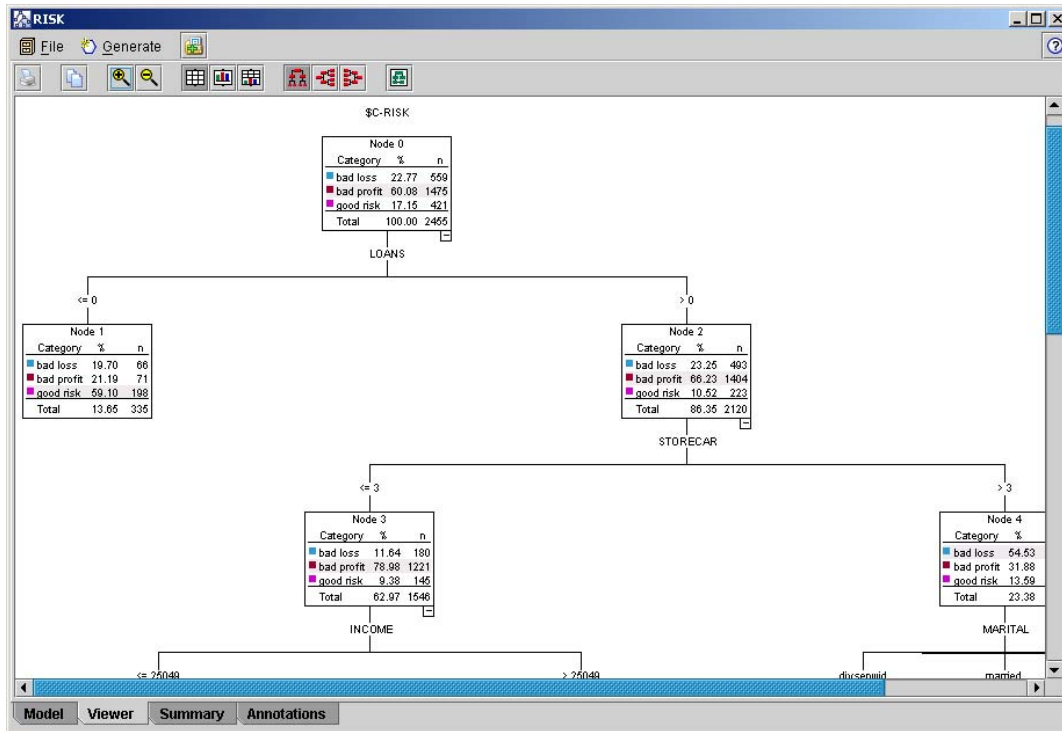
If you scroll to the bottom of the decision tree (not shown), you will find a branch with 0 records. Apparently, no singles were present in the group $LOANS > 0$, $STORECAR > 3$. On the other hand, Clementine has the information from the Type tab that **MARITAL** has three groups. If we were to score another dataset with this model, how should singles having loans and more than 3 store cards be classified? Clementine assigns the group the modal category of the branch. So, as the mode in the $LOANS > 0$ and $STORECAR > 3$ group is *bad loss*, the singles inherit this as their prediction.

If you would like to present the results to others, an alternative format is available that helps visualize the decision tree. The Viewer tab allows you to do so.

Click the **Viewer** tab

Click the Decrease Zoom  tool (to view more of the tree)

Figure 9.9 Decision Tree in the Viewer Tab



The root of the tree shows the overall percentages and counts for the three risk groups. Furthermore, the modal category is highlighted.

The first split is on LOANS, as we have seen already in the text display of the tree. Similar to the text display, we can decide to expand or collapse branches. In the right corner of some nodes a – or + is displayed, referring to an expanded or collapsed branch, respectively. For example, to collapse the tree at node 2:


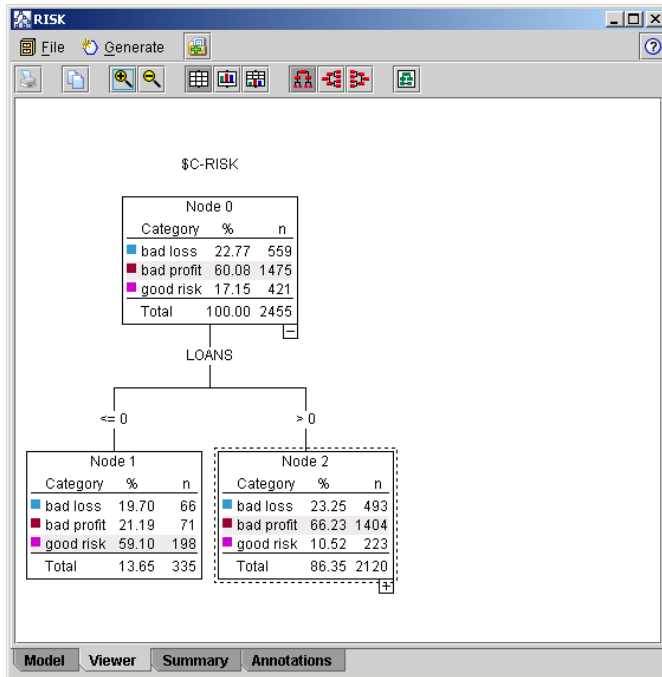
Click  in the lower right corner of **node 2**

Figure 9.10 Collapsing a Branch



In the Viewer tab, toolbar buttons are available for zooming in or out; showing frequency information as graphs and/or as tables; changing the orientation of the tree; and displaying an overall map of the tree in a smaller window (tree map window) that aids navigation in the Viewer tab.

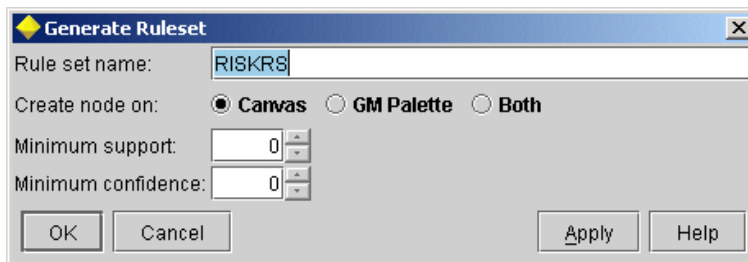
Generating and Browsing a Rule Set

When building a C5.0 model, the C5.0 node can be instructed to generate the model as a rule set, as opposed to a decision tree. A rule set is a number of IF ... THEN ... rules which are gathered together by outcome

A rule set can also be produced from the Generate menu when browsing a C5.0 decision tree model.

In the **C5.0 Rule browser** window, click **Generate..Rule Set**

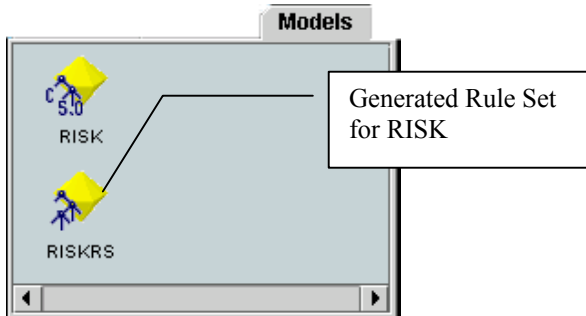
Figure 9.11 Generate Ruleset Dialog



Note that the default *Rule set name* appends the letters “RS” to the output field name. You may specify whether you want the C5.0 Ruleset node to appear in the Stream Canvas (Canvas), the generated Models palette (GM palette), or both. You may also change the name of the rule set and lower limits on support (percentage of records having the particular values on the input fields) and confidence of the produced rules (percentage of records having the particular value for the output field, given values for the input fields).

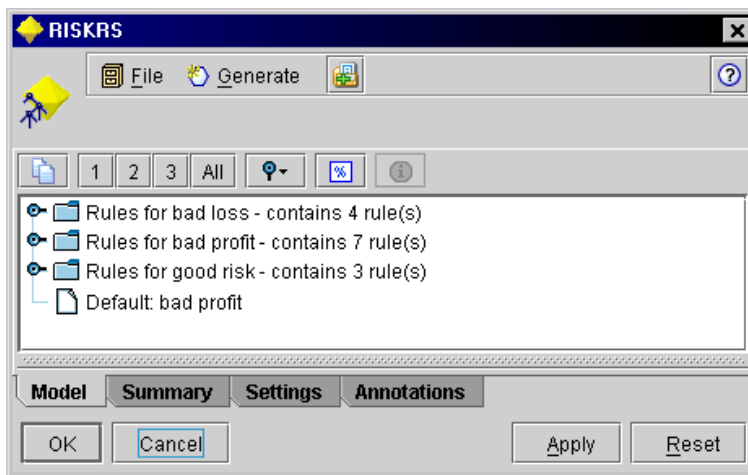
Set **Create node on:** to **GM Palette**
Click **OK**

Figure 9.12 Generated C5.0 Rule Set Node



Click **File..Close** to close the C5.0 Rule browser window
Right-click the C5.0 Rule Set node named **RISKRS** in the generated Models palette in the Manager, then click **Browse**

Figure 9.13 Browsing the C5.0 Generated Rule Set



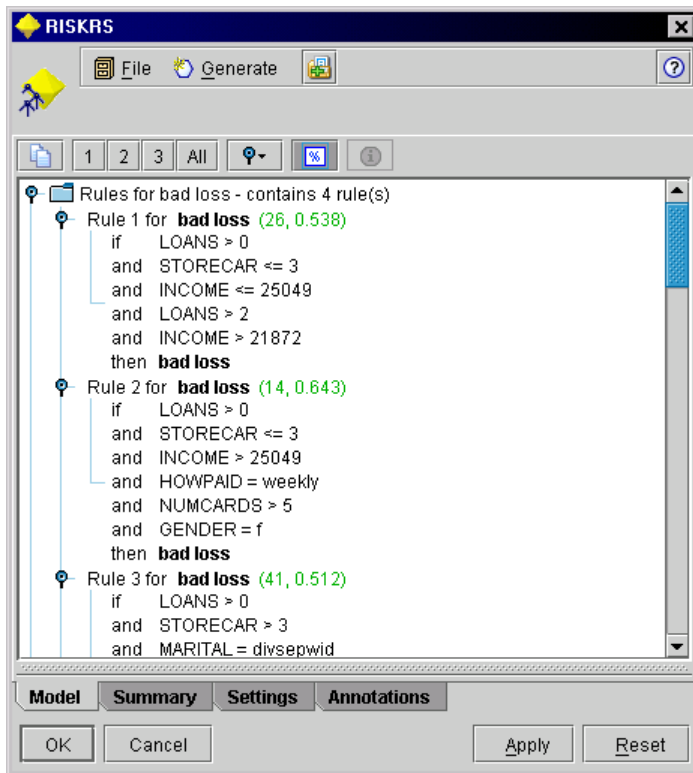
Apart from some details, this window contains the same menus as the browser window for the C5.0 Rule node.

Click **All** button to unfold

Click **Show or hide instance and confidence figures**  button in the toolbar

The numbered rules now expand as shown below.

Figure 9.14 Fully Expanded C5.0 Generated Rule Set



For example Rule #1 (*bad loss*): If the person has more than two loans, 3 or fewer store cards and income above 21,872 but no more than 25,049, then predict *bad loss*. This form of the rules allows you to focus on a particular conclusion rather than having to view the entire tree.

The Settings tab allows you to export the rule set in SQL format, which permits the rules to be directly applied to a database.

Click **File..Close** to close the Rule set browser window

Understanding the Rule and Determining Accuracy

Unlike the generated Neural Net node, the predictive accuracy of the rule induction model is not given directly within the C5.0 node. In the next chapter we will introduce the Analysis node that provides information on model performance; however, at this stage we will use the same approach as in the last chapter.

Creating a Data Table Containing Predicted Values

Place the **generated C5.0 Rule** node named **RISK** from the **Models** palette in the Manager to the right of the **Type** node

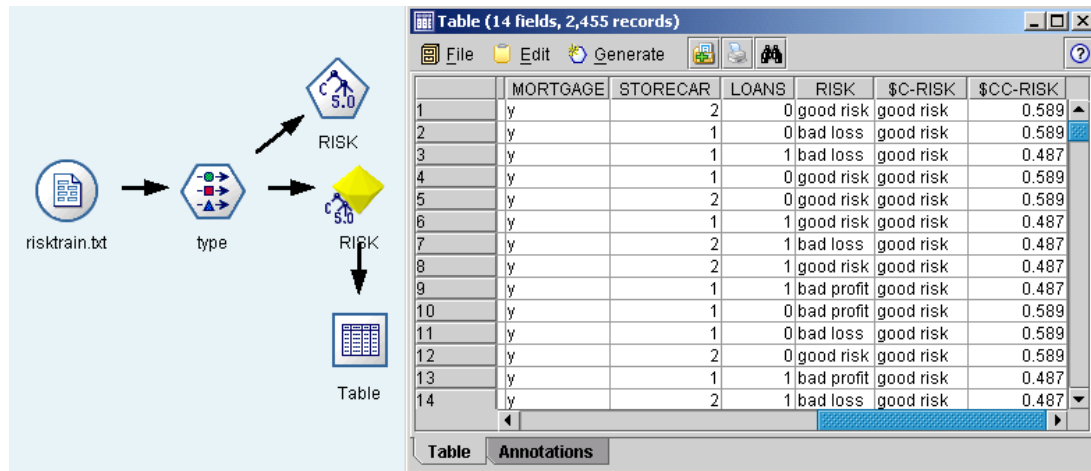
Connect the **Type** node to the **generated C5.0 Rule** node named **RISK**

Place a **Table** node from the Output palette below the generated C5.0 Rule node named **Risk**

Connect the **generated C5.0 Rule** node named **RISK** to the **Table** node

Right-click the **Table** node, then click **Execute** and scroll to the **right** in the table

Figure 9.15 Two New Fields Generated by the C5.0 Rule Node



Two new columns appear in the data table, \$C-RISK and \$CC-RISK. The first represents the predicted value for each record and the second the confidence value for the prediction.

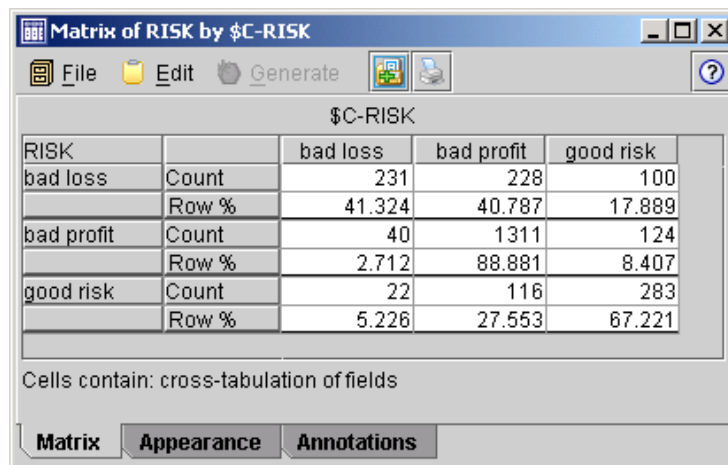
Click **File..Close** to close the Table output window

Comparing Predicted to Actual Values

As in the previous chapter, we will view a data matrix to see where the predictions were correct and evaluate the model graphically with a gains chart.

- Drag the **C5.0 model** node named **RISK** above the **Type** node
- Place a **Matrix** node from the Output palette above the **generated C5.0 Rule** node named **RISK**
- Connect the **generated C5.0 Rule** node named **RISK** to the **Matrix** node
- Double-click the **Matrix** node
- Select **RISK** in the **Rows:** list
- Select **\$C-RISK** field in the **Columns:** list
- Click the **Appearance** tab
- Click the **Percentage of row** check box
- Click **Execute**

Figure 9.16 Matrix of Actual (Rows) and Predicted (Columns) Credit Risk



The model predicts about 67% of the good risk category correctly, almost 89% of the bad but profitable risks, and 41% of the bad with loss risks correctly. These results are a bit worse than those found with a neural network for the bad profit category (89% versus 90%), but better for the bad loss (41% versus 31%) and good risk (67 versus 63%) categories. Overall, the decision tree has performed better on the training file and we will compare the models on validation data in Chapter 10.

Click **File..Close** to close the Matrix window

To produce a gains chart for the bad loss group:

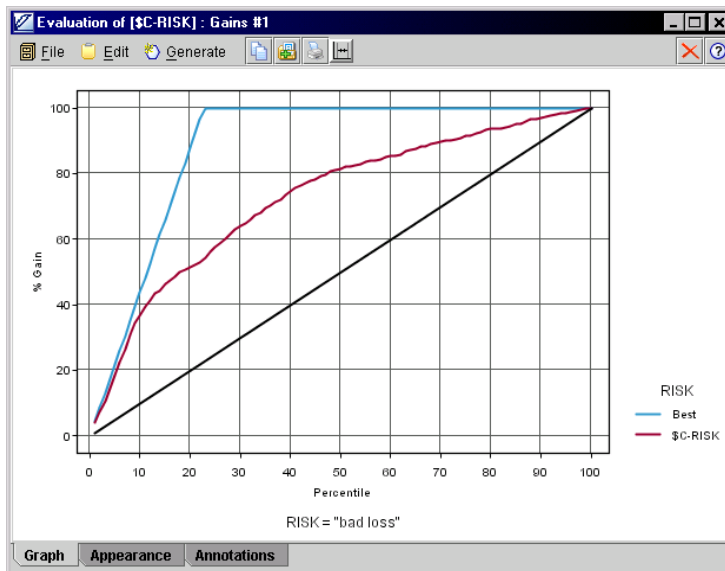
Place the **Evaluation** chart node from the **Graphs** palette to the right of the **generated C5.0 Rule** node named **RISK**

Connect the **generated C5.0 Rule** node named **RISK** to the **Evaluation** chart node

Double-click the **Evaluation** chart node, and click the **Include best line** checkbox

Click **Execute**

Figure 9.17 Gains Chart of Bad Loss Credit Group



The gains line ($\$C-RISK$) rises steeply relative to the baseline, indicating the hits for the *bad loss* outcome are concentrated in the percentiles predicted most likely to contain bad losses according to the model. This gains line is similar to the one for the bad loss category based on the neural net model (Figure 8.17).

Click **File..Close** to close the **Evaluation** chart window

Changing Target Category for Evaluation Charts

By default, an Evaluation chart will use the first target outcome category to define a hit. To change the target category on which the chart is based, we must specify the condition for a *User defined hit* in the Options tab of the Evaluation node. To create a gains chart in which a hit is based on the good risk category:


Double-click the **Evaluation** node


Click the **Options** tab

Click the **User defined hit** checkbox

Click the **Expression Builder** button  in the **User defined hit** group

Click **@Functions** on the functions category drop-down list

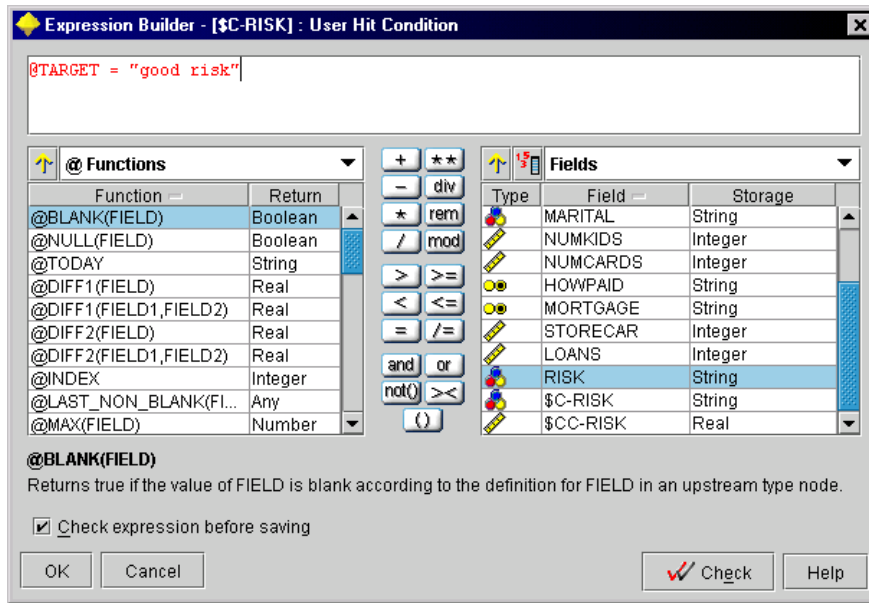
Select **@TARGET** on the functions list, and click the Insert button 

Click the = button 

Right-click **RISK** in the Fields list box, then select **Field Values**

Select **good risk**, and then click **Insert** button

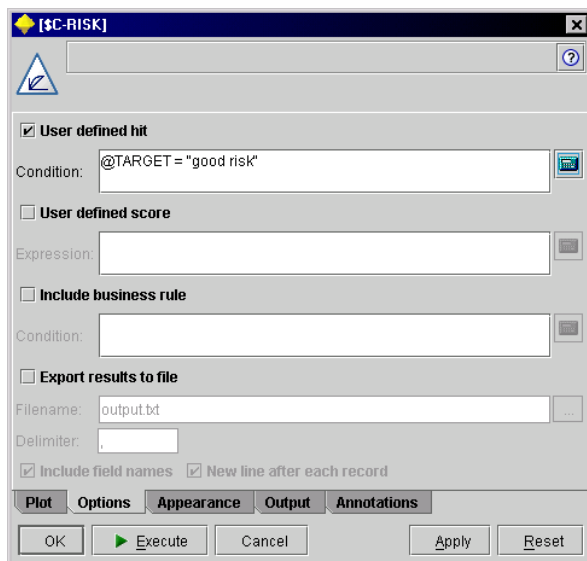
Figure 9.18 Specifying the Hit Condition within the Expression Builder



The condition (good risk is the target value) defining a hit was created using the Expression Builder. Note the expression will be checked when OK is clicked.

Click **OK**

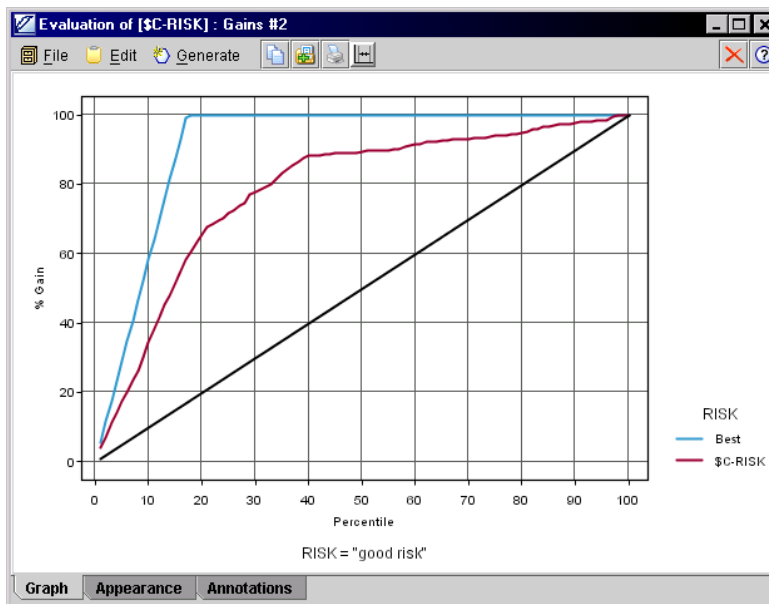
Figure 9.19 Defining the Hit Condition for RISK



In the evaluation chart, a hit will now be based on the good risk target category.

Click **Execute**

Figure 9.20 Gains Chart for the Good Risk Category



The gains chart for the *good risk* category is better (steeper in the early percentiles) than that for the *bad loss* category. For example, the top 20 model-ordered percentiles contain over 60% of the good risks.

Click **File..Close** to close the **Evaluation** chart window

To save this stream for later work:

Click **File..Save Stream As**
 Move to the **c:\Train\ClemIntro** directory
 Type **C5** in the File name: text box
 Click **Save**

Understanding the Most Important Factors in Prediction

An advantage of rule induction models over neural networks is that the decision tree form makes it clear which fields are having an impact on the predicted field. There is no great need to use alternative methods such as web plots and histograms to understand how the rule is working. Of course, you may still use the techniques described in the previous chapter to help understand the model, but they may not be needed.

Unlike the neural network, there is no sensitivity analysis performed on the model. The most important fields in the predictions can be thought of as those that divide the tree in its earliest stages. Thus in this example the most important field in predicting risk is **LOANS** (number of loans). Once it has divided the data into two groups, those with loans and those without, it next makes predictions based on income and number of store cards held.

Summary

In this chapter you have been introduced to the two rule induction algorithms within Clementine, C5.0 and C&R Tree.

You should now be able to:

- Build a rule induction model using C5.0
- Browse the model in the form of a decision tree
- Generate a rule set form of the model
- Look at the instances and confidence of a decision tree branch and interpret the model
- Change the hit condition in an evaluation chart

Chapter 10

Comparing and Combining Models

Overview

- Introduce the Analysis Node
- Compare Models using Evaluation charts and Analysis Node
- Compare Models on Validation Data
- Using Rule Induction with Neural Networks

Objectives

In this session we will introduce the Analysis node as a way of assessing the performance of models and demonstrate a way of comparing the results of different models. We will also compare models within an evaluation chart and then compare two models using a validation data set. We also introduce approaches to combining the results of different models.

Data

We will continue using the training portion of the credit risk data, *RiskTrain.txt*, to predict credit risk (RISK). For the validation analysis, we use a separate validation sample (*RiskValidate.txt*).

Introduction

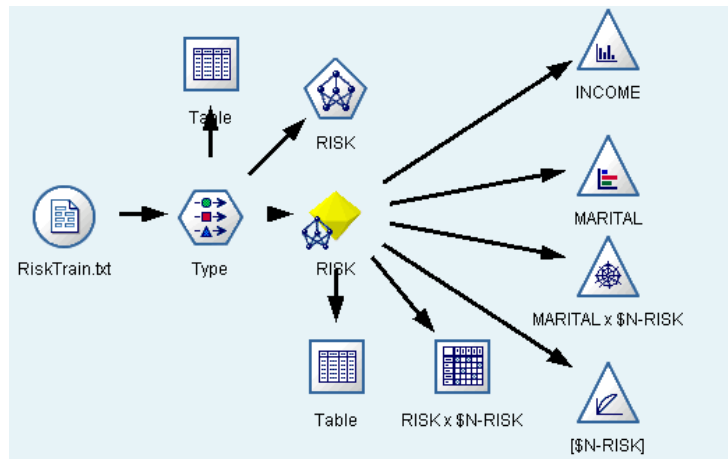
In this chapter we will compare the neural network model to the C5.0 rule induction model. The Analysis node will be introduced as a way of assessing different models and an evaluation chart, introduced earlier, will be used for the same purpose. It is important to evaluate models using validation data and we will demonstrate how to do so with a slight modification to a stream pointing to the training data.

Comparing Models

To be able to compare models, they need to be within the same stream and connected to the same data source node. We will place the generated model nodes we created in the two previous chapters into the same stream.

Click **File..Open Stream**, then move to **the c:\Train\ClemIntro** directory
Double-click **NeuralNet.str** (alternatively, use Backup_NeuralNet.str)

Figure 10.1 Stream that Builds and Explores a Neural Network Model



This stream contains a data source (Var. File) node; a Type node with RISK specified as the output field; and a generated Net node containing the neural net model run in Chapter 8. After removing unneeded nodes, we will rerun the C5.0 rule induction model (done in Chapter 9) and add its generated Rule node to the stream.

Delete **all nodes downstream** of the generated **Net node** [delete all graph nodes, the Table and Matrix nodes] (right-click on a node, then click Delete; or click on a node and then press the Delete key)

Place the **C5.0** node from the Modeling palette to the lower right of the **Type** node in the Stream Canvas

Connect the **Type** node to the **C5.0** node

Double-click on the **C5.0** node

Click the Favor **Generality** option button

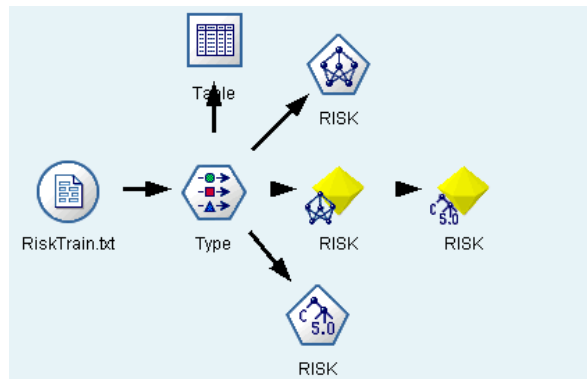
Click **Execute**

Once the algorithm has finished, a generated C5.0 Rule node will appear in the Generated Models palette of the Manager.

Place the generated **C5.0 Rule** node from the **Generated Models palette** to the right of the generated **Net** node in the Stream pane

Connect the generated **Net** node to the generated **C5.0 Rule** node

Figure 10.2 Stream Containing Neural Network and Rule Induction Models



We have two generated model nodes within the same stream connected to a data source, and can now compare these models using the Analysis node.

Analysis Node

The Analysis node is used to evaluate how well predicted values from a generated model fit the actual values. The node can also be used to compare predicted fields from different models and ascertain their overlap, that is, how often they agree in their predictions.

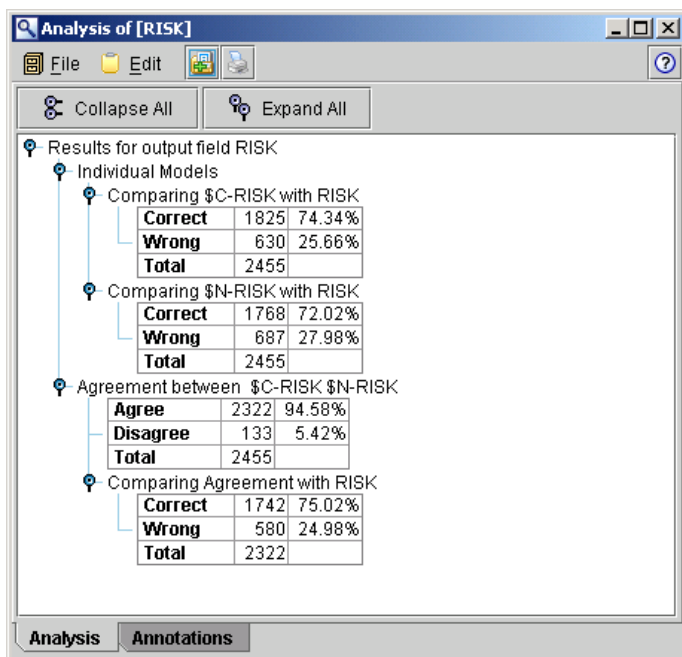
Place an **Analysis** node from the Output palette to the right of the **C5.0 Rule** node in the Stream Canvas

Connect the **C5.0 Rule** node to the **Analysis** node

The Analysis node may be edited to obtain coincidence matrices (also called misclassification tables or confusion matrices— we produced these using the Matrix node) and additional performance statistics. For our purposes the default settings are all that are required. The Help system and *Clementine User's Guide* contain details on all the options.

Right-click the **Analysis** node, then click **Execute**

Figure 10.3 Analysis Output Comparing Neural Net and Rule Induction Models



The first section compares the actual values of RISK with those predicted using the C5.0 rule model, \$C-RISK. The second section compares RISK values with those predicted using the Neural Net node, \$N-RISK. The final section details the level of agreement between the two different predicted fields.

For the training data, the C5.0 model gets about 74% of its predictions correct while the neural network achieves about 72% correct. On the training data, the decision tree slightly outperforms the neural network overall.

The comparison shows that the two models agree for about 94.5% of the records and when they agree, the predictions are correct in about 75% of the instances.

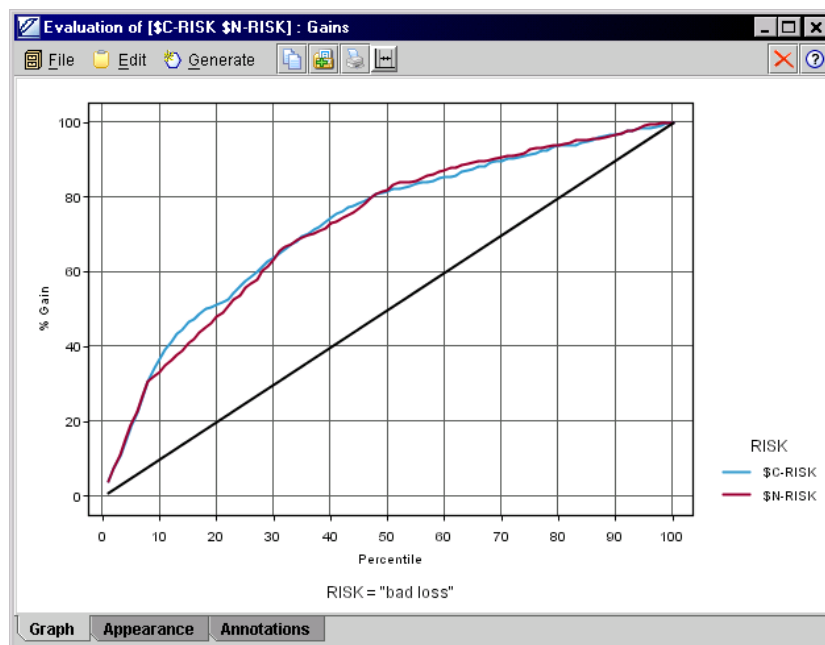
Close the **Analysis** output window

Evaluation Charts for Model Comparison

Evaluation charts can easily compare the performance of different models. To demonstrate we will create a gains chart based on the predictions from the neural network and rule induction models.

- Place the **Evaluation chart** node from the Graphs palette to the lower right of the generated **C5.0 Rule** node in the Stream pane
- Connect the generated **C5.0 Rule** node to the **Evaluation chart** (named Evaluation) node
- Right-click the **Evaluation chart** node, then click **Execute**

Figure 10.4 Gains Chart Results (Bad Loss Category) for Two Models



Now two lines (one for each model) appear in addition to the baseline. The models perform similarly with a slight advantage to the C5.0 model appearing in the 10th through 20th percentiles. Thus under the settings we used, the C5.0 model would be a bit more effective in predicting bad loss. Of course, this chart should also be viewed using validation data, which we turn to next. In this way, different models can be compared in gains, lift, profit, or ROI chart form.

Close the **Evaluation chart** window

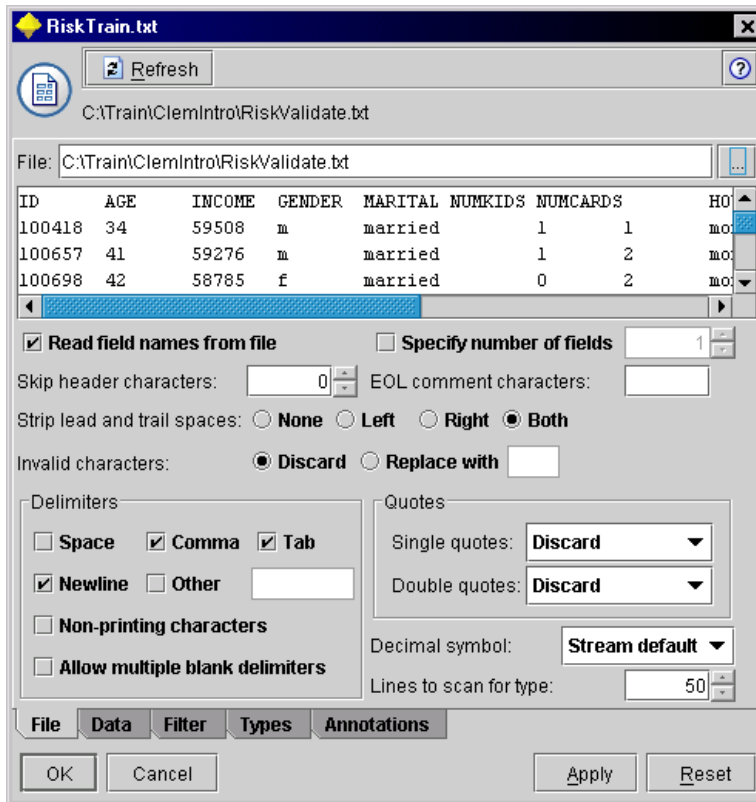
Comparing Models Using Validation Data

The model comparison we just ran used the same data as were used in the original model estimation. We discussed earlier that it is better, when evaluating an individual model or when comparing models, to use a separate validation sample. We will now compare the models using data stored in the *RiskValidate.txt* file.

Given that our stream already passes the data through the two generated model nodes and contains an Analysis and Evaluation chart node, we can easily switch to the validation data by changing the file reference in the data source node (Var. File node). Once that is done, we need only execute the output nodes of interest and summaries will be produced based on the validation file.

Double-click on the **Var. File** node (named RiskTrain.txt)
Replace RiskTrain.txt with **RiskValidate.txt** in the **File** box

Figure 10.5 Reading Validation Data into a Stream



The source node now points to the validation data file.

Click **OK**

Now we rerun the Analysis node.

Right-click the **Analysis** node
Click **Execute**

Figure 10.6 Analysis Output Comparing Neural Net and Rule Induction Models on Validation Data

Analysis of [RISK] #5

File Edit

Collapse All Expand All

Results for output field RISK

- Individual Models
 - Comparing \$C-RISK with RISK

Correct	1309	78.76%
Wrong	353	21.24%
Total	1662	
 - Comparing \$N-RISK with RISK

Correct	1310	78.82%
Wrong	352	21.18%
Total	1662	
 - Agreement between \$C-RISK \$N-RISK

Agree	1600	96.27%
Disagree	62	3.73%
Total	1662	
 - Comparing Agreement with RISK

Correct	1286	80.38%
Wrong	314	19.62%
Total	1600	

Analysis Annotations

The typical outcome when validation data are passed through a model node is that the accuracy drops a small amount. If accuracy drops by a large amount, it suggests that the model overfit the training data or that the validation data differ in some systematic way from the training data (although random sampling is typically used to minimize the chance of this). Here the accuracy of both models improved in the validation sample. While this is a welcome result, it is also uncommon.

The C5.0 model is correct for about 79% of the validation records, which represents an increase from the 74% we found in the training data. Similarly, the neural network is correct for about 79% of the validation records- an increase over the 72% found in the training data. Thus the results from the validation data suggest that the models give very similar results. Since the overall difference in accuracy is quite small, other considerations, such as transparency of the model, could play a role in deciding which model to use.

In this way, a validation sample serves as a useful check on the model accuracy reported with the training data.

Close the **Analysis** window

Before proceeding, we will restore the training data to the stream.

Double-click on the **Var. File** node (named RiskValidate.txt)
 Replace RiskValidate.txt with **RiskTrain.txt** in the **File** box
 Click **OK**

Using Rule Induction with Neural Networks

Using Rule Induction before Neural Networks

A point to bear in mind, when using neural networks other than the Prune algorithm in Clementine, is that all fields presented to the algorithm as inputs will be used in the network. This can have two associated problems:

- The more fields in a network, the longer it can take to train
- Fields that do not have an impact on the output field, remain in the model

Since it is very clear which fields are being used within a rule induction model, and for large files rule induction models tend to be much faster to train than neural networks, these algorithms can be used as a preprocessing tool to reduce the number of input fields entering a neural network.

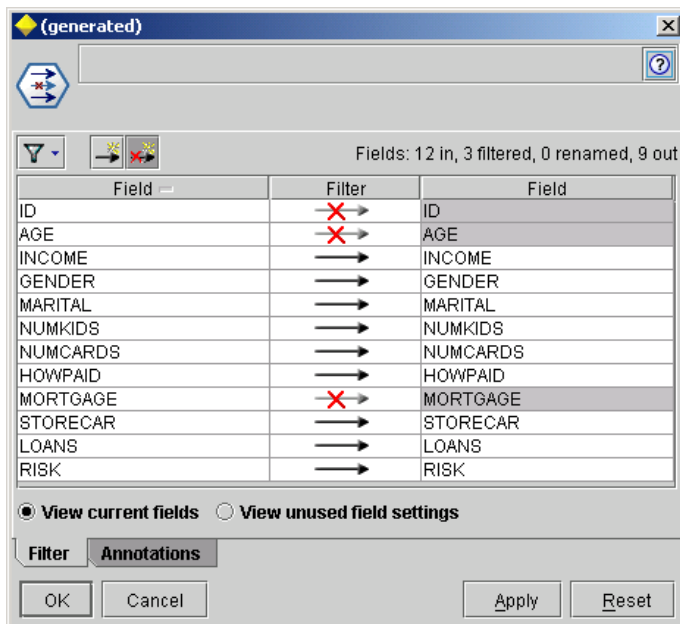
To illustrate how to use rule induction to achieve this, we will return to the generated C5.0 Rule node within the Stream Canvas and browse the node. A Filter node that discards fields not used in the rule induction model can be automatically generated by clicking *Generate..Filter node* within the C5.0 Rule browser window.

Right-click the **C5.0 Rule** node named RISK in the Stream Canvas, then click **Edit**
Click **Generate..Filter node** in the C5.0 Rule browser window
Close the C5.0 Rule browser window

A Filter node labeled (*generated*) appears in the top left corner of the Stream Canvas.

Drag the **Filter** node to right of the **Type** node in the Stream canvas
Connect the **Type** node to the **Filter** node
Double-click the **Filter** node

Figure 10.7 Filter Node Generated from C5.0 Rule

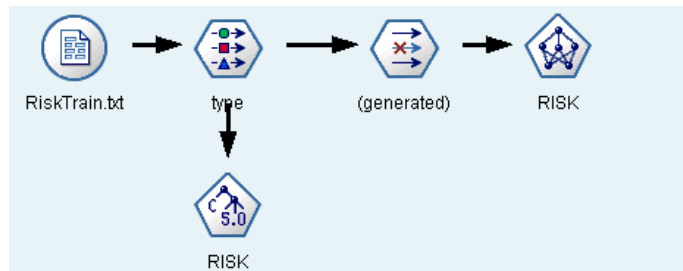


In addition to the ID field, which was typeless, AGE and MORTGAGE are excluded. We could connect a Neural Net node downstream of this Filter node and train a neural network on the fields used by the C5.0 rule induction model. In this way we would use C5.0 to reduce the number of input fields used by the neural network. This can reduce processing when large numbers of fields and records are involved in

modeling. The *Winnow Attributes* option, available as a C5.0 expert option, attempts to build a C5.0 model using fewer input fields. This can further reduce the number of fields passed as inputs to a neural network.

The resulting stream would look similar to the one shown below.

Figure 10.8 Filter Node Discarding Fields Not Used By a C5.0 Model (Prior to Running a Neural Network Model)



Using Rule Induction after Neural Networks

In Chapter 8, we described a number of different methods to help understand the reasoning behind the neural network's predictions. In that chapter we also mentioned that rule induction might be used to help interpret a neural network. We now consider this. The basic logic involves using rule induction to fit the predictions from the neural network model. Since rules are relatively easy to interpret, this may provide insight into the workings of the neural network. We must be aware, however, that we are using one type of model to fit and explain a different type of model, which has limitations.

To help interpret the neural network we will attempt to predict the generated field \$N-RISK (the predicted output from the neural network) using rule induction and the fields used as inputs to the neural network. In order to predict the field, \$N-RISK, generated by the network, the C5.0 node will be placed downstream of the generated Net node.

Delete the **Analysis** node, **Evaluation chart** node, the generated **C5.0 Rule** node, and the generated **Filter** node (right-click on a node, then select Delete from the context menu)

Place a **Type** node from the Field Ops palette to the right of the generated **Net** node named RISK

Connect the generated **Net** node named RISK to the **Type** node

Double-click the new **Type** node

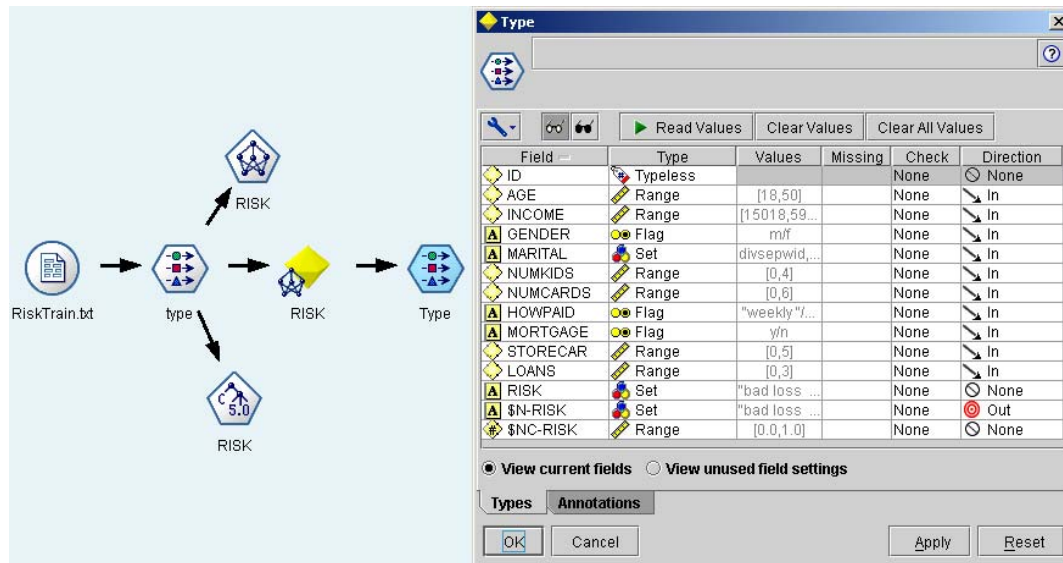
Within the Type node dialog:

Set the **Direction** of **RISK** and **\$NC-RISK** fields to **NONE** (click in Direction cell and select None from the drop-down list)

We do not want to use the actual risk values or the confidence of the network predictions as inputs.

Set the **Direction** of **\$N-RISK** to **OUT** (click in the Direction cell and select Out from the drop-down list)

Figure 10.9 Type Node for Input to C5.0 Modeling Node to Predict Neural Network Results



Click **OK**

Place a **C5.0** node from the Modeling palette to the right of the new **Type** Node in the Stream Canvas

Connect the new **Type** node to the new **C5.0** node

Right-click the new **C5.0** node named **\$N-RISK**, then click **Execute**

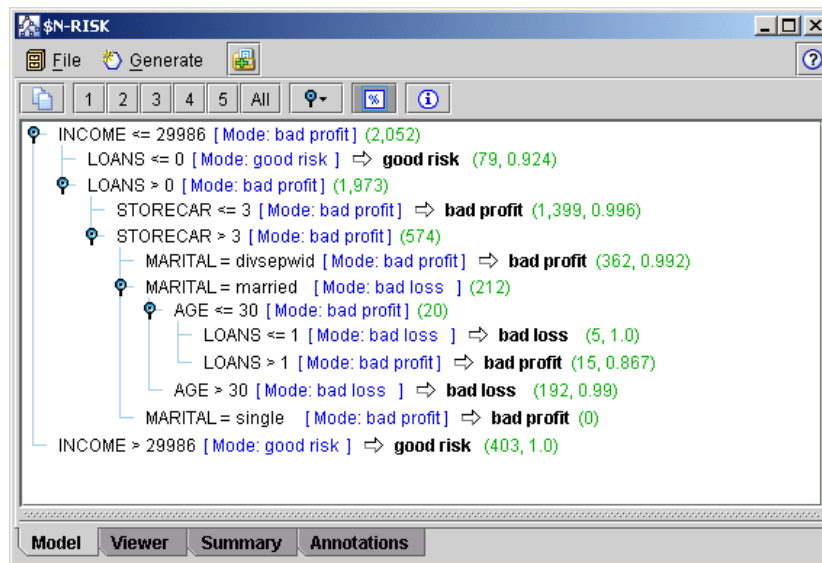
A new C5.0 Rule node, labeled \$N-RISK, should appear in the generated Models palette in the Manager.

Right-click the **generated C5.0 Rule** node named **\$N-RISK** in the generated Models palette of the Manager, then click **Browse**

Click the **All** button

Click the Show instances/confidences figure  button

Figure 10.10 Browsing the C5.0 Rules Based on the Neural Network Predictions



The neural network appears to be predicting records with income over 29,986, or below this but with no loans, as good credit risks. Those records with income at or below 29,986 and with loans are predicted as bad credit risks. They are classified as profitable or loss creating based on whether they have a large number of store cards, their marital status, number of loans and age.

We are now a little closer in understanding the reasoning behind the neural network's predictions.

Summary

In this chapter you have been introduced to some ways of comparing models on the same data and in how to use a validation data sample to evaluate models.

You should now be able to:

- Use the Analysis node to quantify the performance of a predictive model
- Use the Analysis node to compare different predictive models
- Evaluate models using a validation data set
- Use decision trees to discard fields prior to modeling with neural networks
- Use decision trees to gain insight into neural network models

Chapter 11

Kohonen Networks

Overview

- Introduce the Kohonen node
- Build a Kohonen network
- Interpret the results

Objectives

In this session we will introduce the Kohonen node as a way of segmenting or clustering your data. The results will be interpreted.

Data

In this chapter we will attempt to segment a data set containing shopping information, *Shopping.txt*. The file contains fields that indicate whether or not, during a customer visit, a customer purchased a particular product. Thus each record represents a store visit in which at least one product was purchased. The file also contains basic demographics, such as gender and age group, which will be used to help interpret the resulting segments.

Introduction

Kohonen networks perform unsupervised learning; an *Out* field is not specified and the model is, therefore, not given an existing field in the data to predict. As described in Chapter 7, Kohonen networks attempt to find relationships and overall structure in the data. The output from a Kohonen network is a set of (X, Y) coordinates, which can be used to visualize groupings of records and can be combined to create a cluster membership code. It is hoped that the cluster groups or segments are distinct from one another and contain records that are similar in some respect.

In this chapter we will attempt to segment a data set containing purchase information.

Kohonen Node

The Kohonen node is used to create a Kohonen network and is found in the Modeling palette within Clementine.

As with neural networks, the trained network will result in a generated Kohonen model node in the generated Models palette in the Manager. Browsing this node will give information on the structure of the Kohonen network and graphical profiles of the clusters are provided in the cluster viewer.

First we need create a source node to read in the data, and then instantiate the field types.


- Place a **Var. File** node from the Sources palette into the Stream Canvas
- Double-click the **Var. File** node
- Select **Shopping.txt** (in **c:\Train\ClemIntro**) as the File

- Make sure the **Read field names from file** option is checked
- Click the Delimiters: **Tab** check box
- Click the **Types** tab
- Right-click on any field and choose **Select All** from the context menu
- Right-click on any field and click **Set Type..Discrete** from the context menu
- Click **Read Values** button (to instantiate the field types), then click **OK**
- Click **OK** to return to the Stream Canvas
- Place a **Table** node from the Output palette above the **Var. File** node
- Connect the **Var. File** node to the **Table** node
- Execute** the **Table** node

Figure 11.1 Table Node Displaying Part of the Shopping.txt Data

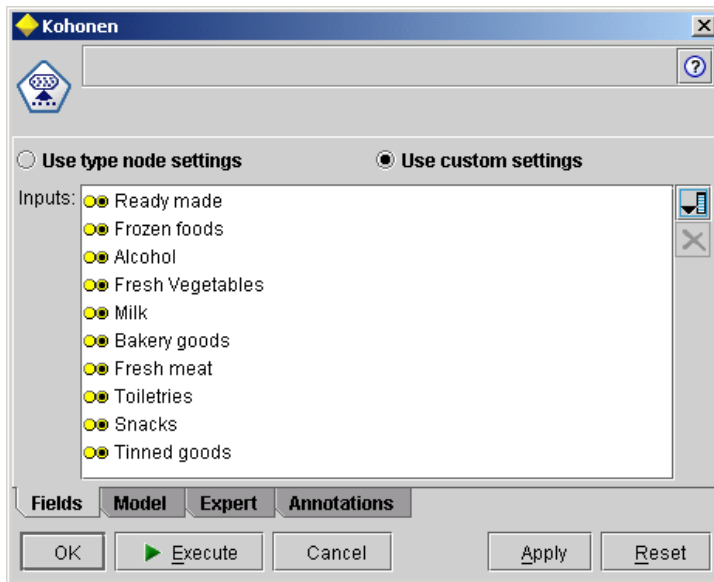
	Ready made	Frozen foods	Alcohol	Fresh Vegetables	Milk	Bakery goods	Fresh meat
1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	1	0	0	0	0	0	0
4	1	0	0	0	1	1	0
5	1	0	0	0	0	0	0
6	1	0	0	0	0	1	0
7	1	0	0	0	0	1	0
8	1	0	0	0	0	0	0
9	1	0	0	0	1	0	0
10	1	0	0	0	0	0	0
11	1	0	0	0	0	0	0
12	1	1	1	0	1	1	0
13	1	0	0	0	0	0	0
14	1	0	0	0	0	0	0
15	1	0	0	0	0	0	0
16	1	0	1	0	0	0	0
17	1	0	0	0	0	0	0
18	1	0	0	0	0	0	0

We want to segment the data based on purchasing behavior, and need to set all other fields to direction *None* so that they are not used in the Kohonen network. We could do this within the Types tab of the Var. File node or add a Type node. However, we can also select fields for analysis from within a modeling node and we will take this approach.

- Close the **Table** node
- Place a **Kohonen** node from the Modeling palette to the right of the **Var. File** node
- Connect the **Var. File** node to the **Kohonen** node
- Double-click the **Kohonen** node
- Click the **Fields** tab
- Click the **Use custom settings** option button
- Click the Field list  button, select **Ready made to Tinned Goods**, then click **OK**

We are now ready to train a Kohonen Network.

Figure 11.2 Selecting Fields for Modeling in the Fields Tab

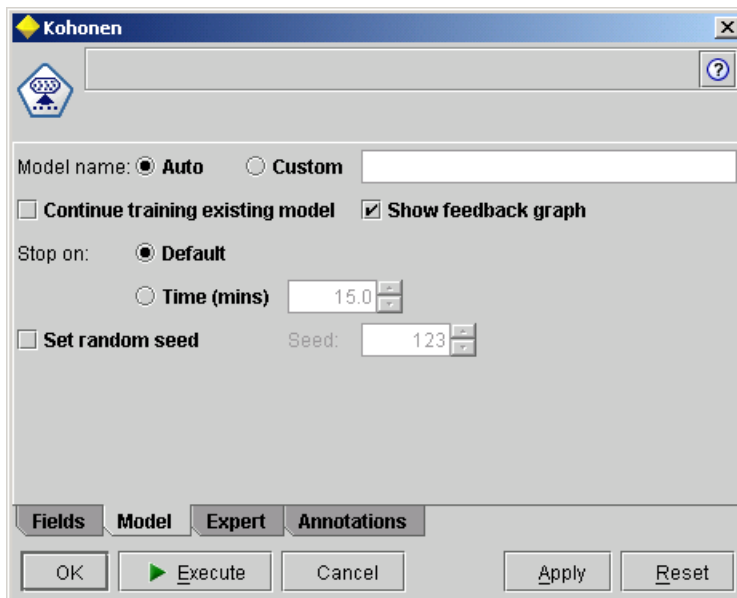


By default, the direction settings from the Type node or Types tab in a source node will determine which fields a modeling node will use. Here we have specified the fields directly in the modeling node. Modeling nodes that distinguish between input and output fields would contain two field lists.

Note that the purchase item fields are of flag type.

Click the **Model** tab

Figure 11.3 Kohonen Node Dialog



The name of the Kohonen network, as in other modeling nodes, can be specified in the *Custom Model name* text box.

By default, each time you execute a Kohonen node, a completely new network is created. If you select the *Continue training existing model* option, training continues with the last network successfully produced by the node.

The feedback graph appears while the network is training and gives information on the progress of the network. It represents a grid of output nodes. As a Kohonen node wins data records, its cell becomes a deeper shade of red. The deeper the shade, the greater is the number of records in the node (cluster). The size of the grid can be thought of as the maximum number of possible clusters required by the user. This can vary and is dependent on the particular application. The size of the grid can be set on the Expert tab.

The Kohonen network can be set to stop training either using the default (a built-in rule of thumb—see the *Clementine User's Guide*) or after a specified time has elapsed.

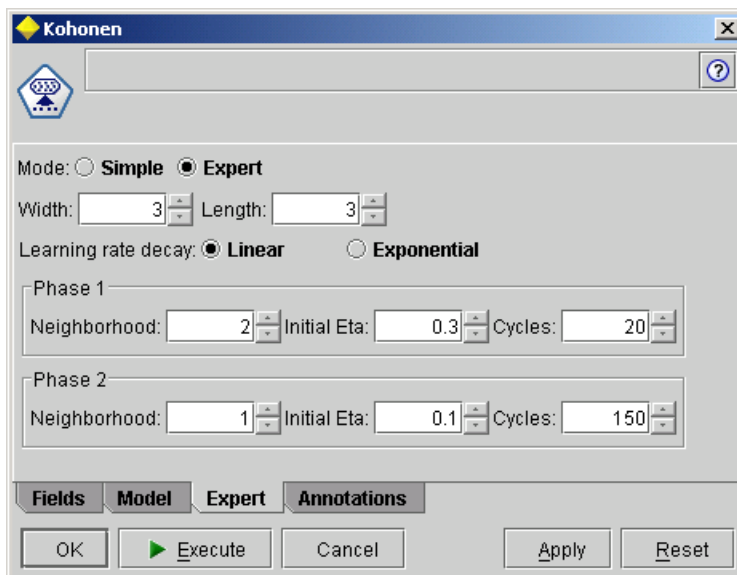
Similar to the Neural Net node, the *Set random seed* option is used to build models that can be reproduced. This is not normal practice and it is advisable to run Kohonen several times to ensure you obtain similar results using random starting points. For our purposes, however, we will fix the random seed value, so that the network in this guide can be reproduced.

Click the **Set random seed** check box
Type **1000** in the **Set random seed** text box

The Expert tab, when chosen, reveals additional settings and allows you to further refine the Kohonen network algorithm. In this chapter we will stay with most of the default settings. The only thing we want to change in the Expert tab is the size of the grid of output nodes to 3 by 3. Note that this would be a rather small topology (grid) for a typical Kohonen network application, but has the advantage of running quickly. (See the *Clementine User's Guide* or the *Advanced Modeling with Clementine* training course for information on other Expert settings.)

Click the **Expert** tab
Click the **Expert** Mode option button
Type **3** in the **Width:** text box
Type **3** in the **Length:** text box

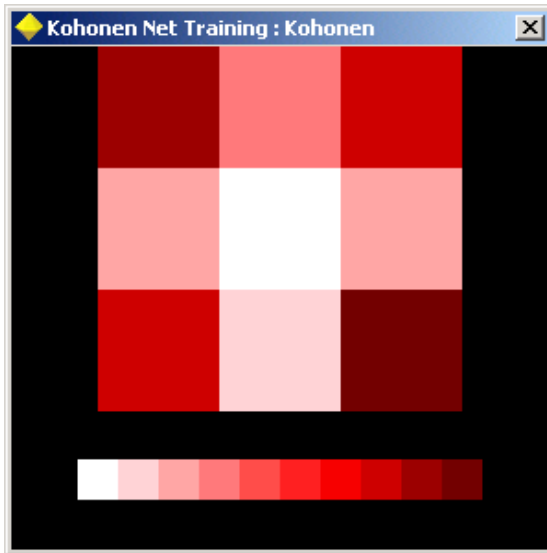
Figure 11.4 Kohonen Network Dialog (Expert Tab)



Click **Execute**

Clementine now passes the data through the Kohonen node and begins to train the network. A feedback graph similar to the one below will appear on the screen.

Figure 11.5 Feedback Graph When Training a Kohonen Network



Once the Kohonen node has finished training, the feedback graph disappears and a generated Kohonen model node appears in the Models palette of the Manager.

Browsing this node yields information describing the output nodes (clusters).

Now that we have produced a Kohonen network we will attempt to understand how the network has clustered the records.

Understanding the Kohonen Network

In this section we will interpret the profiles of the main segments produced by the Kohonen network.

The first step is to see how many distinct clusters the network has found, to assess whether all of them are useful— for example, should any of them should be discarded due to their containing small numbers of, or extreme, records? Note that when using Kohonen networks for fraud or error detection, there is great interest in the small or extreme clusters.

Right-click the **generated Kohonen node** in the Models manager, and then click

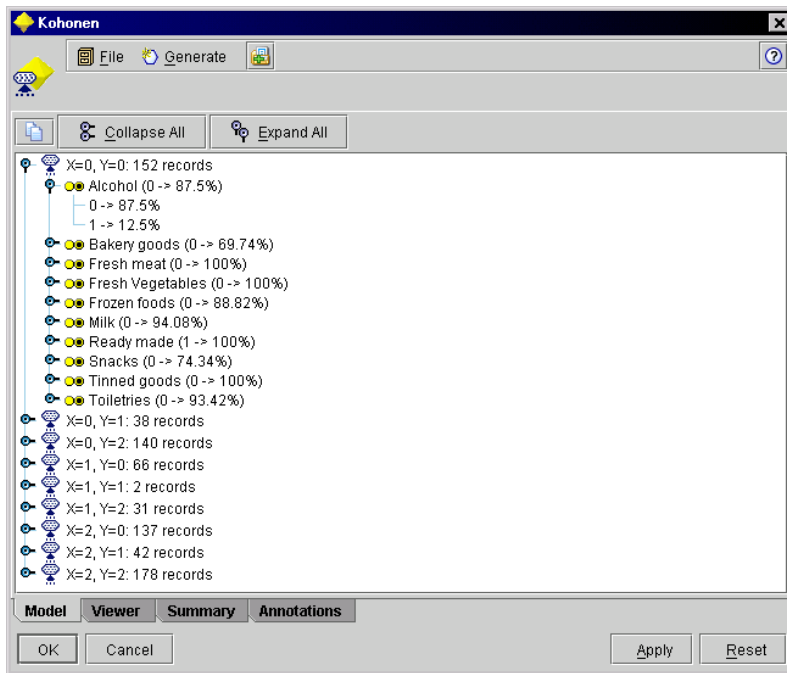
Browse

Click the **Model** tab

Click the expand button  for the **X=0, Y=0** cluster

Click the expand button  for **Alcohol** in the X=0, Y=0 cluster

Figure 11.6 Model Summary for Clusters

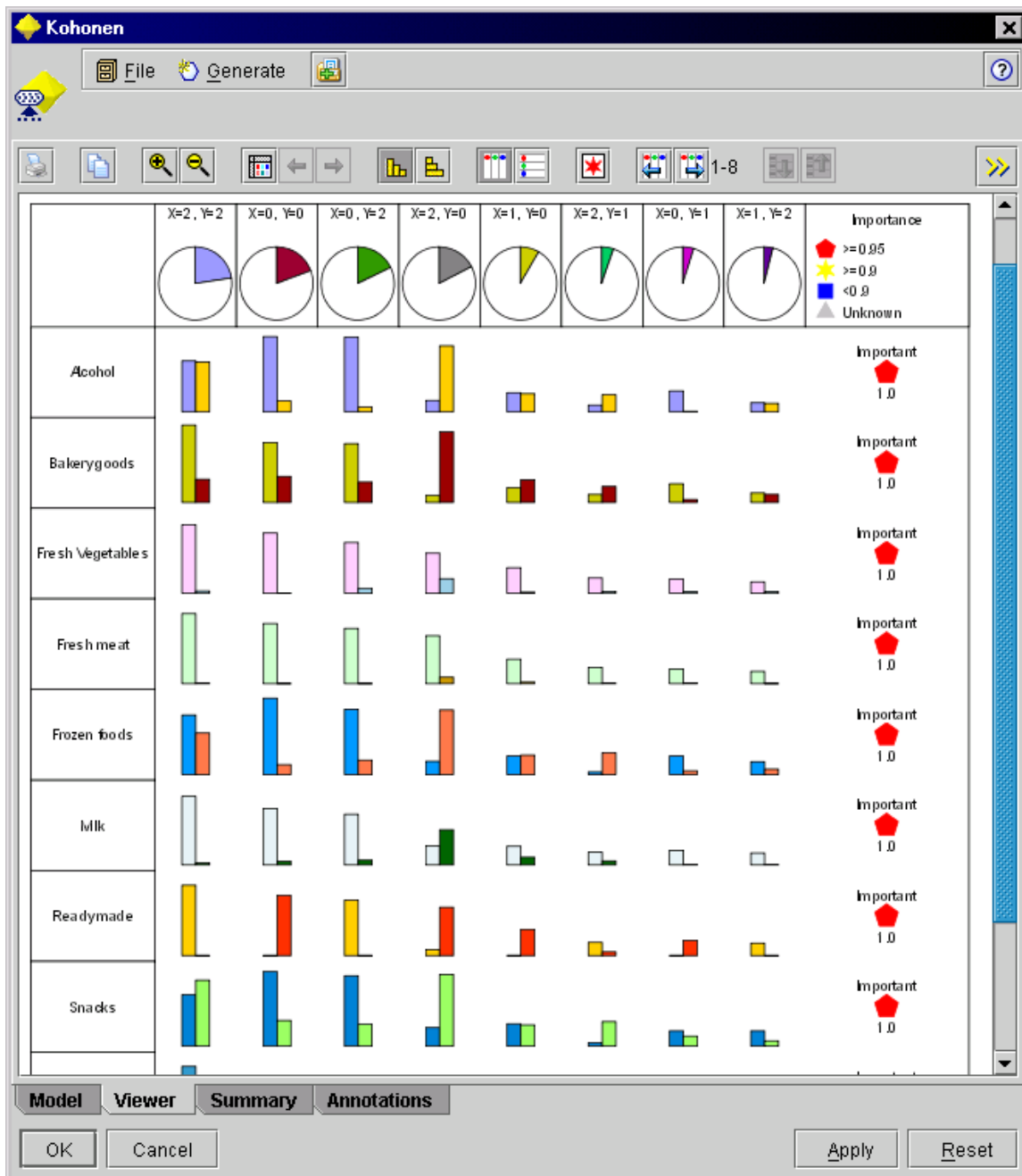


When a data stream passes through the generated model Kohonen node, two new fields are created, representing X- and Y-coordinates of the clusters. The clusters are identified in the Kohonen output window by their values on these coordinates.

The Model tab displays the number of records in each cluster (there are four relatively large clusters). For each cluster it also provides a profile presenting each input field. Since the fields are of type flag, the results present the most popular category (here 0 – not purchased, or 1 – purchased) along with the percent of records in a cluster with that value. By expanding an input variable, a more detailed breakdown appears that shows all categories and their percents (for set and flag fields; means would display for fields of type range). Although cluster descriptions could be developed from this summary, the Viewer tab presents cluster profiles graphically, which are easier to work with.


Click the **Viewer** tab

Figure 11.7 Cluster Viewer



The cluster viewer presents graphical profiles of the clusters. Clusters form the columns and the input fields form the rows in the display. By default, clusters are ordered by size, largest to smallest, and the input fields are ordered alphabetically.

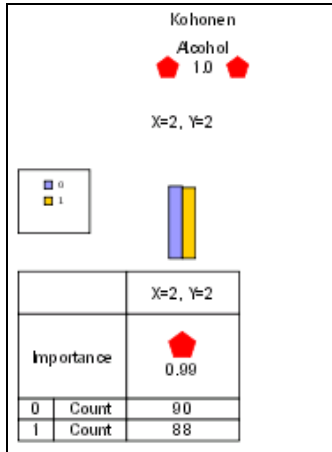
The pie charts in the top row indicate cluster size and each cluster is identified by its X and Y coordinates. Although the network has produced nine clusters (only 8 are visible; clicking the show next set of clusters

button  would advance to the remaining cluster), there appear to be four large clusters. Tool buttons allow you to place the cluster header in the row or column dimension and the bars can be oriented horizontally or vertically.

Each bar chart shows, for a given food and cluster, the distribution of no purchase (0) / purchase (1). For more details on any bar chart, just double-click on it.

Double-click on the bar chart for **Alcohol** in the first cluster column (**X=2, Y=2**)

Figure 11.8 Details for Alcohol Purchases in Cluster (X=2, Y=2)



The legend indicates that the first bar represents 0 (no purchase) and the second bar represents 1 (purchase). The counts appear in the table.

Click the **Back** button 

Bars in a column provide a profile of a cluster across the fields used in the Kohonen analysis. For example, looking down the first column, we see that members of cluster (X=2, Y=2) tend to purchase alcohol, frozen foods and snacks. Examining the profiles for the first four clusters in Figure 11.7, we might describe them as follows:

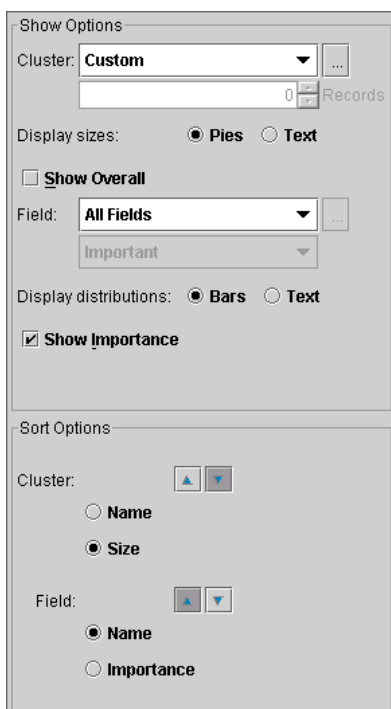
Cluster X=2, Y=2	This group is associated with Alcohol, Frozen foods, and Snacks.
Cluster X=0, Y=0	This group is mainly associated with Ready-made foods
Cluster X=0, Y=2	This group is strongly associated with Tinned goods.
Cluster X=2, Y=0	This group is associated with Alcohol, Bakery goods, Frozen foods, Ready-made, and Tinned goods.

We have started to build pictures of the four groups using the fields on which they were clustered. We will later use some of the other fields in the data set to build an expanded profile of each of the four groups.

The last column contains an *Importance* measure for each input field. It is based on a significance test (chi-square test of independence for symbolic inputs and t-test for range inputs) between clusters. It is calculated as $[1 - p(\text{significance value})]$ and so the more an input field differs between clusters, the closer to 1 is the importance value. The icons used to signify importance can be customized. When many fields are used to cluster data, the importance values can indicate which were more important in cluster formation. For this data, all the input fields had high importance values, but when some input fields have high importance values and others don't, you would place more weight on fields with higher importance when interpreting the cluster profiles.

We viewed the cluster profiles under the default settings (clusters ordered by size, fields ordered alphabetically, pie charts display cluster size, etc.). To control these settings:

Click the **Show Controls** button 

Figure 11.9 Options in Cluster Viewer

The *Show Options* group controls which clusters and input fields display, whether graphs or text is used to present the summaries, and whether importance values appear. The pie chart for cluster size and bar charts for input distributions can be replaced by text values. You have various options (Cluster and Field drop-down lists) concerning which clusters and input fields to display. For example, you can request that only clusters that exceed a certain size (or are smaller than a certain size) display, or that only fields meeting a specified importance criterion appear. The *Show Overall* option, when checked, will add a column corresponding to the overall data.

The *Sort Options* group controls the cluster and field order in the Viewer.

Click the Hide Controls button



Focusing on the Main Segments

The four main clusters are those labeled X=2, Y=2 (22.%), X=0, Y=0 (19.%), X=0, Y=2 (17.%), and X=2, Y=0 (17.%) [percents were found by clicking the Text option for Display Sizes in the Show Options group]. These clusters cover a total of about 77% of the records. It may be the case that the smaller groups are of interest (e.g. targeting a specialist product group). However, in this example we will concentrate on profiling the four largest groups. For this reason, we will generate a Select node to select only records from these four clusters. The easiest way to achieve this is to select the clusters in the Viewer and then to generate a Select node using the Generate menu.

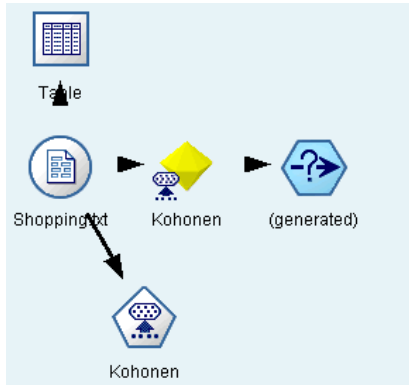
Click **cluster X=2, Y=2** and then shift-click **cluster X=2, Y=0** in the Viewer (to select those clusters)

Click **Generate..Select Node (from selection)**

Close the **generated Kohonen node** browse window

- Click on the **generated Kohonen** node in the Models manager, and then place it to the right of the **Var. File** node in the Stream Canvas
- Click and drag the **Select node [named (generated)]** from the upper-left corner of the Stream Canvas to the right of the **generated Kohonen** node
- Connect the **Var. File** node to the **generated Kohonen** node
- Connect the **generated Kohonen** node to the **Select** node

Figure 11.10 Selecting the Four Largest Clusters



Creating a Reference Value for Each Cluster Group

Currently, each Kohonen group (cluster) is identified by X and Y coefficient values. We need to create a single field in the data that denotes into which segment or cluster each record falls. To do this we concatenate the coordinate fields to form a two-digit reference number. This involves using a Derive node.

- Place a **Derive** node from the Field Ops palette to the right of the **Select** node
- Connect the **Select** node to the **Derive** node
- Double-click the **Derive** node
- Type **cluster** in the **Derive field** text box



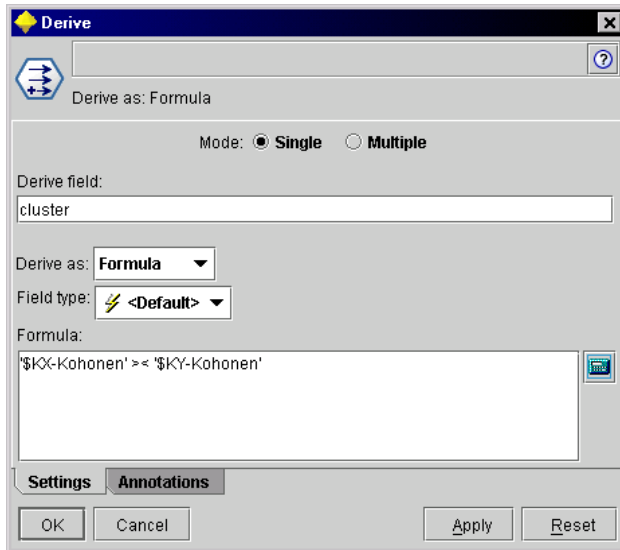
- Click the Expression builder button 
- Move **\$KX-Kohonen** from the **Fields** list to the Expression Builder text box
- Click the  (concatenate) button
- Move **\$KY-Kohonen** from the **Fields** list to the Expression Builder text box
- Click **OK**

Figure 11.11 Completed Derive Node to Create a Reference Number for Each Cluster



The formula contains the concatenate operator $><$ and the two cluster-coordinate fields. Remember that CLEM is case sensitive and field names beginning with \$ need to be enclosed in single quotes. The Expression Builder handles this automatically.

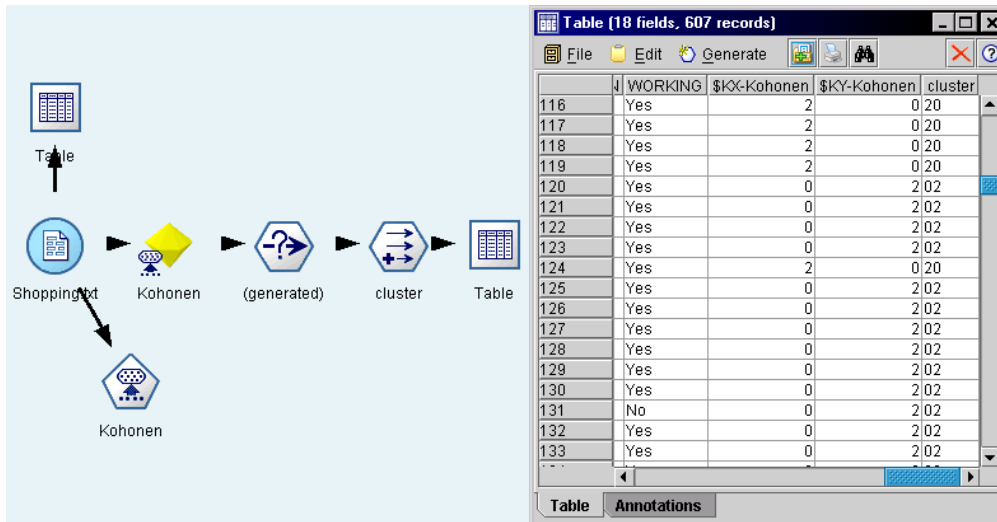
Click **OK** to return to the Stream Canvas

Place a **Table** node from the Output palette to the **right** of the Derive node named **cluster**

Connect the **Derive** node to the **Table** node

Right-click the **Table** node, then click **Execute**

Figure 11.12 Cluster Field Created Using the Derive Node



The resulting table contains a field called cluster that consists of a combination of the two coordinates. The new cluster field will now be used in overlay Distribution plots to describe the clusters in terms of demographic fields.

Using Other Fields to Build Profiles

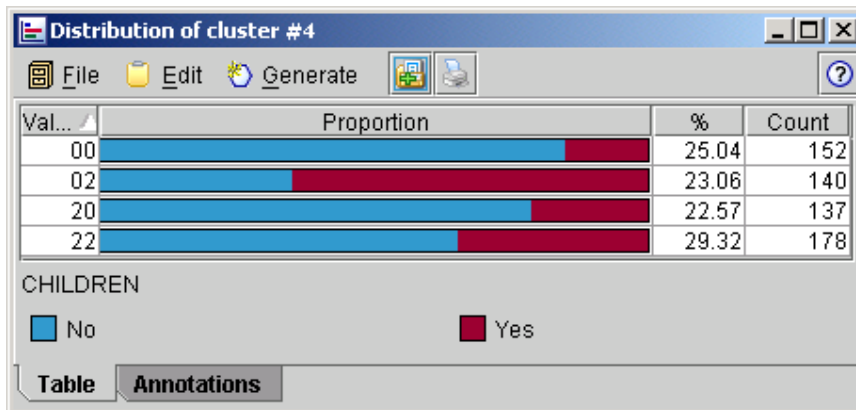
In this section we use a Distribution plot to investigate whether there are any relationships between the cluster groups and the basic demographics within the data set. This will help us to extend the profiles of the four groups.

- Place a **Distribution** node from the Graphs palette to the right of the Derive node named **cluster** in the Stream Canvas
- Connect the **Derive** node to the **Distribution** node
- Double-click the **Distribution** node
- Select **cluster** in the **Field:** field list
- Select **CHILDREN** in the **Overlay color:** field list
- Click the **Normalize by color** check box

We normalize the plot in order to view the proportions of the overlay field for each cluster.

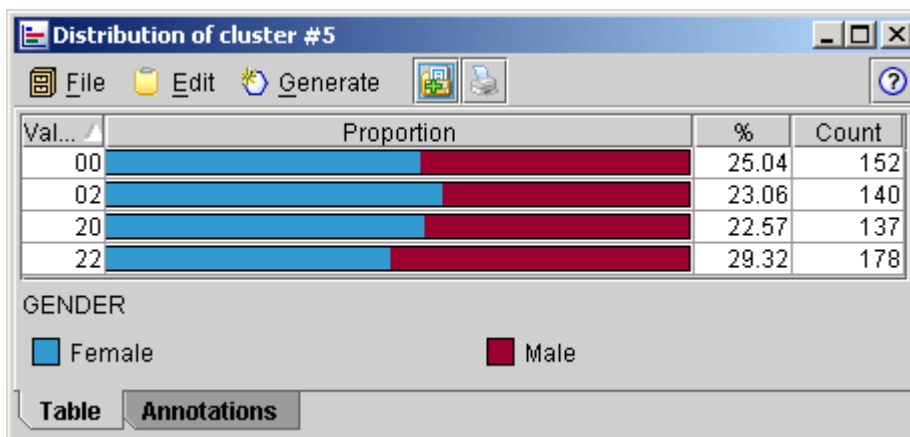
- Click **Execute**
- Repeat this process for all demographic fields of interest

Figure 11.13 Normalized Distribution of Clusters with Number of Children Overlay



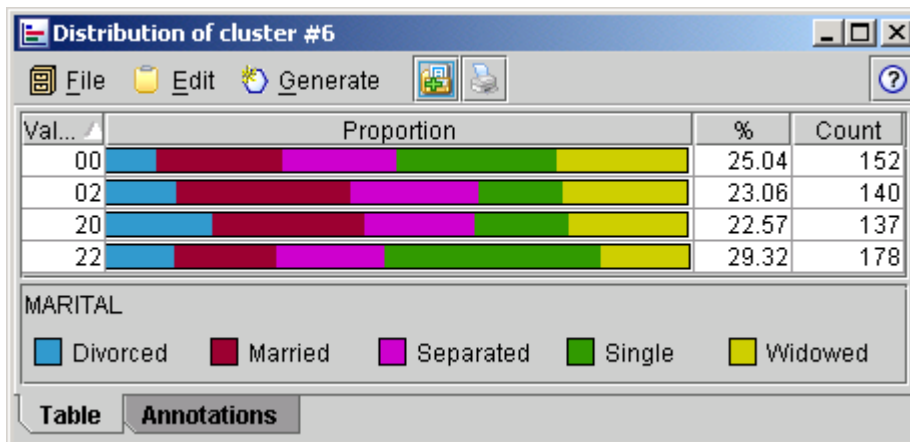
Cluster 00 consists of, in the main, individuals with no children and cluster 02 has the highest proportion of individuals with children.

Figure 11.14 Normalized Distribution of Clusters with Gender Overlay



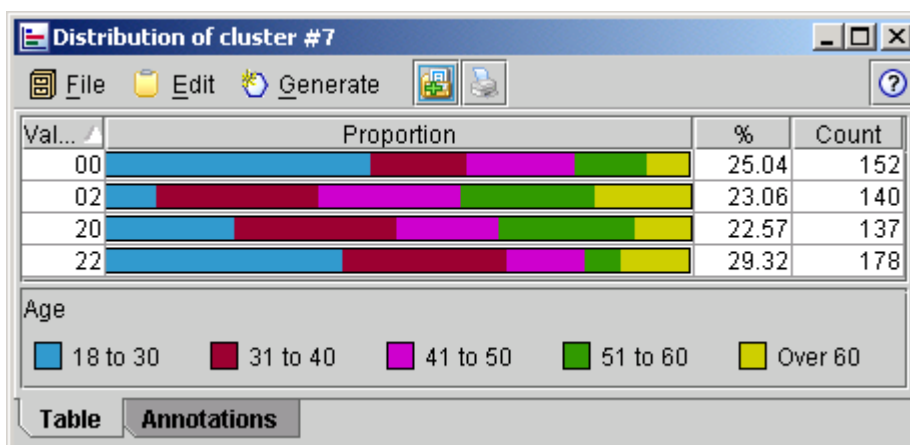
All clusters appear to have similar proportions of men and women. Cluster 02 has the most and cluster22 the fewest women.

Figure 11.15 Normalized Distribution of Clusters with Marital Status Overlay



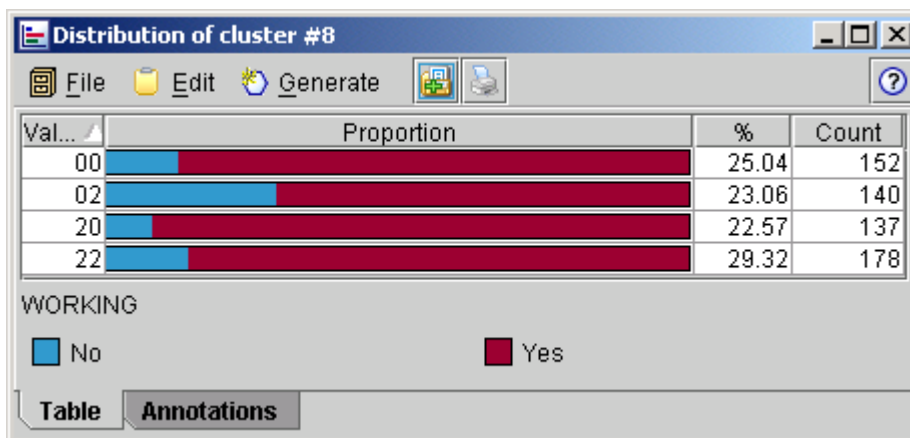
Cluster 20 has the highest proportion of divorced individuals; cluster 00 has the highest proportion of widowed individuals; cluster 22 has a large proportion of single individuals.

Figure 11.15 Normalized distribution of Clusters with Age Group Overlay



Clusters 00 and 22 have higher proportions of those 30 or under; the majority in cluster 22 are 40 or under; cluster 20 has a relatively even age distribution; cluster 02 has a large proportion of those over 30.

Figure 11.16 Normalized Distribution of Clusters with Work Status Overlay



Cluster 20 has the highest proportion of individuals who are working and cluster 02 has the highest proportion of individuals who are not working.

We can now finish our profiles of the four main segments in the data

- Cluster 00** This group, associated with purchasing ready-made foods, tends to be composed of those in the younger age groups (high proportion 30 or under), those who have no children, and those who are working.
- Cluster 02** This group is strongly associated with tinned goods. Almost all are older than 30. It has the highest proportion of those with children, the highest proportion of women, and also includes the highest proportion of individuals not working.
- Cluster 20** This group is associated with Alcohol, Bakery goods, Frozen foods, Ready made, and Tinned goods. It contains a high proportion of individuals with no children and the largest proportion of divorced individuals. This group has the highest proportion of working people.
- Cluster 22** This group is weakly connected to Alcohol, Frozen foods, and Snacks. This group has a high proportion of working people, those under 40, single people, and most are without children.

Summary

In this chapter you have been introduced to the basic capabilities of Kohonen networks within Clementine and interpreted cluster profiles using the cluster viewer and the Distribution node.

You should now be able to:

- Build a Kohonen network
- Use the Derive node to create a field containing each record's cluster membership
- Profile the clusters using the Cluster Viewer and Distribution nodes

Appendix: Viewing Clusters in a Scatterplot

When a Kohonen analysis is run, records are placed in nodes within a two-dimensional grid, which is why the clusters are identified by their X and Y coordinate values. In a large Kohonen network it might be of interest to see which clusters are close together and which are more isolated. This can be determined from the cluster viewer [noting each cluster's X/Y coordinate values], but a plot of the coordinate values is easy to do and provide a visual display. Since we ran a small topology (3 by 3), this isn't necessary for our analysis but would be helpful when larger topologies (for example, a 10 by 7) are run.

We are able to view the clusters with the Plot node.

Place a **Plot** node from the Graphs palette below the **generated Kohonen node** named **Kohonen**

Connect the **Kohonen** node to the **Plot** node

Double-click the **Plot** node

Recall that \$KX-Kohonen and \$KY-Kohonen represent the X and Y coordinates of the Kohonen network.

Select **\$KX-Kohonen** from the **X field:** Field list

Select **\$KY-Kohonen** in the **Y field:** Field list

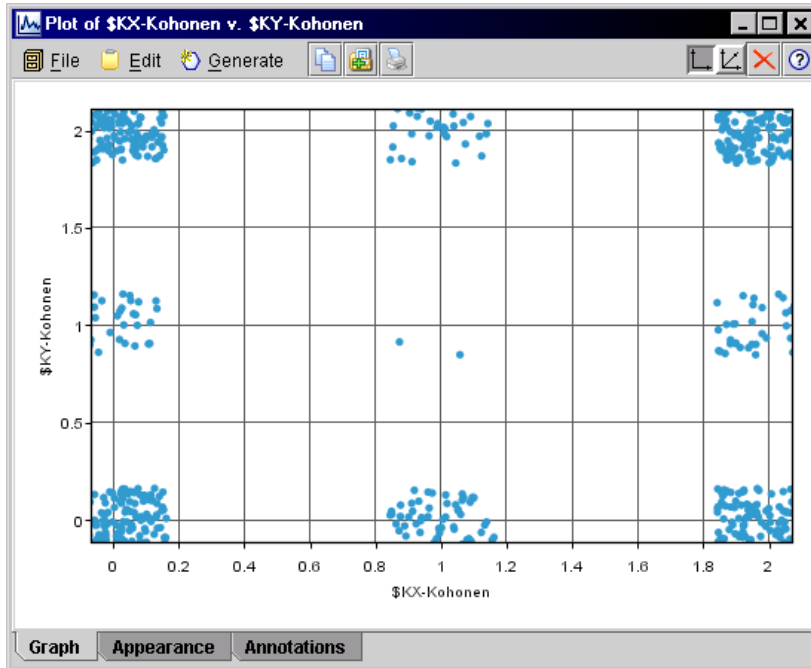
Because a large number of records will have identical coordinates, a small random component should be applied to the graph so that the cluster size can be viewed.

Click the **Options** tab

Type **0.3** in the **X field** and **Y field Agitation** text boxes (or use the spin control)

Click **Execute**

Figure 11.17 Plot of \$KX-Kohonen and \$KY-Kohonen with Agitation to Show Clusters



We see that although the network has produced nine clusters there are four large clusters. If the topology were larger, it would be of interest to see which clusters are nearby (similar) to which others.

As it stands, the four large clusters are well separated, given the limitations of the topology (3 by 3 grid).

Chapter 12

Association Rules

Overview

- Introduce two methods of generating association rules
- Use the Apriori node to build a set of association rules
- Interpret the results

Objectives

In this chapter we introduce how Clementine can provide a set of association rules using the GRI or the Apriori node. The Apriori node will be demonstrated and the resulting unrefined model will be browsed.

Data

In this chapter we will attempt a market basket analysis of a data set containing shopping information, *Shopping.txt*. The file contains fields that indicate whether or not during a customer visit, a customer purchased a particular product. Thus each record represents a store visit in which at least one product was purchased. The file also contains basic demographics, such as gender and age group.

Introduction

When people buy cigarettes do they tend to buy chocolate or beer? If people have high cholesterol do they also tend to have high blood pressure? If people buy car insurance do they also buy house insurance?

Answers to such questions can form the basis of brand positioning, advertising and even direct marketing. But how do we find whether associations such as these exist, and how can we begin to search for them when our databases have tens of thousands of records and many fields?

Association detection algorithms give rules describing which values of fields typically occur together. They can therefore be used as an approach to this area of data understanding.

Clementine contains two different algorithms that perform association detection: Generalized Rule Induction (GRI) and Apriori.

GRI searches for the most “interesting” (using a technical, information-theory based definition of interesting [the J measure]) independent rules in the data and tends to find them very quickly. One advantage to GRI is that numeric fields can be used as inputs.

Apriori has a slightly more efficient approach to association detection but has the limitation in that it only accepts symbolic fields as inputs. It also contains options that provide choice in the criterion measure used to guide rule generation. More detailed discussion of the GRI and Apriori algorithms is provided in the *Advanced Modeling with Clementine* training course and the *Clementine Algorithms Guide*.

Both procedures produce unrefined model nodes in the Models manager. These nodes, like the generated model nodes in neural networks and rule induction, can be browsed to view the set of association rules. However, they cannot be placed directly on the Stream Canvas or have data passed through them.

Association rules are presented in the format:

	Consequent	Antecedent1	Antecedent2	...	AntecedentN
Rule1
Rule2
...					
RuleR

For example:

Consequent	Antecedent1	Antecedent2
Newspapers	Fuel	Chocolate

Individuals who buy fuel and chocolate are likely to buy newspapers also.

When Clementine produces a set of association rules it also gives measures indicating the frequency and strength of the association for each rule. These measures are referred to as rule support and rule confidence and are given in the format:

Instances	Support	Confidence	Lift	Consequent	Antecedent1	Antecedent2
------------------	----------------	-------------------	-------------	-------------------	--------------------	--------------------

Instances is the number of records in the data set that match the antecedents.

Rule support is the percentage of records in the data set that match the antecedents.

Rule confidence is the percentage of all records matching the antecedents that also match the consequent.

Lift is the expected return using a model or rule. In this context it is the ratio of the rule confidence to the overall occurrence percentage of the consequent.

Therefore the full format of a rule will appear as:

Instances	Support	Confidence	Lift	Consequent	Antecedent1	Antecedent2
2051	15.0	71.0	2.00	Newspapers	Fuel	Chocolate

This means that 15% of the customers (2051 individuals in the data) bought fuel and chocolate. Of these 2051, 71% also bought newspapers. The lift value of 2.00 indicates that those who purchase fuel and chocolate are twice (2.00) as likely to buy newspapers as the overall sample (71.0% versus 35.5%[not shown]).

The Apriori Node

Since our data fields are of symbolic type, we will demonstrate the Apriori rule association detection algorithm.

The Apriori node is used to create a set of association rules and can be found in the Modeling palette.

On completion of modeling, the result will be an unrefined model node in the generated Models palette in the Manager window, which will be labeled with the number of fields used by the Apriori node. This unrefined model node contains the set of association rules and it can be browsed. Unlike the fully refined model nodes, it cannot be placed on the Stream Canvas or have data passed through it.

As with all the modeling nodes, Type declaration (via a source or Type node) must precede the Apriori node in the stream. Also, all field types must be fully instantiated. The Apriori, GRI and Sequence nodes are the only modeling nodes that recognize the direction setting of BOTH, since a field may appear as an antecedent and a consequent.

We will begin by opening a previously prepared Clementine stream file that contains a source node that reads the Shopping.txt data file, along with Type and Table nodes.

- Click **File..Open Stream**, move to the **c:\Train\ClemIntro** directory and double-click **Shoppingdef.str**
- Double-click the **Type** node
- Change the **Direction** setting for all **product fields (Ready made to Tinned goods)** to **Both** (select all product fields, right-click and choose **Set Direction..Both** from the context menu)
- Change the **Direction** setting for the **demographic fields** to **None** (select the fields, right-click and choose **Set Direction..None** from the context menu)

Figure 12.1 Type Node for Rule Association Modeling (Apriori)

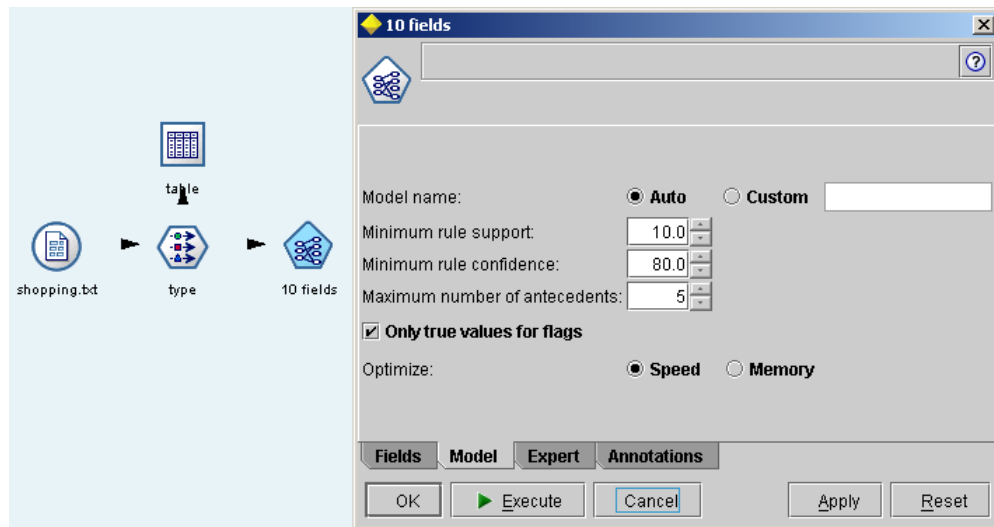


Notice that the shopping fields are set to flag type even they are coded 0,1. As in the Kohonen example earlier, we set their type to Discrete and then read the data. Otherwise, they would have type range, which isn't appropriate.

We are now ready to find a set of association rules using Apriori.

- Click **OK** to close the **Type** node
- Place the **Apriori** node from the Modeling palette to the right of the **Type** node
- Connect the **Type** node to the **Apriori** node
- Double-click the **Apriori** node named **10 fields**

Figure 12.2 Apriori Model Node Dialog



The default name of the resulting unrefined model is the number of fields. A new name can be entered in the Custom Model name text box.

Apriori will produce rules that, by default, have a minimum support of 10% of the sample, and a minimum confidence of 80%. These values can be changed using the spin controls. In practice there is some trial and error involved in finding useful values (too high and there are no rules generated; too low and there are many rules generated). Examining distribution plots on the fields to be analyzed provides base rates for the fields, which can provide some guidance in setting the minimum support value.

The maximum number of antecedents in a rule is set using the *Maximum number of antecedents* option. Increasing this value tends to increase processing time.

By default, rules will only contain the true value of fields that are defined as flags. This can be changed by deselecting the *Only true values for flags* check box. Since we are interested in rules describing what individuals purchase, rather than what they don't purchase, we will retain this setting.

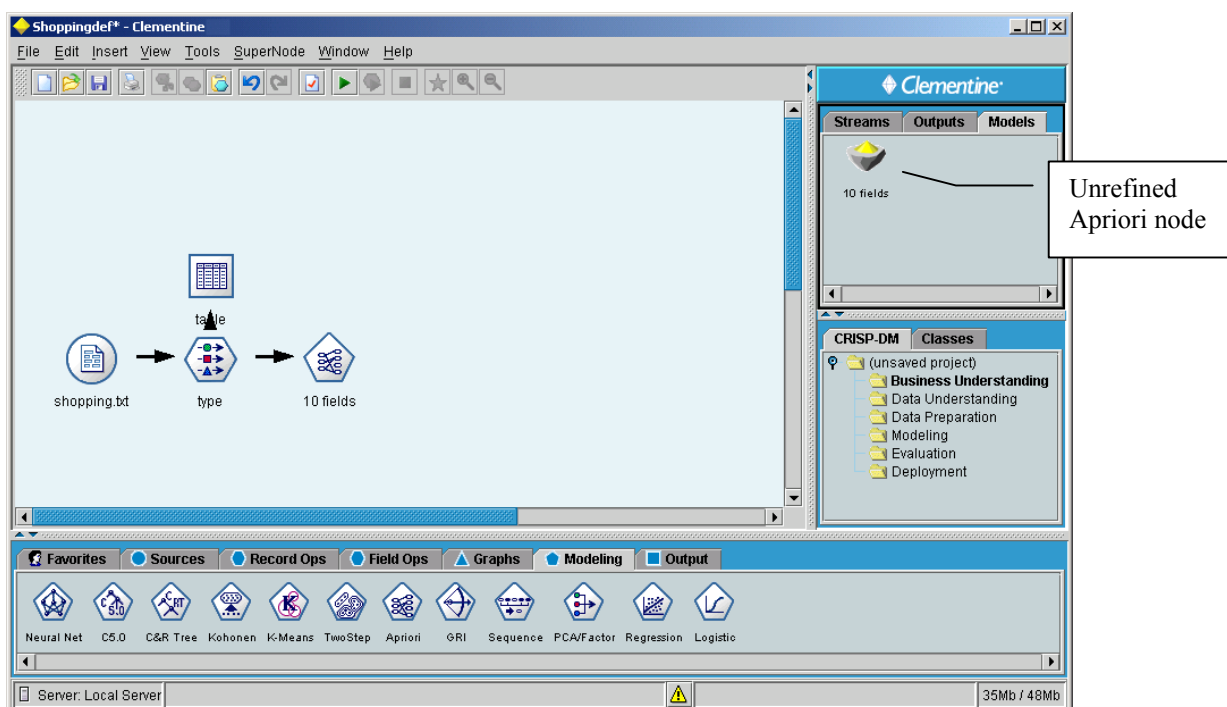
The algorithm can be optimized in terms of its *Speed* or its *Memory* usage.

Apriori searches for rules and discards rules that are not of interest. As in the other modeling tools, Expert options give the user greater control over the search. The reader is referred to the *Clementine User's Guide* and the *Advanced Modeling with Clementine* training course for more information on expert settings. In this example we will start with the default settings. We may find that the support and confidence values need to be adjusted.

Click **Execute**

An unrefined node will appear in the Models manager.

Figure 12.3 Unrefined Apriori Rules Node



Right-click the unrefined **Apriori** node in the Models palette of the Manager window, then click **Browse**



Click **Show/Hide criteria**  button, so support, confidence and lift values display

Figure 12.4 Browsing the Unrefined Model Generated by Apriori

The screenshot shows the '10 fields' model browser window. The 'Sort by' dropdown is set to 'Confidence' and the 'Sort by' button is active. The table below displays the generated association rules.

Instances	Support	Confidence	Lift	Consequent	Antecedent 1	Antecedent 2	Antecedent 3
85	10.800	83.500	1.948	Bakery goo...	Milk	Frozen foods	
95	12.100	83.200	1.940	Bakery goo...	Alcohol	Tinned goo...	Ready made
90	11.500	82.200	1.918	Bakery goo...	Frozen foods	Tinned goo...	Snacks
97	12.300	81.400	1.654	Ready made	Alcohol	Bakery goo...	Tinned goo...

At the bottom of the window, there are tabs for 'Model', 'Summary', and 'Annotations'.

The Apriori algorithm has found only four association rules. Rules can be sorted by *Support*, by *Confidence*, by the product of support and confidence ($Support \times Confidence$), alphabetically by *Consequent*, or by *Length* of the rule. Simply select the order using the *Sort by* drop-down list and the *Sort by* button  controls the direction of the sort.

The *Lift* value is the expected return resulting from using the model or rule. Here it is the ratio of the confidence to the base rate of the consequent. For example, bakery goods were purchased on 42.88% of the trips overall (can use a Distribution node to show this), but was purchased 83.5% of the time when milk, and frozen food were purchased. The lift from using this rule is $83.5/42.88$ or 1.948, and means that the chances of buying bakery goods almost double when mild and frozen foods are purchased.

The *Generate* menu allows you to generate a selection node for records that conform to a rule; just select the rule and choose *Select Node* from the *Generate* menu. A complete rule set for a specified consequent can also be generated.

Association rules can be saved as HTML, text, or PMML using the *File..Export* menu. This menu also has a print option.

We will now investigate whether, by dropping the confidence of the rules to 75%, we can obtain a larger number of associations.

Click **File..Close** to close the Apriori Association Rules Browser window

Double-click the **Apriori** modeling node

Enter **75** in the **Minimum rule confidence:** text box (type or use the spin control)

Click **Execute**

Right-click the unrefined **Apriori node** in the Models palette of the Manager window, then click **Browse**

Click the **Show/Hide criteria** button



Figure 12.5 Association Rules Generated by Apriori with Lower Minimum Confidence

Instances	Support	Confidence	Lift	Consequent	Antecedent 1	Antecedent 2	Antecedent 3	Antecedent 4
85	10.800	83.500	1.948	Bakery goo...	Milk	Frozen foods		
95	12.100	83.200	1.940	Bakery goo...	Alcohol	Tinned goo...	Ready made	
90	11.500	82.200	1.918	Bakery goo...	Frozen foods	Tinned goo...	Snacks	
97	12.300	81.400	1.654	Ready made	Alcohol	Bakery goo...	Tinned goo...	
91	11.600	79.100	1.607	Ready made	Alcohol	Tinned goo...	Snacks	
100	12.700	79.000	1.843	Bakery goo...	Milk	Tinned goo...		
105	13.400	79.000	1.844	Bakery goo...	Milk	Ready made		
90	11.500	78.900	1.840	Bakery goo...	Milk	Alcohol		
99	12.600	78.800	1.838	Bakery goo...	Frozen foods	Tinned goo...	Ready made	
79	10.100	78.500	1.594	Ready made	Milk	Bakery goo...	Tinned goo...	
95	12.100	77.900	1.817	Bakery goo...	Alcohol	Frozen foods	Tinned goo...	
90	11.500	77.800	1.580	Ready made	Milk	Alcohol		
85	10.800	77.600	1.969	Alcohol	Milk	Frozen foods		
98	12.500	77.600	1.809	Bakery goo...	Milk	Snacks		
79	10.100	77.200	1.921	Frozen foods	Alcohol	Bakery goo...	Tinned goo...	Ready made
79	10.100	77.200	1.627	Snacks	Alcohol	Bakery goo...	Tinned goo...	Ready made
97	12.300	76.300	1.898	Frozen foods	Alcohol	Bakery goo...	Tinned goo...	
80	10.200	76.300	1.674	Tinned goo...	Alcohol	Bakery goo...	Snacks	Ready made
100	12.700	76.000	1.927	Alcohol	Frozen foods	Snacks	Ready made	
91	11.600	75.800	1.768	Bakery goo...	Alcohol	Tinned goo...	Snacks	
95	12.100	75.800	1.597	Snacks	Alcohol	Tinned goo...	Ready made	
110	14.000	75.500	1.532	Ready made	Milk	Bakery goo...		
106	13.500	75.500	1.760	Bakery goo...	Alcohol	Snacks	Ready made	
85	10.800	75.300	1.653	Tinned goo...	Milk	Frozen foods		
85	10.800	75.300	1.529	Ready made	Milk	Frozen foods		
109	13.900	75.200	1.907	Alcohol	Frozen foods	Bakery goo...	Ready made	

This is a far richer set of associations. The first three rules (most confident rules- over 82%) involve the same consequent, bakery goods, with support of approximately 11-12%. The challenge is now to examine the rules for those that might be useful in the context of your business or research.

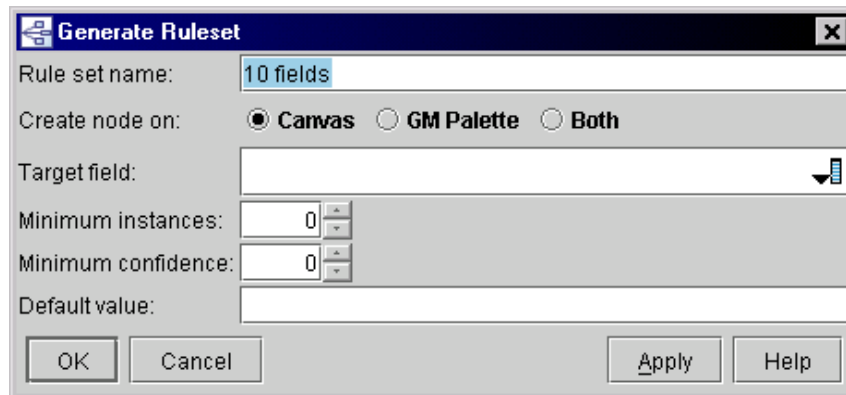
Using the Associations

A limitation of Apriori and GRI is that they do not produce model nodes that perform operations on the data; that is, the unrefined model can be browsed but cannot have data passed through it.

However, a rule set, similar to that of the rule induction models, can be created for a chosen conclusion by selecting the Rule Set option under the Generate menu of the Association Rules browser window. This will generate a new, fully refined model in the Models palette of the Manager window. This model, when placed in a data stream, will create a field indicating whether a rule in the rule set applies to the record and its confidence. Note that more than a single rule may apply to a record and, by default, the confidence value is based on the first rule whose conditions match the record. We will create a rule set for the *Alcohol* field.

Click **Generate..Rule Set**

Figure 12.6 Generate Ruleset Dialog



A model node, labeled by the name in the *Rule set name* text box, can be created on the Stream Canvas (*Canvas*), the Models palette (*GM Palette*), or *Both*.

A new association rule set node for the *Target field* will be created. This node will contain all rules with the specified *Target field* as the conclusion. When the data stream is passed through the generated Rule Set node, it will create a new field in the data recording whether a record has met the conditions for one or more of the rules in the rule set.

You may specify a *Default value* for the rule set (value if no rule applies to a record), as well as *Minimum support* and *Minimum confidence* for the rules.

- Type **Alcohol** in the **Rule set name** text box
- Select **Alcohol** in the **Target Field** list
- Type **0** in the **Default value:** text box
- Click **OK**
- Click **File..Close** to close the Rule browser window

A generated Apriori Rule Set node named *Alcohol* will appear in the upper left corner of the Stream Canvas.


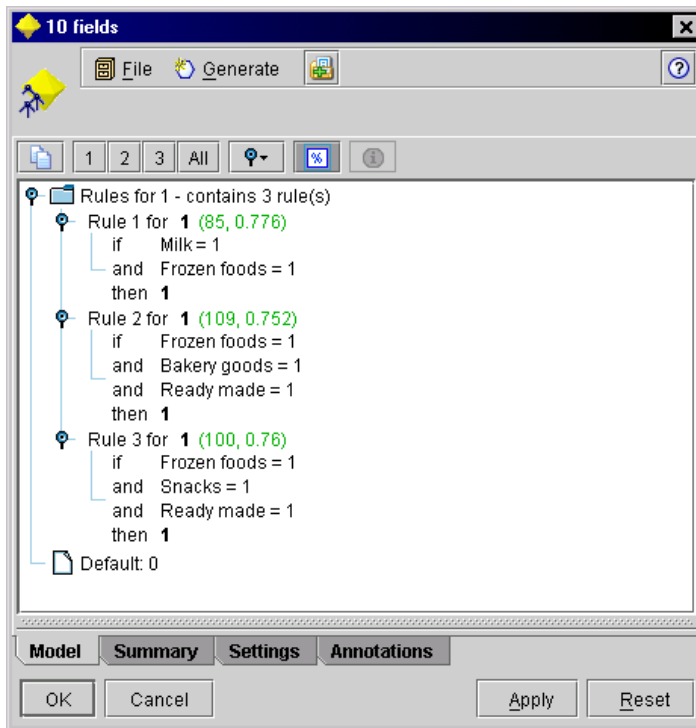
- Double-click the generated **Apriori Rule Set** node named **Alcohol** in the Stream Canvas
- Click the **All** button
- Click the **Show or hide instance and confidence figures**  button

Figure 12.7 Fully Unfolded Rule Set Generated from Apriori

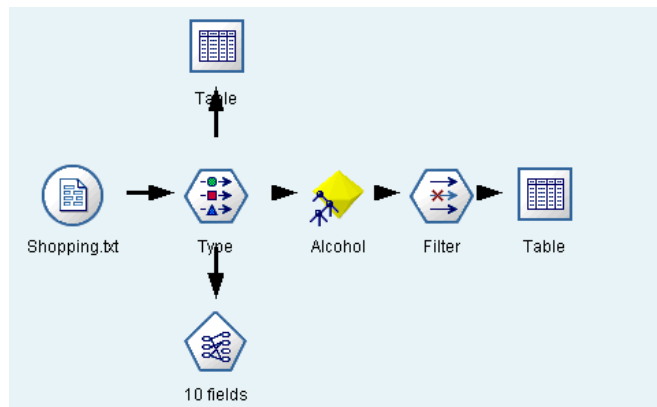


In this example the rule set contains three rules whose consequent is buying Alcohol. The first associates the purchase of milk and frozen foods. The second associates frozen foods, bakery goods and ready-made meals. The third associates frozen foods, snacks and ready made-meals meals.

We can now pass the data through this node and generate a field indicating whether or not a record complies with the conditions of any of the three rules.

- Click **OK** to close the Rule Set Browser window
- Drag the Apriori modeling node named **10 fields** below the **Type** node
- Drag the **Apriori Rule Set** node named **Alcohol** to the right of the **Type** node
- Connect the **Type** node to the **Apriori Rule Set** node named **Alcohol**
- Place a **Filter** node from the Field Ops palette to the right of the **Apriori Rule Set** node named **Alcohol**, and connect the **Apriori Rule Set** node to it
- Edit the **Filter** node and **deselect** the fields that are not used in the rules, **Fresh Vegetables**, **Fresh Meat**, **Toiletries**, **Tinned Goods** and the **demographic fields**, then click **OK**
- Place a **Table** node from the Output palette to the right of the **Filter** node
- Connect the **Filter** node to the new **Table** node

Figure 12.8 Stream with Generated Rule Set



Right-click the new **Table** node, then click **Execute**
Scroll down in the table to find a row with **1** in the **\$A-Alcohol** column

Figure 12.9 Fields Created by the Generated Apriori Rule Set Node

	Ready made	Frozen foods	Alcohol	Milk	Bakery goods	Snacks	\$A-Alcohol	\$AC-Alcohol
1	1	0	0	0	0	1	0	0.500
2	1	0	0	0	0	0	0	0.500
3	1	0	0	0	0	1	0	0.500
4	1	0	0	1	1	0	0	0.500
5	1	0	0	0	0	0	0	0.500
6	1	0	0	0	1	1	0	0.500
7	1	0	0	0	1	0	0	0.500
8	1	0	0	0	0	0	0	0.500
9	1	0	0	1	0	1	0	0.500
10	1	0	0	0	0	0	0	0.500
11	1	0	0	0	0	0	0	0.500
12	1	1	1	1	1	1	1	0.763
13	1	0	0	0	0	0	0	0.500
14	1	0	0	0	0	1	0	0.500
15	1	0	0	0	0	0	0	0.500
16	1	0	1	0	0	0	0	0.500
17	1	0	0	0	0	0	0	0.500
18	1	0	0	0	0	1	0	0.500

Two new fields appear in the data table. The first, *\$A-Alcohol*, is 0 unless one of the three rules in the rule set applies to the record, in which case it has a value of 1. The second field, *\$AC-Alcohol*, represents the confidence figure for the rules decision. Notice that when the conditions of the rules do not apply to a record, its confidence value is .5.

Extension: Exporting Model Values and Exporting Rule Sets as SQL

If you wish to pass model values (predictions, confidence values, cluster groups) to other programs, you can easily do so using the Flat File, SPSS Export, or SAS Export nodes in the Output palette. In addition, the Clementine Solution Publisher contains options to export scores to databases. Finally, SQL can be generated from the rule set (use the Settings tab in the generated Ruleset node) and applied to a database.

Summary

In this chapter you have been introduced to association rule detection within Clementine.

You should now be able to:

- Create a set of association rules using Apriori
- View the resulting rules by browsing the unrefined model
- Understand the meaning of rule confidence, support, and lift
- Sort the rules based on different criteria
- Create a rule set and use this to identify those records whose conditions are related to a selected conclusion

Chapter 13

Sequence Detection

Overview

- Introduce sequence detection methods
- Use the Sequence node to find a set of common sequences
- Interpret the sequence rules and add sequence predictions to the stream

Objectives

In this chapter we introduce how Clementine can identify common sequences in time-ordered data and sequences that are associated with an event. The Sequence node will be demonstrated, results in the generated model node will be browsed, and predictions will be added to the stream

Data

In this chapter we will look for common sequences of diagnostic/repair steps needed to resolve problems with telecom service. When a customer reports a problem with telecom service, different tests and operations are performed to resolve the problem. In some cases only a single test is needed, but sometimes many steps are required. There is interest in identifying common sequences needed to resolve service problems, discovering any repeating patterns (repetition of steps), and identifying sequences that were related to failure to resolve the problem. Data codes are masked and values are simulated based on a customer analysis. The data are stored in the tab-separated file *Telrepair.txt*. The file contains three fields: an ID number corresponding to the problem report, a sequence code for each step taken during problem resolution, and a code representing the diagnostic/repair activity in the step. Code 90 represents the reporting of the original problem (all reports should begin with code 90, but not all do). Codes 210 and 299 are termination codes: code 210 indicates the problem was resolved, while code 299 indicates it was not successfully resolved. Codes between 100 and 195 represent different diagnostic/repair activities. The file contains information on 750 service problems.

Introduction

What are the most common sequences of web-clicks when visiting the web site of an online retailer? Does a pattern of retail purchases predict the future purchase of an item? If a manufactured part, insurance claim, or sales order must go through a number of steps to completion, what are the most common sequences and do any of these involve repeating steps? These questions involve looking for patterns in data that are time ordered. In Chapter 12 we discussed general association rules; here the event sequence is formally taken into account.

The Sequence node in Clementine performs sequence detection analysis. In addition, an algorithm add-on, CaprI, performs sequence detection using a different algorithm and provides greater flexibility in specifying the types of sequences you are interested in investigating. In this chapter we will use the

Sequence node to explore common sequences of diagnostic/repair steps taken when attempting to solve telecom service problems.

Results from a Sequence node analysis will be of the form:

	Consequent	Antecedent1	Antecedent2	...	AntecedentN
Rule1
Rule2
...					
RuleR

For example:

This is a similar format to Apriori and GRI, although for Sequence there is an ordering to the antecedents and consequent. A natural way of thinking about a sequence rule is shown below:

Antecedent1 > Antecedent2 > ... > AntecedentN => Consequent

For example:

Base and Regression Models > Advanced Models => Clementine

Individuals who buy SPSS Base and Regression Models and later buy Advanced Models, are likely to later buy Clementine.

The “and” indicates that the two items are members of an item set. Thus, “Base and Regression Models” means that both items were purchased at the same time, while Base is antecedent1 and Regression Models is antecedent2 implies that the purchase of SPSS Base preceded the purchase of Regression Models.

When the Sequence node produces a set of sequences, it provides evaluation measures similar to those we reviewed when we discussed association rules. The measures are called **support** and **confidence**. Support refers to the number or percentage of cases (where a case is linked to a unique ID number) to which the rule applies—that is, the number of cases for which the antecedents and consequent appear in the proper order. Confidence refers to that proportion of the cases to which the ordered antecedents apply that the consequent later follows.

These measures are presented in the following format:

Instances Support Confidence Consequent Antecedents

Thus the full Sequence format would appear as:

Instances	Support	Confidence	Consequent	Antecedent1	Antecedent2
48	.12	.60	Clementine	Base and Regression Models	Advanced Models

This means that 12% (48 individuals) of customers purchased SPSS Base and Regression Models at the same time, then later purchased the Advanced Models, and even later purchased Clementine. Of the customers who purchased Base and Regression Models, then Advanced Models, 60% later purchased Clementine.

Data Organization for Sequence Detection

The Sequence node (and CaprI) can analyze data in either of two formats. In the Tabular data format, each item is represented by a flag field coded to record the presence or absence of the item. In transactional data format, item values are stored in one or more content fields, usually defined as type set.

Consider the software transaction example used earlier, in which a customer first purchased SPSS Base and Regression Models, then later purchased Advanced Models, and then purchased Clementine.

It could appear in tabular data format as follows:

Customer	Date	Base	Regression	Adv Models	Clementine	...	Decision Time
101	Feb 2, 2001	T	T	F	F	...	F
101	May 1, 2002	F	F	T	F	...	F
101	Dec 31, 2002	F	F	F	T	...	F

The same sequence in transactional data format would be:

Customer	Date	Purchase
101	Feb 2, 2001	Base
101	Feb 2, 2001	Regression
101	May 1, 2002	Adv Models
101	Dec 31, 2002	Clementine

In tabular format, it is clear that SPSS Base and Regression Models were purchased together (they are treated as an item set). In transactional format, the same items would be treated as an item set if the date field were specified as a time field within the Sequence node.

Sequence Node Versus CaprI

Both the Sequence node and CaprI perform sequence detection analysis. The Sequence node is one of the algorithms included with Clementine, while CaprI is an algorithm add-on. The trick to sequence analysis is to find a quick, memory efficient, minimal data-pass way of determining the sequences. The Sequence node and CaprI use different algorithms to accomplish this. Both permit you to specify various criteria (for example- maximum sequence size, minimum support, minimum confidence) that control the sequence search.

There are some considerations that might help you choose between them for a specific application. The Sequence node permits you to create generated model nodes that can be used to identify sequences and produce predictions in other data files. CaprI has a more extensive set of controls that determine the types of sequences you want counted. For example, suppose “SPSS Base, SPSS Regression Models, Clementine” is an observed purchasing sequence. Then the sequence “SPSS Base, Clementine” would be considered a partial sequence, since it appears within the observed sequence. If *Partial Sequences Pruning* were requested, then the sequence “SPSS Base, Clementine” would not appear among the rules if “SPSS Base, SPSS Regression Models, Clementine” did appear. Such pruning options provide a greater degree of control over what will be counted as a sequence and presented. In addition, CaprI allows you to search for only sequences that start or end with certain items. This might be useful if you are primarily interested in searching for sequences that lead to a certain web page or result.

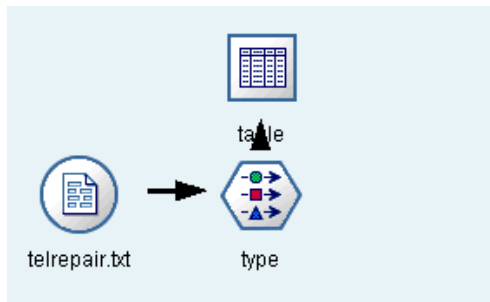
In short, both algorithms have advantages, which is why they are made available in Clementine (CaprI as an add-on algorithm).

The Sequence Node

We will load a previously defined stream to access the data and then add the Sequence node.

Click **File..Open Stream**, move to the **c:\Train\ClemIntro** directory and double-click on **Telrepairdef.str**

Figure 13.1 Telrepairdef Stream



Right-click the **Table** node, then click **Execute**

Figure 13.2 Telecom Repair Sequence Data

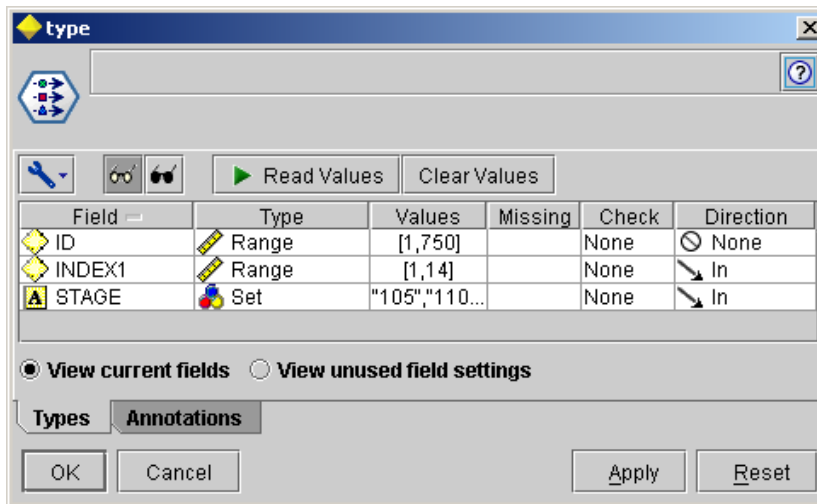
	ID	INDEX1	STAGE
1	1	1	90
2	1	2	170
3	1	3	110
4	1	4	150
5	1	5	120
6	1	6	125
7	1	7	180
8	1	8	195
9	1	9	210
10	2	1	90
11	2	2	110
12	2	3	105
13	2	4	180

Each service problem is identified by a unique ID value. The field INDEX1 records the sequence in which the diagnostic/repair steps were performed and the STAGE field contains the actual diagnostic/repair codes. All repair sequences should begin with code 90 and a successful repair has 210 as the final code (299 is used if the problem was not successfully resolved).

The data file is presorted by ID and by Index1 within ID. The Sequence node has an option to sort the data prior to analysis or the Sort node (located in the Record Ops palette) could be used.

Close the **Table** window
Double-click the **Type** node

Figure 13.3 Type Node for Sequence Detection Modeling



Even though numeric codes are used for the diagnostic/repair values in *Stage*, this field is declared as type set. This was done to emphasize that the STAGE values represent categories: different diagnostic/testing steps by the sequence detection algorithm. [In order to accomplish this, the storage for Stage was overridden and set to string in the Var. File node.] The analysis could be run with STAGE having type range and if there were a large number of distinct values in the STAGE field, then declaring it as type range would make the stream more efficient. Even if the content field were type range, the sequence algorithm would treat values as categorical; that is, 90 as 95 would be treated as two categories, not as similar numeric values.

In this analysis the content to be analyzed is contained in a single field: STAGE. The field(s) containing the content can be of direction *In*, *Out*, or *Both*. If there are multiple content fields, they all must be the same type.

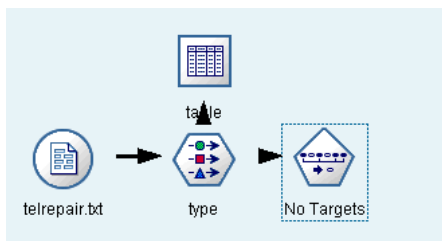
The field (here ID) that identifies the unit of analysis can be either numeric or symbolic type and can have any direction – here it is set to *None*.

Close the **Type** node

Place the **Sequence** node from the Modeling palette to the right of the **Type** node

Connect the **Type** node to the **Sequence** node

Figure 13.4 Sequence Node Added to Stream



Double-click the **Sequence** node

Select **ID** in the **ID** field box

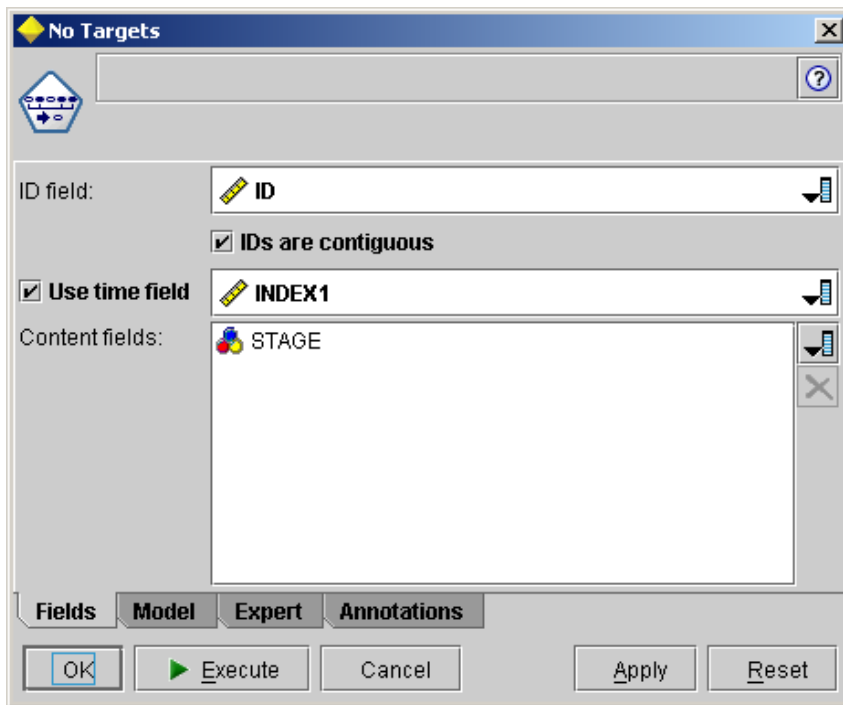
Click **Use time field** check box (so it is checked)

Select **Index1** in the **Use time field** box

Select **Stage** in the **Content fields** box

Click the **IDs are contiguous** checkbox

Figure 13.5 Sequence Node Dialog



By default, the model node is named after the ID field and you can change this in the Annotations tab of the Sequence dialog. The ID field defines the basic unit of analysis for the Sequence node. In our example, the analysis unit is the service problem and each problem has a unique value for the field named ID.

A time field is not required and if no time field is specified, the data are assumed to be time ordered for a given ID. We indicate INDEX1 is the time field for this analysis, although since the data records are ordered by INDEX1 for each ID, this is not necessary. Under expert options (Expert tab), you have additional controls based on the time field (for example, an event occurring more than a user-specified interval since an event can be considered to begin a new sequence).

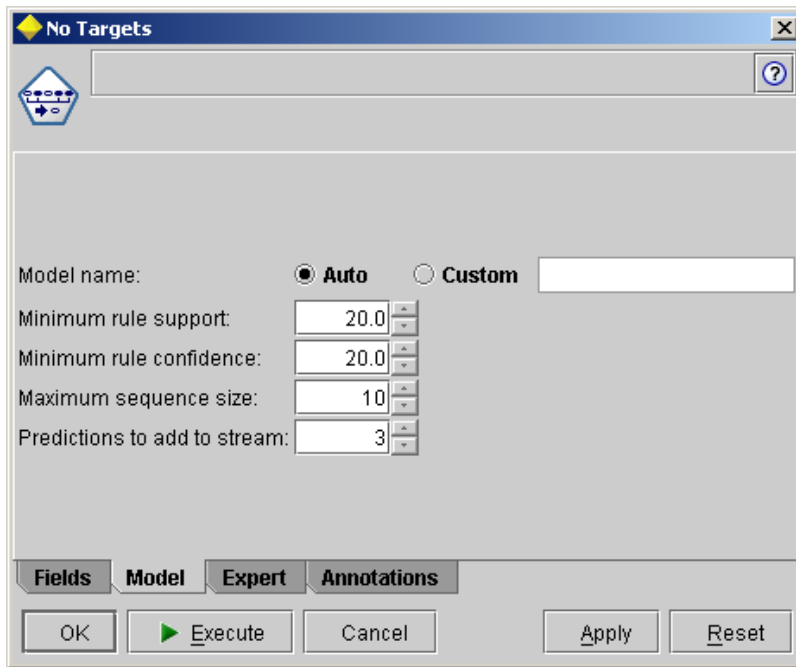
The *Content fields* contain the values that constitute the sequences. In our example, the content is stored in a single field, but multiple fields can be analyzed.

If the data records are already sorted so that all records for an ID are contiguous, you can check the *IDs are contiguous* check box, in which case the Sequence node will not resort the data, saving resources.

Model options provide greater control over various aspects of the analysis. We illustrate these options by examining the Support and Confidence controls.

Click **Model** tab

Figure 13.6 Sequence Node Dialog: Model Tab




We see that the Model options allow us to set the *Minimum rule support* (default 20%) and *Minimum Rule Confidence* (20%) values for sequences. It is useful to be aware of these values, since as with association rules, depending on the number and distribution of data values, the default settings might produce too many or too few sequences.

Expert options are discussed in the *Clementine User's Guide* and the *Advanced Modeling with Clementine* course.

Click **Execute**

Exploring Sequences

When the sequence detection analysis is complete, a generated sequence ruleset node  will appear in the Models tab of the Manager.

Right-click on the **generated Sequence ruleset node** in the Models tab of the Manager, and then click **Browse** on the Context menu

Click **Show/Hide Criteria** button 

If necessary, **change** the **column widths** to better read the item values

Figure 13.7 Sequence Rule Sets

Instances	Support	Confidence	Consequent	Antecedent 1	Antecedent 2	Antecedent 3
173	0.231	0.983	210	125	195	
163	0.217	0.982	210	90	125	195
156	0.208	0.969	210	180	195	
153	0.204	0.968	210	90	180	195
150	0.200	0.968	210	90	110	120
265	0.353	0.967	210	90	120	
250	0.333	0.965	210	90	105	
275	0.367	0.965	210	120		
162	0.216	0.964	210	125	180	
188	0.251	0.964	210	90	110	195
212	0.283	0.964	210	90	110	125
158	0.211	0.963	210	125	170	
157	0.209	0.963	210	90	125	180
260	0.347	0.963	210	105		
155	0.207	0.963	210	125	110	
154	0.205	0.962	210	90	125	170
152	0.203	0.962	210	110	120	

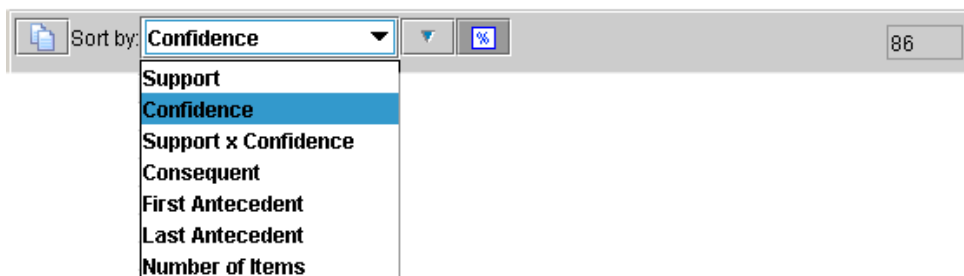
The Sequence node found 86 rules, which are presented in descending order by rule confidence. The second rule “90 > 125 > 195 => 210” is a sequence that begins with code 90 (all actual repair/diagnostic sequences should start with 90), followed sometime later by code 125, then later by code 195, and then later by code 210 (successful resolution). This sequence was found for 163 IDs, which constitute 21.7% of the IDs (there are 750 IDs in the data). These are the support values. Thus almost one fourth of all service problems in the data showed this pattern. Of the cases containing the sequence “90 > 125 > 195”, code 210 occurred later in 98.2% of these instances (confidence).

Notice that codes 90 and 210 appear frequently in the rules. This is because almost all service problem sequences begin with 90 and end with 210. Someone with domain knowledge of this area could now examine the sequences to determine if there is anything interesting or unexpected – for example a sequence that should not occur given the nature of the diagnostic tests/repairs, or a repeating sequence.

The sequence rule sets are ordered by confidence value (descending order). To view the most common sequences, we simply sort by support value.

Click the **Sort by** drop-down list

Figure 13.8 Sort Options for Sequence Rule Sets



The sequence rules can be sorted in a number of ways. For those interested in sequences beginning or ending with a particular event (for example, clicking on a specific web-page), the sorts by first antecedent or consequent would be of interest. Notice that the sorts can be done in ascending or descending order.

Click **Support** on the **Sort by** drop-down list

Figure 13.9 Sequence Rule Sets Sorted by Support

Instances	Support	Confidence	Consequent	Antecedent 1	Antecedent 2	Antecedent 3
690	0.920	0.949	210	90		
492	0.656	0.677	110	90		
488	0.651	0.961	210	110		
477	0.636	0.656	125	90		
474	0.632	0.958	210	125		
473	0.631	0.961	210	90	110	
457	0.609	0.958	210	90	125	
436	0.581	0.600	180	90		
428	0.571	0.955	210	180		
417	0.556	0.956	210	90	180	
404	0.539	0.556	195	90		
403	0.537	0.960	210	195		
388	0.517	0.960	210	90	195	
283	0.377	0.389	170	90		
278	0.371	0.955	210	170		
275	0.367	0.965	210	120		
274	0.365	0.377	120	90		

Code 110 appears in two of the three most frequent rules. The sequence 90 followed by 210 occurs in about 92% of the service problems, which we would expect in a very high proportion of the sequences. Code 299, which indicates the problem was not resolved, has not appeared. This is because it is relatively infrequent (fortunately so, for the business and customers). If we were interested in sequences containing 299, we would have to lower the support to below 5%, which is the base rate for code 299. The exercises for this chapter perform some exploration of sequences in which the problem was not resolved.

A domain expert would be interested in the most frequent sequences, which describe the typical path a service problem follows. If some stages were more expensive or time consuming, they would attract particular attention. We will view the results in one other order.

Click **Number of Items** on the **Sort by** drop-down list
Scroll down to the **bottom** of the list (showing two-item sequences)

Figure 13.10 Sequence Rule Sets Sorted by Number of Items in Sequence

Instances	Support	Confidence	Consequent	Antecedent 1	Antecedent 2	Antecedent 3
239	0.319	0.329	150	90		
237	0.316	0.956	210	150		
225	0.300	0.443	125	110		
202	0.269	0.398	195	110		
191	0.255	0.376	180	110		
183	0.244	0.408	125	180		
176	0.235	0.356	195	125		
169	0.225	0.333	170	110		
168	0.224	0.339	180	125		
166	0.221	0.335	125	125		
164	0.219	0.331	170	125		
161	0.215	0.359	195	180		
161	0.215	0.325	110	125		
160	0.213	0.315	140	110		
158	0.211	0.311	120	110		
154	0.205	0.367	125	195		
154	0.205	0.303	105	110		
152	0.203	0.339	110	180		

The sequences are now sorted by the number of items. About ten lines from the bottom we find the sequence 125 => 125, which occurs in 22% of the sequences. This would be of interest, because, ideally, a diagnostic/repair stage should not be repeated. Someone familiar with the diagnostic/repair process would look into why this stage is repeating (erroneous test results at that stage, records not being forwarded properly, etc.) and modify the process to reduce it. Other repeating sequences may be present, but do not meet the minimum support and confidence criteria.

Model Predictions

Next we view the model predictions.

- Click the **Sequence generated model** node named ID in the Models tab of the Manager, then place it to the **right** of the **Type** node in the stream canvas
- Connect the **Type** node to the **Sequence generated model** node
- Place a **Table** node from the Output palette to the right of the **Sequence generated model** node
- Connect the **Sequence generated model** node to the **Table** node
- Execute** the **Table** node attached to the Sequence generated model node

Figure 13.11 Top Three Sequence Predictions

	ID	INDEX1	STAGE	\$S-ID-1	\$SC-ID-1	\$S-ID-2	\$SC-ID-2	\$S-ID-3	\$SC-ID-3
1	1	1	90	210	0.949	110	0.677	125	0.656
2	1	2	170	210	0.955	110	0.677	125	0.656
3	1	3	110	210	0.961	125	0.656	180	0.600
4	1	4	150	210	0.961	125	0.656	180	0.600
5	1	5	120	210	0.968	125	0.656	180	0.600
6	1	6	125	210	0.968	180	0.600	195	0.556
7	1	7	180	210	0.968	195	0.556	125	0.408
8	1	8	195	210	0.983	125	0.408	105	0.356
9	1	9	210	125	0.408	105	0.356	140	0.354
10	2	1	90	210	0.949	110	0.677	125	0.656
11	2	2	110	210	0.961	125	0.656	180	0.600
12	2	3	105	210	0.965	125	0.656	180	0.600
13	2	4	180	210	0.965	125	0.656	195	0.556
14	2	5	125	210	0.965	195	0.556	170	0.389
15	2	6	140	210	0.965	195	0.556	170	0.389
16	2	7	195	210	0.983	170	0.389	120	0.377
17	2	8	130	210	0.983	170	0.389	120	0.377
18	2	9	210	170	0.389	120	0.377	125	0.371
19	3	1	90	210	0.949	110	0.677	125	0.656
20	3	2	190	210	0.949	110	0.677	125	0.656

By default, the Sequence generated model node contains three prediction fields (prefixed with “\$S-”), containing the three most confident predictions of codes that will appear later in the sequence, predicted from the sequence observed to that point. The confidence value for each prediction is stored in a field prefixed with “\$SC-”.

The sequence value in the first record is stage 90 (for ID=1, Index1=1), which is the problem report. The most likely stage to occur later, given that stage 90 has occurred, is stage 210 with confidence .949. (Note: this rule can be seen in Figure 13.9.) Since most sequences end with stage 210, the second and third most confident predictions are, in some sense, more interesting for this analysis. Thus, the next most likely stage to occur later, given that stage 90 has occurred, is stage 110 with confidence .677. And the third most likely is stage 125. In this way, the three most confident future predictions, based on the observed sequence, are generated.

Examining the predictions for ID 1, notice that the most likely item to occur later can change as the observed sequence changes. This makes sense, since as more information becomes available about a sequence, additional rules can apply.

Extensions

Thus far we have explored the sequence rules and sequence predictions. The Generate menu in the rule browser window can be used to create supernodes (star-shaped nodes that encapsulate other nodes), which when added to the stream, can detect sequences, count sequences, and generate predictions. By default, this and the predictions from the Sequence generated rule node are done for the three most confident rules in the rule set, but this number can be increased in the Model tab of the Sequence node. Supernodes are discussed in more detail in the *Data Manipulation with Clementine* training course.

Summary

In this chapter you have been introduced to sequence detection within Clementine.

You should now be able to:

- Create a set of sequence rules using the Sequence node
- View the resulting sequences by browsing the generated model node
- Understand the meaning of confidence and support of the sequences
- Produce predictions using the Sequence generated model node

Chapter 14

Other Topics

Overview

- Introduce Clementine Server
- Introduce Clementine Solution Publisher
- Introduce CEMI
- Introduce Clementine Scripts
- Introduce Cleo
- Introduce Text Mining for Clementine
- Predictive Marketing and Predictive Web Analytics
- Suggestions for Improving Model Performance

Introduction

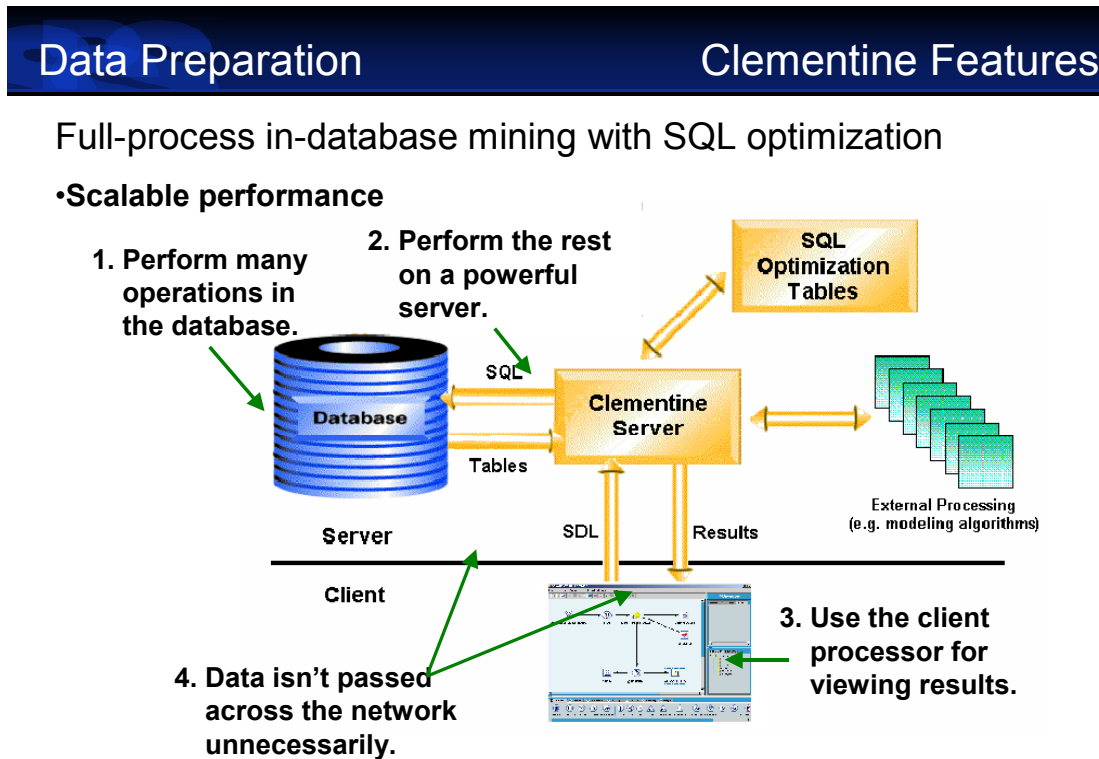
In this chapter we conclude your introduction to Clementine and data mining. We will suggest methods of improving the models you build. We will briefly introduce you to other Clementine products and features that can improve performance (Clementine Server), deploy model solutions (Clementine Solution Publisher, Cleo), and incorporate new algorithms into Clementine (CEMI). In addition, we mention other SPSS products and solutions that incorporate Clementine or its model scenarios (Text Mining for Clementine, PredictiveMarketing, Predictive Web Analytics).

We will also suggest ways to improve model performance.

Clementine Server

Clementine achieves scalability through a distributed architecture that consists of three tiers: the database, Clementine Server and the Clementine client.

Figure 14.1 Clementine Server Architecture



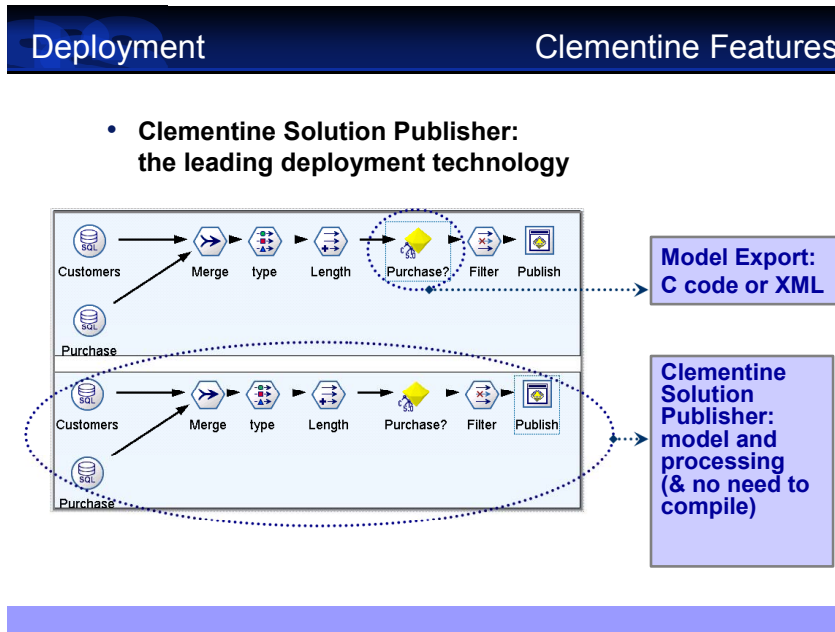
Clementine distributes processing to where it can be handled most efficiently. Many data operations such as aggregations, joins, etc. are performed (pushed back) in the database, where they are performed more efficiently.

Operations that cannot be expressed in SQL and therefore cannot be pushed back into the database are processed on a more powerful application server tier.

The client processor is only used for presentation of relevant results. For example, when viewing a table, only the rows that can be seen are “paged” down to the client.

Clementine Solution Publisher

Figure 14.2 Clementine Solution Publisher



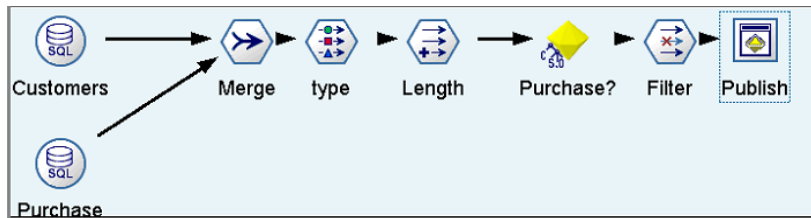
Clementine Solution Publisher is a deployment technology for delivering data mining solutions. Many data mining tools enable export of a model as code for use in custom applications. Clementine Solution Publisher exports the entire data mining stream, which often includes complex pre- and post-processing steps.

Processes for Publishing Solution require that you:

- Build the data mining process: First, you use Clementine to build a complete data mining solution—from data access and transformation to models and results.
- Publish the process: Add the Clementine Solution Publisher "node" to the stream, specify options and publish the data mining solution as files that will run with the Clementine Solutions Publisher executable.
- Build the application: Create an optional user interface to tailor the application for deployment for end users.
- Deploy the application: Deploy the data mining application throughout your organization. Score a database on a mainframe or other platform that Clementine doesn't support. Or, empower decision makers to make better informed decisions in the day-to-day tasks through a customized interface.
- Continually improve your solution: Finally, improve your solution by analyzing your results in operation. When you use the solution in your day-to-day activities, you create a closed-loop system that enables you to keep up-to-date with changes in your organization and in the market

Below we show a stream (derived from a rule induction model).

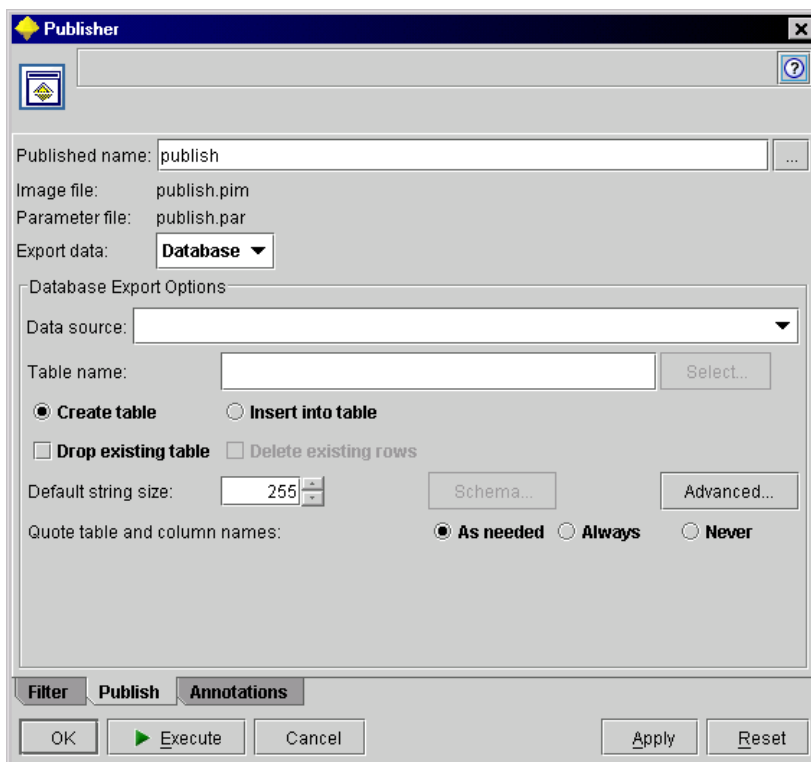
Figure 14.3 Clementine Stream with Solution Publisher Node



Although this stream is very simple, it contains the basic components needed for model deployment via the Clementine Solution Publisher: a data source node, a type specification (in Type node), a generated model node, and a Publisher node.

We examine the Publish node dialog below.

Figure 14.4 Publisher Node Dialog



The Publish node will create two files (with different extensions) using the name in the Published Name text box: an image file (*.pim) containing information about operations in the stream; and a parameter file (*.par) containing configurable information about data sources, output files, and execution options. The lower half of the dialog will vary depending on the format option chosen on the Export data drop-down menu. A database file will be written by default, but the Publisher node can also export the data as text, or write it in SPSS or SAS format.

The image and parameter files, created by the Publisher node, can be submitted later to the Clementine Runtime engine, which probably resides on a server, to perform scoring (calculating predicted and confidence values) on new records.

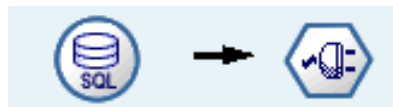
CEMI

The CEMI (pronounced CHIMMEY) is the Clementine External Modeling interface. Clementine External Module Interface is a mechanism for integrating external programs into Clementine.

Figure 14.5 CEMI Node



- **Need something else?**
- **Add new algorithms, data preparation, graphics, maps call other applications, etc ... even those that have been developed in-house**



CEMI gives you developer-like functionality without having to understand Clementine's internals. It allows integration of new and/or application-specific techniques such as: data processing, visualization, and modeling.

CEMI supports calling other programs from within the Clementine interface and sending data to these programs. Thus you can add to the algorithms supplied with Clementine. CEMI is documented in the *Clementine User's Guide*.

Clementine Scripts

Clementine Scripts is a programming language that can be used to modify and execute Clementine streams. For example, you might wish to run a series of Kohonen Networks in which the set of input fields vary for each model or a set of C5.0 models in which the severity of pruning varies. Clementine Scripts provide a way of passing parameters to nodes and otherwise modifying streams. Looping and conditional logic are supported within Clementine Scripts.

Scripts are discussed in the *Clementine User's Guide* and a sample script appears below.

Figure 14.6 Clementine Script

Clementine Stream Script

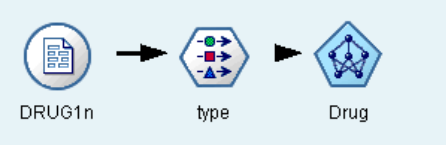
```

# Start with empty palette.
clear generated palette

# Now loop through all IN fields and
# train with each in turn set to NONE.
# Give models appropriate names.
for f in_fields_at type
  if type.direction.^f = 'IN'
    set type.direction.^f = 'NONE'
    set :trainnet.netname = 'omit_' >< f
    execute :trainnet
    set type.direction.^f = 'IN'
  endif
endfor

# Save the palette
save generated palette as multitrain.gen

```



Notes:

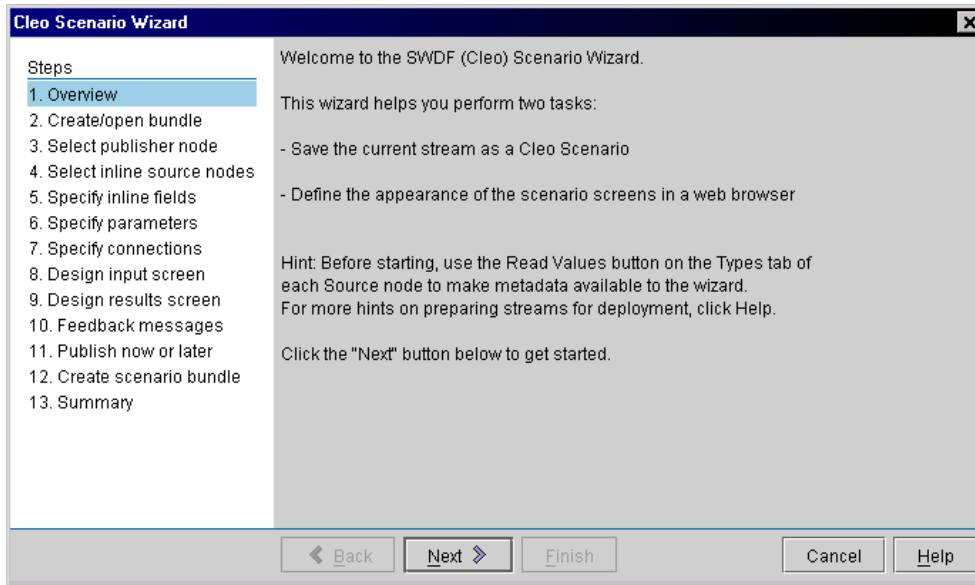
- For loop using `in_fields`.
- Loop applies to `IN` fields only..
- Train net node changes name so refer to it by `type` (there's only one).
- As usual, script "goes with" stream.

The script loops through the input fields in the `type` node. In each iteration a different input is excluded (direction set to None) and a neural network is run. Each generated model will have a unique name that indicates which input was excluded (for example `omit_age`). The results are retained because the generated model palette is saved. This is a very simply script, yet is serves to demonstrate the form of the language. (If you would like to examine this script and run it, open the *ScriptDrug.str* stream in the `C:\Train\IntroClem` directory, then click Tools..Stream Properties.. Script to view the script.)

Cleo

Clementine version 8 added to the Tools menu a Cleo Scenario Wizard, which creates Cleo scenario bundles from modeling streams. Scenarios can be used to publish models, run models on additional data, and score data from within a browser framework. The Cleo Scenario Wizard within Clementine steps you through the process of creating a scenario bundle for later use within Cleo.

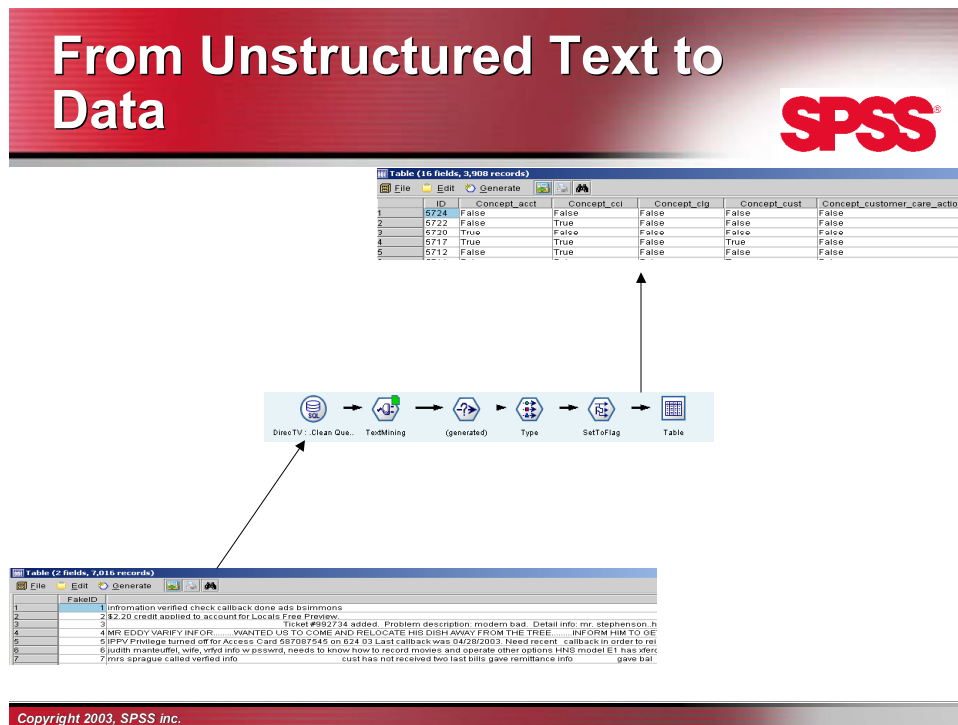
Figure 14.7 Cleo Scenario Wizard



Text Mining for Clementine

The Text Mining for Clementine product extends Clementine's capabilities by allowing it to incorporate information from unstructured text into predictive analytics. Using linguistic techniques, concepts are extracted from text (for example, notes taken during sales conversations, warranty reports, description fields in insurance claims, email or documents), converted into structured data (selected concepts are scored as True (present) or False (absent) from a document, that can be merged with other data (for example, customer-level sales records) and incorporated in Clementine analyses (predictive modeling, clustering).

Figure 14.8 Text mining for Clementine Stream



SPSS PredictiveMarketing and Predictive Web Analytics

SPSS PredictiveMarketing

SPSS PredictiveMarketing™ is an analytical tool for marketers that enables them to analyze customer data, find answers and build campaigns — without IT involvement. Marketers use SPSS PredictiveMarketing's built-in templates to get all of the benefits of powerful predictive analysis without waiting for reports from analysts.

Your organization gets:

- Improved cross-selling and up-selling: Discover which products to market together
- More effective customer retention programs: Identify which valuable customers are most likely to leave, and how to cost-effectively keep them
- Better-targeted campaigns: Find your most valuable customers and best prospects, and develop offers they'll respond to

Predictive Web Analytics

Web analytics + predictive modeling = better online understanding

Predictive Web Analytics combines SPSS Inc.'s enterprise Web analytics and data mining technologies to give you a complete picture of online activity.

NetGenesis enterprise Web analytics

NetGenesis is a Web analytics platform designed to manage the large data volumes produced by complex online businesses. The NetGenesis eDataMart, or customer data repository, supplies the comprehensive view of online customer activity you need to accurately predict behavior. The browser-based, customizable NetGenesis InfraLens reporting interface enables you to send reports and results to decision makers throughout your organization.

Clementine predictive modeling

Clementine's powerful analytical engine performs advanced predictive analysis on the customer data stored in the NetGenesis eDataMart. Clementine's predictive models incorporate your real-world business expertise to solve specific online business problems — such as detecting significant online activity sequences or converting online visitors to buyers.

Together, Clementine and NetGenesis bring the power of Predictive Web Analytics to online enterprises.

Suggestions for Improving Model Performance

When building models, it may be the case that they do not perform as well as we would like. The following paragraphs suggest possible ways in improving model performance:

- **Train and Validate:** Divide the sample into two sections. The first is used to train the model; the second is used to evaluate the performance of the model. This will help prevent the problem of over-fitting the training data. A model should be general enough that it predicts nearly as well to holdout data.
- **Balance the Data:** If the data file contains a significant imbalance in outcomes (for example, one outcome category occurs rarely), it can make it difficult to build predictive models that accurately predict the infrequent outcome. Balancing the data can help solve this problem. Clementine contains a Balance node that can be used to address this problem. The reader is referred to the Help system or the *Clementine User's Guide* for details on the Balance node.
- **Transform the Data:** Machine learning techniques (neural networks in particular) perform better when numeric fields have an even distribution of values. An uneven distribution of numeric outcomes can be transformed (with a Derive node) to produce a more even distribution.
- **Combine Modeling Methods:** Methods (for example neural network and rule induction methods) can be combined to help to refine models. Rules can help identify which fields are useful as inputs into the network. Also, the predictions from one model can be used as an input field to a different model, which might improve predictive accuracy.

All of the above points are purely suggestions and sometimes, due to the nature of the data, models simply cannot be built to a high degree of accuracy.

Demos and CATs

Clementine ships with a number of sample streams. Their descriptions appear in appendices within the *Clementine User's Guide* and the stream and data files are located in the Demos subdirectory (found under the Clementine version number folder). These streams provide additional application examples and we recommend you examine them for ideas on how to approach modeling problems.

For more detailed information on some popular applications, Clementine ships with Clementine Application Templates (CAT): currently, *Clementine Application Template for Analytical CRM in Telecommunications*, *Clementine Application Template for Web Mining*, *Clementine Application Template for CRM*, and *Clementine Application Template for Fraud, Waste and Abuse Detection* (additional option). Each CAT contains documentation (found in the documents directory within the Clementine application directory), along with many stream files (found in subdirectories Telco, Web, CRM, and Fraud, which can be reached by clicking File..Template Library) that step you through the application analysis. For those interested in web mining, CRM (Customer Relationship Management), telecom churn, or fraud detection applications the CATs provide detailed, working templates for the data organization, display, and modeling required for such projects. As such, they can be quite valuable to an analyst beginning one of these projects. However one should bear in mind that each CAT may be modified and applied to a range of tasks by people in many different industry sectors. We suggest you begin by examining the document (in Adobe Acrobat .pdf format) that accompanies a CAT and then begin to work with the streams. There is even a data-mapping tool (the Help system describes its use) that can be used to map your data fields to those used in the CAT streams. Finally, note that some of the streams in the Web Mining CAT use CapRI (a sequence analysis algorithm), which is a Clementine add-on provided as a separate product.

Data Mining References

Berry, Michael J. and G. Linoff. (1997) *Data Mining Techniques: For Marketing, Sales and Customer Support*. New York: Wiley.

Berry, Michael J. and G. Linoff (2000) *Mastering Data Mining: The Art and Science of Customer Relationship Management*. New York: Wiley.

Berry, Michael J. and G. Linoff. (2002) *Mining the Web: Transforming Customer Data into Customer Value*. New York: Wiley.

Berson, Alex and S. J. Smith. (1997) *Data Warehousing, Data Mining, & OLAP*. New York: McGraw Hill.

Fayyad, Usama M., Piatetsky-Shariro, G., Smyth, P. and R. Uthurusamy. (1996) *Advances in Knowledge Discovery and Data Mining*. Menlo Park, CA.; Cambridge, Mass: AAAI Press and MIT Press.

Han, Jiawei and M. Kamber. (2000) *Data Mining: Concepts and Techniques*. San Francisco: Morgan Kaufman.

Hand, David J. (1998) "Data Mining: Statistics and More?" *The American Statistician*. Vol. 52, No. 2.

Hastie, Trevor, Tibshirani, R. and J. H. Friedman (2001) *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York. Springer Verlag.

Heikki, Mannila, Smyth, P. and D. Hand. (2001) *Principles of Data Mining*. Boston, Mass. MIT Press.

SPSS Inc. (2002) *Data Mining with Confidence. 2nd Edition*. Chicago: SPSS Inc.

Westphal, Christopher and T. Blaxon. (1998) *Data Mining Solutions*. New York: Wiley.

Version 1.0 of the CRISP-DM (Cross-Industry Standard Process for Data Mining) process document can be downloaded from the following website: www.crisp-dm.org

Exercises

Note Concerning Data for this Course

Data for this course are assumed to be stored in the folder \Train\ClemIntro. At SPSS training centers, the data will be located in c:\Train\ClemIntro. If you are working on your own computer, the \Train\ClemIntro directory can be created on your machine and the data copied from the accompanying floppy disk or CD-ROM. Note that if you are running Clementine in distributed (Server) mode— see note about Clementine Server in Chapter 2— then the data should be copied to the server machine or the directory containing the data must be mapped from the server machine.

Chapter 2

Introducing SPSS Clementine

1. Start Clementine
2. Familiarize yourself with the Clementine environment – the menus and the help facilities. Search the help for any topics you are familiar with, or terms you have heard. Locate the help topic on the Select node and familiarize yourself with its operation.
3. Practice placing nodes on the Stream canvas.
4. Select the Var. File node from the Sources palette and place it on the Stream canvas.
5. Select the Table node from the Output palette and place it next to the Var. File node.
6. Connect these two nodes.
7. Edit the Var. File node and view the options.
8. Disconnect the two nodes.
9. Delete one of the nodes in the stream.
10. Exit Clementine without saving the stream.

Chapter 3

Reading in Data Files

In these exercises we will practice using the source nodes demonstrated in Chapter 3. The exercise data file is to be used throughout the course and exists in two formats; comma delimited (**charity.csv**) and SPSS data file (**charity.sav**). If possible, both are to be used in this session. The file originates from a charity and contains information on individuals who were mailed a promotion. The file contains details including whether the individuals responded to the campaign, their spending behavior with the charity and basic demographics such as age, gender and mosaic (cluster) group.

1. Start Clementine, if you haven't done so already, and ensure the Stream canvas is clear.
2. Select a Var. File node from the Sources palette and place it on the Stream canvas
3. Edit this node and set the file to C:\Train\ClemIntro\charity.csv. The file contains the field names in the first row, so check the option that instructs Clementine to read these from the file.
4. Return to the Stream canvas by clicking the OK button.
5. If present on the Sources palette, select the SPSS File node and place it also on the Stream canvas
6. Edit this node and set the file to c:\Train\ClemIntro\charity.sav. The SPSS data file contains value and variable labels. Check the options to use both of these.
7. Return to the Stream canvas by clicking the OK button.
8. To check that both source nodes are working correctly, connect a Table node to each.
9. Execute both streams individually using the Execute option in the Context (pop-up menu following a right click) menu.
10. Scroll across the tables and familiarize yourself with the fields in the data set. When you have finished close the Table windows (click File..Close).
11. We will use one of the streams in the following exercises. Choose which source node you prefer and delete the stream containing the other. Save the single stream within the c:\Train\ClemIntro directory under the name *ExerChapter 3.str*.

Chapter 4

Data Quality

In this session we will use the stream created in the previous exercises and check the integrity of the charity data file.

1. Load the stream you created and saved in the last exercise, *ExerChapter 3.str*, using the File..Open Stream menu choice.
2. Edit the Types tab in the source node and click Read Values to force type instantiation. Check that you agree with the chosen types. Change the types if necessary. Investigate the definitions for blanks for a few of the fields.
3. Attach a Quality node to the source node and execute this section of the stream. Are all the fields valid? Do you have any concerns with the data?
4. Attach a Data Audit node to the source node and examine the results for odd distributions and values. Give extra attention to fields related to pre- and post-campaign expenditure and visits.
5. Select one of the symbolic fields (sets or flags) and examine its distribution in more detail (double-click on the graph in the Data Audit output window).
6. Select one of the range fields and examine its distribution in more detail.
7. Save an updated copy of the stream.

Chapter 5

Data Manipulation

In this session we will use the stream created in the previous exercises and perform some manipulation on the fields in the data.

1. Open the stream saved in the previous exercise.
2. Create a histogram of the field called Total Spend.
3. We are going to automatically generate a Derive node that creates a new field containing four bands of *Total Spend*. Use the mouse to create three lines on the histogram where you would like to split the data. Generate the Derive node, using the Generate menu.
4. Connect the new Derive node to the Var. File source node. Edit the Derive node and change the name of the new field to *Banded Total Spend*. Attach a Table node to the Derive node and execute this section of the stream. View the new field in the data table.
5. Use a Reclassify node to create a field named Title_Gender, which is coded Male or Female based on the values in the Title field.
6. Create a Select node that selects those records for female age 50 or over. (Hint: use the Expression Builder.) Test the node. After verifying that it works, delete the Select node from the stream.
7. Save the stream under its existing name.

Chapter 6

Looking for Relationships in Data

In this session we will use the stream created in the previous exercises and investigate whether there are any simple relationships in the data. In future chapters we will attempt to predict the field *Response to campaign*, and so we will focus on relationships between this field and others in the data.

1. Using the stream from the previous exercises, connect a Web node from the Graphs palette to the Var. File node.
2. Edit the Web node to produce a web plot showing the relationships between the following fields:
Response to campaign
Pre-campaign visit
Pre-campaign spend category
Gender
Age category
3. Due to the substantial number of records in the data, set *Show only links above* to 200, set *Weak links below* to 300, and set *Strong links above* to 400. Execute the node.
4. Edit the web plot by hiding irrelevant connections. What are the three strongest connections with the responder value of the response to campaign field? Which age groups are most associated with the non-responders?
5. Investigate a relationship between the range fields of *Pre-campaign expenditure* and *Pre-campaign visits* using the Plot node. Does there appear to be a relationship in this data between these two fields?
6. Using a histogram with an overlay, investigate whether there is a relationship between the *Pre-campaign expenditure* and the *Response to campaign* field. Try normalizing the plot to make the relationship clearer. Does there seem to be a relationship between the two fields? If so, is this consistent with your conclusions from the web plot?
7. Save a copy of the stream under the name *Visual.str*.

Chapter 8

Neural Networks

In this session we will attempt to predict the field “response to campaign” using a neural network.

1. Begin with a clear Stream canvas. Place a source node (either Var. File or SPSS File) on the Stream canvas and connect it to the file used in the previous exercises *charity.csv* or *charity.sav*, whichever is the relevant format for the source node. In the case of the Variable File node, read the field names from the file. In the case of the SPSS data file, use variable and value labels.
2. Attach a Type and Table node in a stream to the source node. Execute the stream and allow Clementine to automatically define the types of the fields.
3. Edit the Type node. Set all of the fields to direction NONE.
4. We will attempt to predict response to campaign using the fields listed below. Set the direction of all five of these fields to IN and the “*response to campaign*” field to OUT. Close the Type node dialog box.
 - Pre-campaign expenditure*
 - Pre-campaign visits*
 - Gender*
 - Age*
 - Mosaic Bands* (which should be changed to type Set)
5. Attach a Neural Net node to the Type node. Execute the Neural Net node with the default settings.
6. Once the model has finished training, browse the generated Net node within the Generated Models palette in the Manager. What is the predicted accuracy of the neural network? What were the most important fields within the network?
7. Place the generated Net node on the Stream canvas and connect the Type node to it. Connect the generated Net node to a Matrix node and create a data matrix of actual response against predicted response. Which group is the model predicting well?
8. Use some of the methods introduced in the chapter, such as web plots and histograms (or use the Data Audit node with an overlay field), to try to understand the reasoning behind the network’s predictions.
9. Save a copy of the stream as *Network.str*.

Chapter 9

Rule Induction

In this session we will attempt to predict the field “response to campaign” using C5.0 rule induction.

1. If it is not already loaded, open the stream created in the previous chapter, *Network.str*.
2. Connect a C5.0 node to the Type node and execute the stream, using the default settings.
3. Once the C5.0 rule induction model has been generated, browse the C5.0 Rule node in the Generated Models palette. Fully unfold the rules to understand the decision process. Is the tree behaving in a similar way to the previously created neural network?
4. Connect a Matrix node to the generated C5.0 Rule node and create a data matrix of actual response against predicted response. Does this model appear to be predicting more accurately than the neural network?
5. Save an updated copy of your stream.

Chapter 10

Comparing and Combining Models

In this session we will compare the two models built in the previous exercises.

1. If it is not already opened, open the stream updated in the previous chapter, *Network.str*.
2. Arrange the stream so the generated Net node and the C5.0 Rule node are in the same stream, connected to the same Type node.
3. Attach an Analysis node to the end of this stream and execute the stream. Which of the models is predicting with greater accuracy? How consistent are the two models in their decisions?
4. Compare the two models using an Evaluation plot. Change the target category used to define a hit and create a new plot.

For those with extra time

1. Browse the C5.0 Rule node and automatically generate a Filter node. Attach this Filter node to the Type node and edit it. Is the rule induction method using all of the input fields?

Chapter 11

Kohonen Networks

In this session we will attempt to segment a data set containing information on SPSS course attendees. The data file, UKTraining.txt, contains information regarding the type of courses over 2000 individuals have attended.

1. Begin with a clean Stream canvas.
2. Place a Var. File node on the Stream canvas and set it to read the tab-separated text file named *UKTraining.txt*. This file contains field names in the first row
3. Click the Types tab of the Var. File node.
4. Set the fields from Number of courses to ANOVA models to Discrete type by selecting them, then right-clicking and clicking Set Type..Discrete.
5. Change the type of the following fields to the given type:

<i>Total Spend</i>	Range
<i>Number of Courses</i>	Range
<i>id</i>	Typeless

6. Set the Direction to NONE for the following fields:
 - Id*
 - Total Spend*
 - Sector*
 - Venue*
 - Number of Courses*
 - Mini Subscription*
 - Privilege Card*
 - Subscription*
7. Attach a Table node and execute the stream. Check that the types have been correctly defined.
8. Connect a Kohonen node to the Var. File node and edit the Kohonen node. In the Expert tab change the Width and Length options to 3. Execute this section of the stream.
9. Once the Kohonen network has finished training, browse the generated Kohonen node, examine the cluster viewer (Viewer tab), and try to describe the main clusters.
10. Generate a Select node to select the main clusters
11. Place the generated Kohonen node in the Stream canvas and connect the Type node to it.
12. Connect the generated Kohonen node to the generated Select node
13. Create a field (use a Derive node) that uniquely identifies each Kohonen group and add it to the stream

14. How do the Kohonen clusters relate to the following fields?
Region
Mini Subscription or 5 day deal
Subscription
Privilege Card
15. Are there any further patterns to be found?
16. Save a copy of the stream using the name *Kohonen.str*.

Chapter 12

Association Rules

In this session we will attempt to run a market basket type analysis using the data from the previous exercise, to establish which courses lead to other courses or are purchased together.

1. Open the stream saved in the previous chapter, *Kohonen.str*.
2. Edit the Var. File node and select the Types tab.
3. The majority of the settings can be kept as they were in the previous chapter, apart from the directions of the course fields.
4. Change all of the course title fields to direction BOTH, so that they appear in rules as either an antecedent or consequent, apart from the following:
Introduction to SPSS
Introduction to SPSS and Statistics
FastTrack

which must all be set to IN. (We are interested in which courses lead to others, therefore, assuming the first course attended is an introductory course, these fields will act as conditions only.)

5. Connect an Apriori node to the Type node and edit the Apriori node.
6. Set the Minimum rule support to 1% and the Minimum rule confidence to 50%. Click the *Only true values for flags* check box (so it is checked).
7. Execute this section of the stream
8. Browse the resulting unrefined model in the Generated Models palette and sort the rules by support x confidence.
9. Do the rules make sense?

Chapter 13

Sequence Detection

In this session we will explore sequences associated with a specific event: failure to resolve a telecom service problem. We will use a subset of records from the data set used in Chapter 13: sequences with termination code 299 (meaning that the problem was not resolved).

1. Read the tab-delimited text file named *FailTelRepair.txt* into Clementine (you can use a Var. File node or modify the *Telrepairdef.str* stream file).
2. Click the Types tab, then click the Read Values button, and then click OK
3. Add a Table node to the stream and execute it.
4. Add a Sequence node to the stream. Specify ID as the ID field, Index1 as the time field, and Stage as the content field. Run the sequence detection analysis in Simple mode
5. Browse the generated rule set. What are the most confident and the most common sequences? Try different sorts of the generated rules to see if they provide additional insight.
6. Compare these results to those reported in Chapter 13. Do you find any stage codes related that frequently appear in sequences involving code 299 (failure to resolve problem) that do not frequently appear in sequences involving code 210 (problem solved)? These might provide some insight into which stages lead to a repair failure.