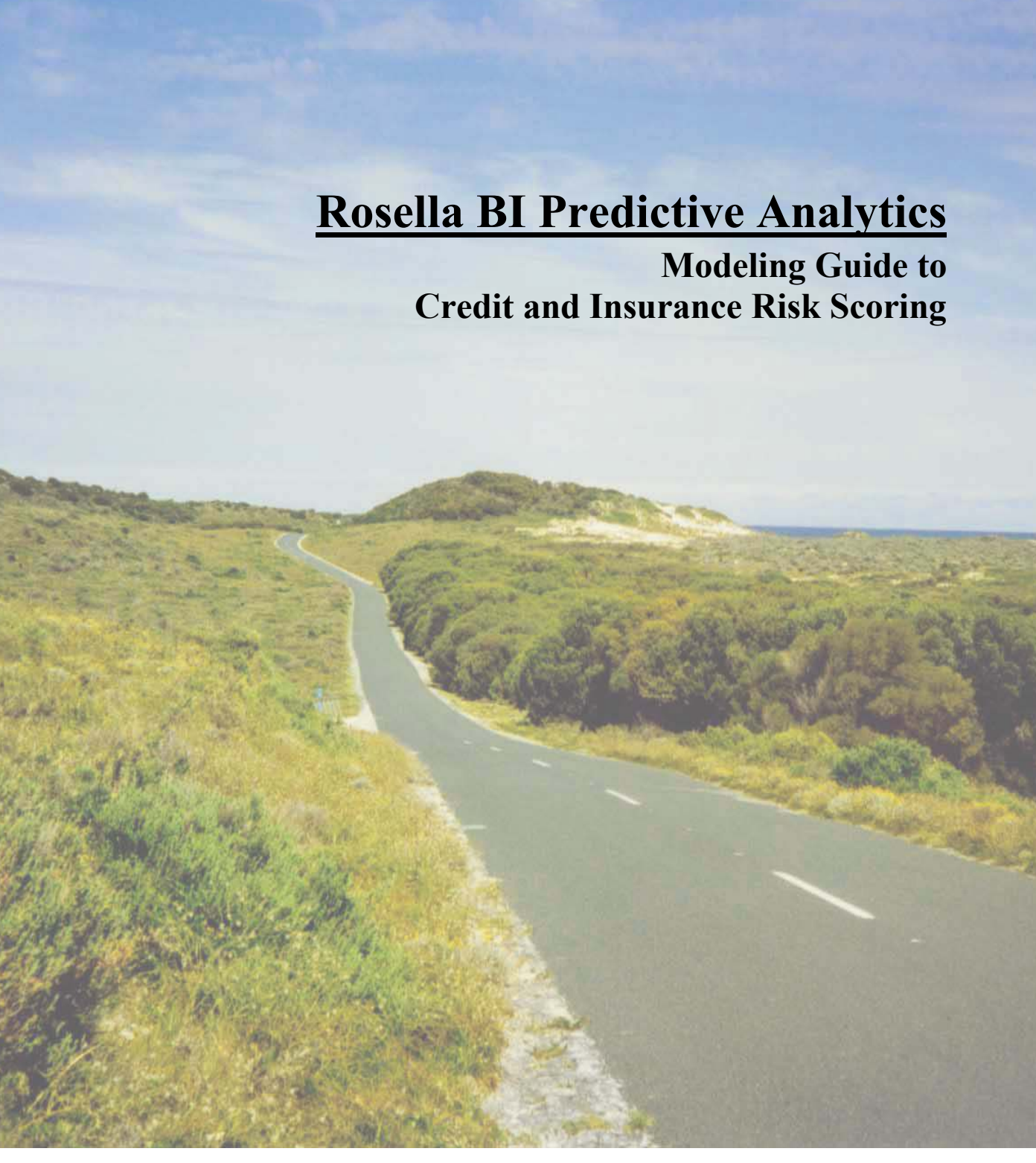
Rosella BI Predictive Analytics

Modeling Guide to
Credit and Insurance Risk Scoring



Rosella Software

www.roselladb.com

Sydney Australia

© Copyright 2007-. All contents and designs of software interfaces used are copyright of Rosella Software.
Unauthorized distribution or copy is not permitted.

Preface

This manual is designed for model developers, specially targeted on Credit and Insurance Scoring Models on Rosella BI predictive inferential platform.

May 17, 2007.
June 14, 2007 - Revised.
October 29, 2008 – Revised for CMSR.
January 21, 2010 - Revised.
October 24, 2016 – Revised for MyDataSay.
February 24, 2017 – Revised for Deep Learning

Table of Contents

1. RISK SCORING SYSTEMS - INTRODUCTION 1

2. DATA PREPARATION 2

2.1 CUSTOMER INFORMATION USED IN SCORING 3
2.2 PAST OUTCOME VARIABLES 4
2.3 WHERE TO STORE YOUR SAMPLE DATA? 4
2.4 CLEANING YOUR SAMPLE DATA 5
2.5 WHAT ARE TRAINING DATASETS AND TEST DATASETS? 6
2.6 CONNECTING YOUR DATABASE TO CMSR 7
2.7 IMPORTING DATA INTO CMSR 9

3. ANALYZING VARIABLE RELEVANCY 9

3.1 DISTRIBUTION ANALYSIS 10
3.2 CORRELATION ANALYSIS 11

4. HOTSPOT AND EXCEPTION ANALYSIS 13

5. SEGMENTATION AND PROBABILITY SCORING 15

5.1 DEVELOPING DECISION TREE MODELS 17
5.2 VALIDATION OF MODELS 19
5.3 APPLYING TO CUSTOMER DATABASE 20
5.4 ANALYZING SCORE DISTRIBUTION 21

6. NEURAL NETWORK MODELING 22

Neurons in Biological Neural Network 22
What is Artificial Neural Network? 23
How neural network predicts? 23
How neural network is trained? 24
6.1 DEVELOPING NEURAL NETWORK MODELS 25
6.2 APPLYING TO CUSTOMER DATABASE 28
6.3 ANALYZING SCORE DISTRIBUTION 29
6.4 MODELING RISK AMOUNTS 30

7. REGRESSION MODELING 32

8. PUTTING THEM TOGETHER 34

9. DEPLOYMENT OF MODELS 35

9.1 MYDATASAY ANDROID APP 35
9.2 WEB-BASED MODEL DEPLOYMENT 36

APPENDIX I. CUSTOMER CHURN ANALYSIS 37

INDEX 38

1. Risk Scoring Systems - Introduction

Risk management is an essential part of credit and insurance industries. Risk scoring is a numerical rating of risk level involved in credit loans and insurance policies. Risk sores, such as credit scores and insurance scores, provide objective rating to professionals. Risk scoring models are aimed to help professionals to arrive at a statistical understanding of the probability of risk. That is to say, risk scoring models are aimed to identify those applications that are highly likely to going bad.

With the maturity of advanced predictive modeling technology, risk scoring methods have improved significantly. In this manual, we describe how sophisticated scoring models can be developed out of past data. This manual covers the following topics;

- How to identify relevant modeling variables?
- How to identify risk hotspots and exceptional customer segments?
- How to develop statistical probability scoring models using decision tree?
- How to develop neural network scoring models?
- How to develop regression models?
- How to combine multiple models and scoring systems?
- How to deploy models for customer-facing staffs.

Detailed procedures are described in the subsequent chapters.

2. Data Preparation

The first step developing risk scoring models is data preparation. Generally, data fields fall into two major categories;

- **Numerical data**: Numerical data, also known as linear data, represent numerically measurable values such as age, height, weight, quantities, amount, numbers of years, etc. Example values include 45, 35000, 0.123, 0.6542E06, etc.
- **Categorical data**: A categorical variable, also known as nominal variable, has a finite number of values. For example, gender has "Male" and "Female". Other examples are vocation, job position, religion, etc. They normally have a small number of pre-determined values. Boolean variables may be handled as a categorical variable: "YES" or "NO" categories.

In addition, the following temporal data may be included in input data, after transformation into either numerical or categorical values;

- **Date and time**: Date and time is a special data type that may not be used directly in analytics. This requires transformation into either numerical or categorical values. For example, date of birth can be transformed into a numerical field "age". Calendar months can be translated into a categorical field "seasons": Spring, Summer, Fall, and Winter.

Although identifiers are not used in analytic tools, identifiers are need in updating database records. When predictive or segmentations models are applied to database records, CMSR requires a unique identifier field to locate customer records.

- **Identifiers**: Examples of identifiers includes employee number, social security number, customer registration numbers, etc.

Finally, textual information has no values in analytics. They should not be included in prepared data.

- **Textual descriptive information**: Textual descriptions, such as book abstract, addresses, product descriptions, etc., cannot be used in data mining directly. If they contain any useful information, they should be extracted and transformed into either numerical or categorical variables.

2.1 Customer information used in scoring

Customer databases may contain a variety of information. Among them, the following information may be useful for developing risk models. As described in the previous page, only numerical and categorical fields may be included in prepared data.

- **Demographic variables** describe characteristics of customers and include age, gender, race, education, occupation, income, religion, marital status, family size, children, home ownership, socioeconomic status, and so on. Note that *demographic segmentation* normally refers to segmentation with these demographic variables.
- **Geographic variables** include information on geographic areas. For example, zip code, state, country, region, climate, population, and other geographical census data. Note that this information can come from national census data.
- **Psychographic variables** describe life style, personality, values, attitudes, and so on. Note that *psychographic segmentation* normally refers to segmentation with these psychographic variables.
- **Behavioral variables** include product usage rate, brand royalty, benefit sought, decision making units, ready-to-buy stage, and so on.
- **Financial variables** provide important financial information: assets, loans and debts, credits, investment, and so on. Note that this may apply both individuals and companies. Information appearing in credit bureau credit reports (for practical reasons, at the time of application) can be important variables.
- **Employment information** provides important personal financial information. This may include type of employment, type of jobs, length of employment, salary, etc.
- **Health information** is also important for healthcare and life insurance applications.
- **Insured objects** also have own attributes to consider in modeling. For example, in motor insurance, properties of motor vehicles play a part significantly.

It is important to note that federal and state laws and regulations may prohibit using certain variables in credit and insurance scoring. For example, in USA, the Equal Credit Opportunity Act prohibits creditors from discriminating customers for the reasons of race, color, religion, national origin, gender, marital status, age, etc. Please check your local laws and regulations.

2.2 Past outcome variables

In addition to the variables described in the previous section, variables representing past outcome are essential in developing risk scoring systems. Generally, past outcome is indicated by one of the followings;

- Insurance claim amounts or numbers of claims made (for insurance scoring).
- Credit default amounts or numbers of loans defaulted (for credit scoring).

Based on this information, we can derive the following variables. Note that you may choose variable names of your own. For convenience, however, we will use the following names consistently, throughout this the manual.

- **"RISKVOLUME" :** This is the values that have lost as the result of going bad. For insurance, this may be total claims amount. For credit and finances, this is default amounts.
- **"RISKFLAG" :** If "RISKVOLUME" is grater than zero, value is **1.0**. Otherwise value is **0.0**. . (Note that this value encoding is used in neural network modeling and variable relevancy analysis.)
- **"RISKCLASS" :** If "RISKFLAG" is 1.0, this may be coded as "**Risky**". Otherwise, "**Safe**". (Note that this coding is used in decision tree as well as in dimensional analysis.)

2.3 Where to store your sample data?

Before collecting analytic data, you need to determine data storage. It is best to collect data in JDBC enabled relational database systems.

You may need to prepare datasets into a number of sub datasets so that you can use for modeling as well as testing (See section "2.5 What are training datasets and Test datasets?"). You may create a single master table containing all the customers. Sub-datasets may be derived from the master dataset as sub-datasets are needed.

It is important to note that customer data records are required to have a unique identifier field, e.g., customer number, or policy number, or company number, etc. Record identifiers are essential in updating customer records from predictive models. The identifier field must be either primary key or indexed. If not, create an index for the identifier field. Otherwise updates can take very long!

2.4 Cleaning your sample data

Data you collected often contain information that may distort analytic results. They may contain corrupt data, null values, coding errors, etc. There are numerous reasons for corruption of data. You need to identify problems and correct them as much as possible. Most common problems include null values, synonymous names, bogus values, and outliers.

Null and missing values

Values may be missing either because of human error or because information is not available. The latter case is normal and CMSR tools automatically handle such information at processing time. However, if possible, the former problem needs to be addressed. It is important to note that you should not replace null values with artificial values as suggested in some literature! CMSR can handle null information appropriately. Replacing nulls with artificial values may prevent analytics performing appropriate processing. If proper replacements are not available, just leave nulls as nulls.

Synonymous names

Synonymous names represent the same entities but use different names, e.g., “January”, “Jan.”, “Jan”, “jan”, etc. Synonymous names split statistics. Synonymous names in customer database tables can be identified with the following SQL query statement;

```
SELECT DISTINCT Variable_Name FROM your_table
```

Bogus values

Bogus values are normally inserted with software bugs and system glitches. For categorical variables, bogus values can be identified using the query for synonymous names. For numerical variables, bogus values can appear as outliers. You may be able to check with the following SQL statement;

```
SELECT MIN(Variable_Name), MAX(Variable_Name) FROM your_table
```

Outliers

Outliers are extreme numerical values. Outliers can distort statistics and models. As models will try to adjust to extreme values, outliers will lead to distortion, resulting in less accuracy. It is desirable to handle outliers specially, say, using Rule-based Modeling. Outliers can be detected using scatter plots and histograms.

Too many categorical items

Categorical variables may contain many items. Such variables are difficult to use in predictive modeling. Using correlation analysis described section 3.2, identify relevant items. Redesign items. For example, if “Item1”, “Item2”, and “Item3” are significant, then the variable can be recoded as “Item1”, “Item2”, “Item3” and “Other”.

2.5 What are Training datasets and Test datasets?

Training and test datasets are specially-reserved concepts for predictive analytics. If you are not developing predictive models, you may not need to prepare training and test datasets.

Literally speaking, a training dataset is a data that is used in developing (or training) predictive models. Test datasets are prepared to test goodness of predictive models. You will normally prepare a dataset for developing models, and several for testing purposes. Keep in mind that you are building models using data collected in the past. The data represent past patterns. This does not necessarily imply that models developed using the past data will predict future events accurately! When selecting a training dataset, you need to choose carefully. Data used in training should be **"a good reflection of future"**.

Test datasets may be derived from past datasets in a number of different ways;

- **Different time periods**: Create different datasets for each different time period.
- **Random selections**: Select past data using different randomizing methods.
- **Artificial subsets**: Different artificial subsets of data may be used for testing.

Test datasets are subsets of the whole data. It is recommended to store each dataset into a different database table. This will make it easier to apply CMSR validations tools.

What is Overfitting?

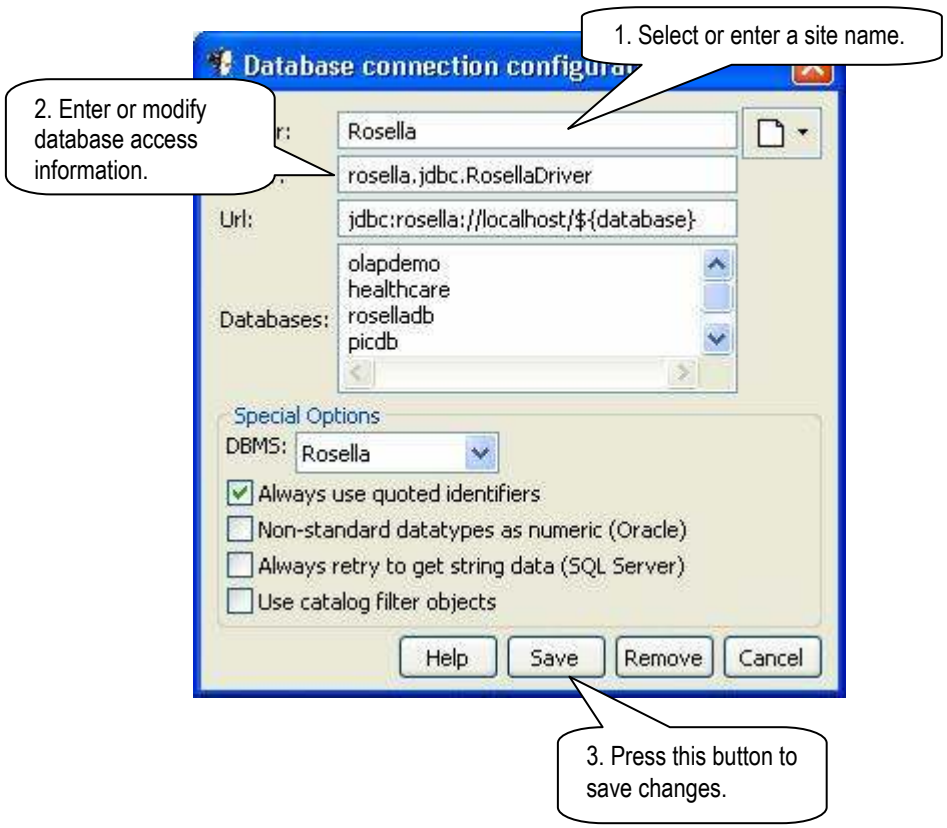
Overfitting is a phenomenon where predictive models learn training datasets too detail, i.e., "overly fit" to the training datasets. Although it is desirable that models learn as detail as possible, the negative side of over-learning is that models tend to fail in other datasets. In another words, overfitting models have less accuracy in predicting future events. Therefore, testing of models with various datasets is very important. Validating against several test datasets will provide indication how general your predictive models are. There are several ways you can avoid overfitting. Methods may differ based on the type of modeling tools;

- **Neural network**: A general approach is to reduce the number of internal nodes so that network can learn less in detail. Note that overfitting problem is more acute if training data is small. Smaller network nodes imply less learning capability. This will force network to learn more general patterns than details.
- **Decision tree**: The best approach is to prevent decision tree nodes that may have too little statistical support. Using higher support level, overfitting can be avoided.

If you are not familiar with neural network and decision tree, skip to the next section. More on overfitting is described later.

2.6 Connecting your database to CMSR

To import data stored in database systems, you need to setup database connections between CMSR and your database systems. CMSR uses connection facilities called JDBC (Java DataBase Connectivity). Each database system has own JDBC connection drivers. CMSR connects through JDBC drivers. You can install JDBC drivers and configure connections from "DBMS Configuration" of the "Data" menu. The following figure shows the database configuration window. Perform the steps described in the figure;



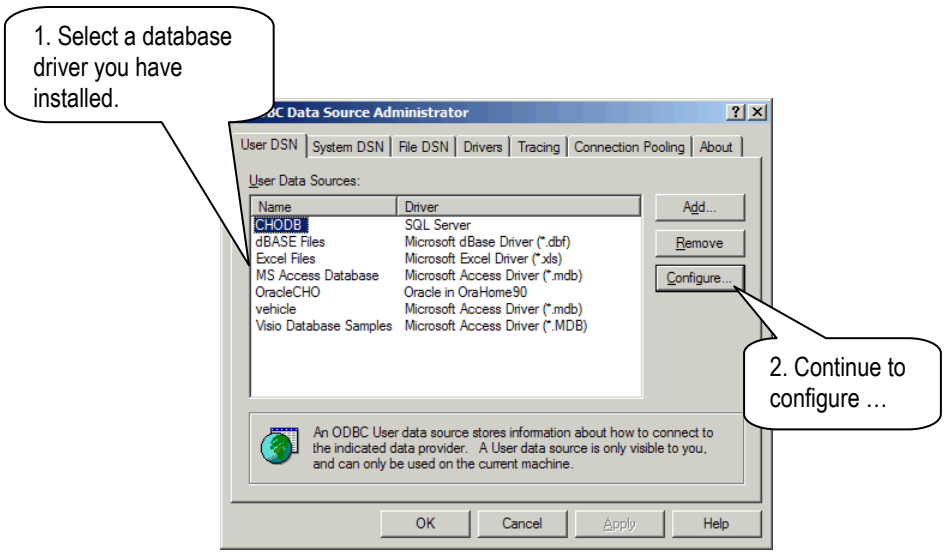
JDBC drivers are available from database system vendors. JDBC drivers normally have file names with suffix ".jar". To install JDBC drivers, select "Install JDBC Drivers" of the "File" menu of CMSR.

On Windows, you can use ODBC (Open DataBase Connectivity) drivers. Note that to use ODBC, you need Java 7 or earlier version of Java Runtime Environment (JRE). From Java 8, ODBC is not supported. For installation of Java 7, read the installation guides "Java7-HOWTO.txt" available in your "CMSR_HOME/JDBC-HOWTOs" directory.

When you install your database systems on your Windows operating system, ODBC drivers are installed automatically. For example, if you install MS Office with Access database system, Access ODBC drivers are installed automatically. If your database is an external source (or not installed inside your own system), you may need to install drivers manually.

To connect database systems through ODBC, perform the followings;

1. Install ODBC drivers (if you are accessing external databases).
2. Create an ODBC DSN (Data Source Name) for your database. To create DSNs, open Windows "Control Panel" – "Administrative Tools" – "Data Sources (ODBC)". Then you will see the following window. Select a driver and "Configure…" your database. Note that this will create a DSN for your database. In JDBC configuration described in the previous page, "DSN"s are equivalent to "database names"!



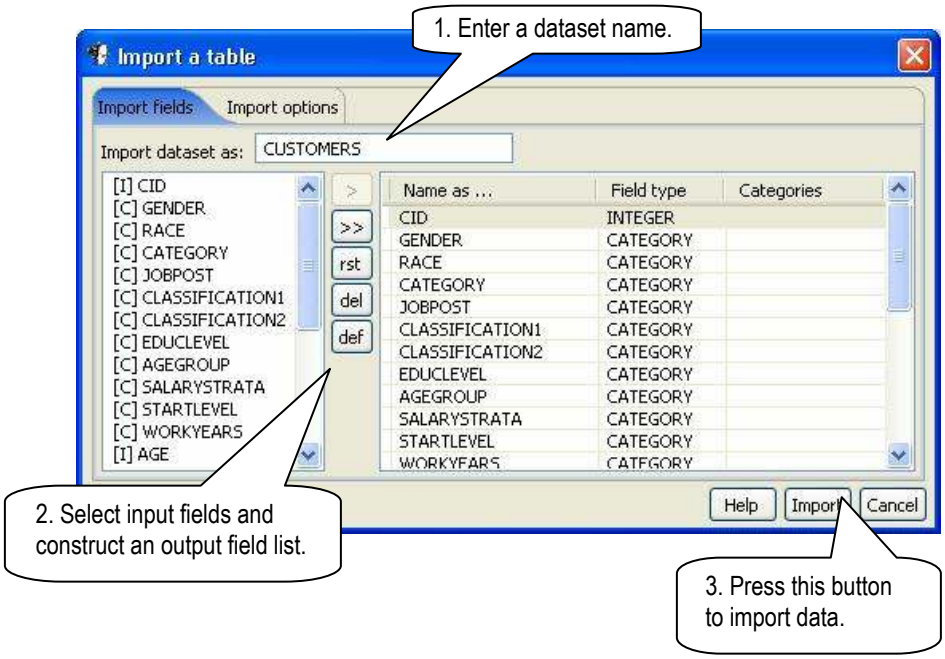
3. As described in the previous page, configure a JDBC configuration for ODBC. Add the DSNs you created to the "Databases:" list. CMSR has several pre-defined configurations that you can modify. Use one of "ODBC" site configurations.

It is noted that you can configure ODBC and JDBC configurations in any sequence!

2.7 Importing data into CMSR

To use modeling tools of CMSR, you need to import data from your customer database. It is noted that you can import data from flat files. However, we will assume that your data is stored in database systems. This will allow you to perform advanced operations (such as model applications).

To import a database table, select a database from “Databases”. Then select “Import from tables” of the “Data” menu. A selection dialog will pop up. Select a schema and a table. Then the following dialog will appear. Perform the steps described in the following figure;

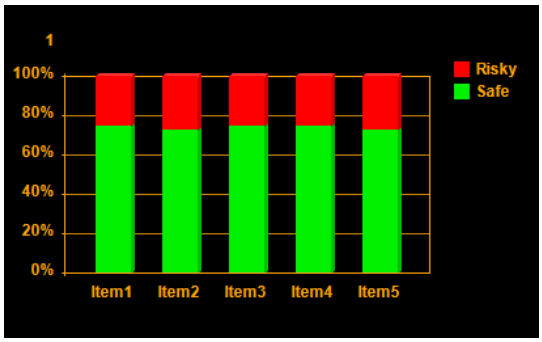


3. Analyzing Variable Relevancy

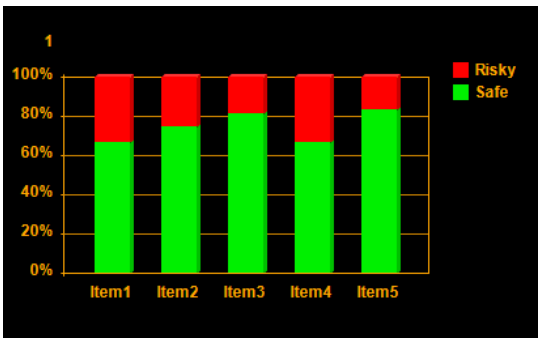
How do you know that data you collected are any indicator for predicting risk scores? In another words, how can you determine that a data variable has any relevancy to credit defaults or to insurance claims? The following sections will show two methods.

3.1 Distribution Analysis

The first method is to analyze distribution of outcome values (=”RISKCLASS”) using bar charts (for categorical variables) and histogram charts (for numerical variables). The following is a bar chart showing categorical proportions of risk classes of items of a categorical variable. It was produced using a categorical field as “Item” variable and “RISKCLASS” as “Category” and “1” as “Value(s)” from CMSR bar chart. From bar chart, select “Categoric proportion” for “Percent” and “Stacked/Cumulative” for “Stacking.



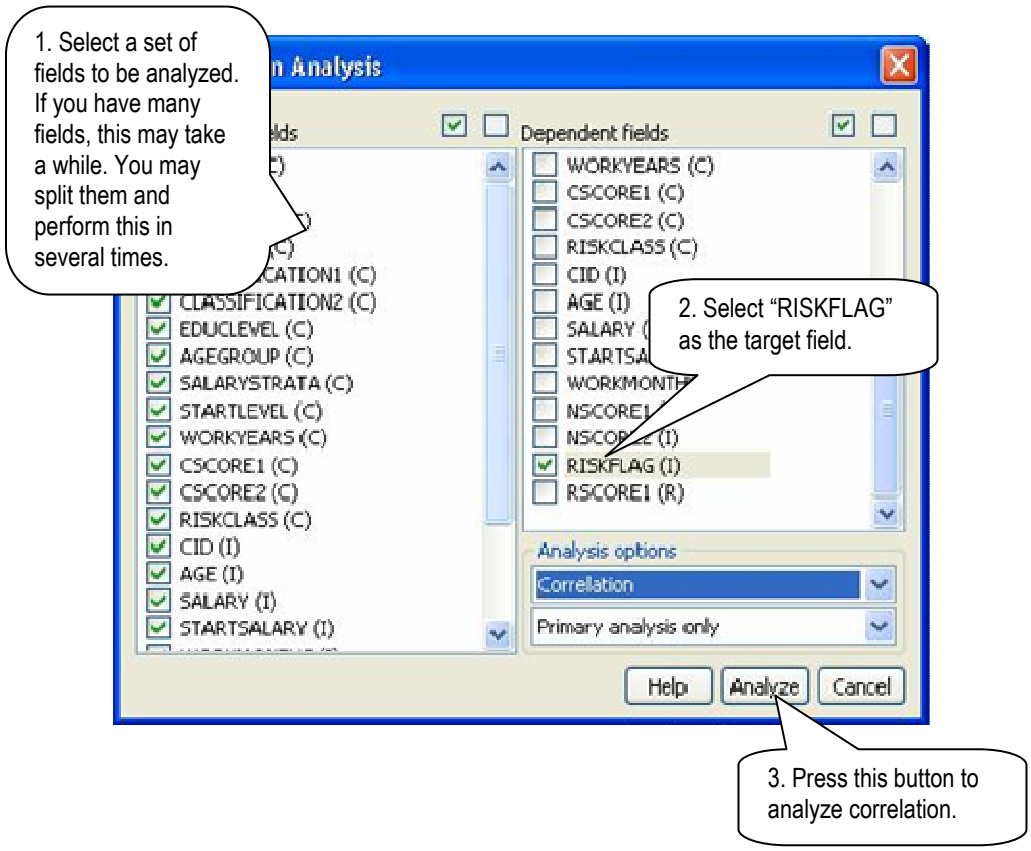
There are very little differences in risky/safe proportions amongst items. This type of variables is not useful for predictive modeling as they do not differentiate risk levels. The following is a same chart. But this variable shows differences in proportions of risk classes. This type of variables can be used in predictive modeling.



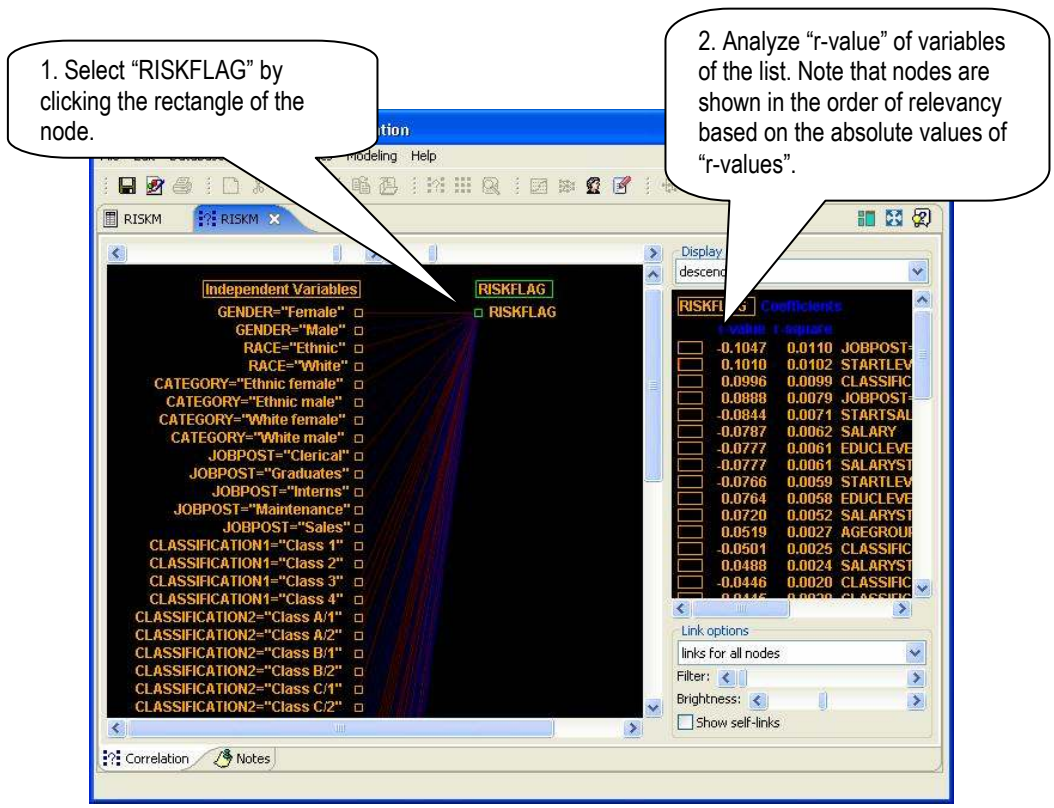
For numerical variables, similar analysis can be performed using histograms of CMSR. If ranges show differences in risk class distribution as seen in the above figure, they can be used in predictive modeling. Use the “RISKCLASS” variable as “Category” in histograms. Note that bar charts and histograms showing categorical proportions will be used often in this manual.

3.2 Correlation Analysis

The next method is correlation analysis. Correlation is measured between -1 and 1. It indicates the degree of association between two variables. If coefficient is 1, two have perfectly positive correlation. If it's -1, two have perfectly negative correlation. If it is 0, two have no association at all. CMSR link analysis is a visualization tool for many-to-many variable correlation analysis. From CMSR main window, select "Correlation Analysis" or the "Analytics" menu. Then follow the procedures in the following figure;



Once computation is completed, the following diagram panel will appear. Analyze as described in the figure;



It is noted that the “r-values” indicates correlation coefficients. It is safe to assume that if the absolute values of “r-value” is greater than 0.1, two variables may have at least certain association. If the value is greater than 0.5, it has significant correlation. It is also safe to assume that if the value is smaller than 0.01, the variable may not have any association to credit defaults or insurance claims, and therefore it may be excluded in scoring systems.

“R-values” close to 1.0 or -1.0 may indicate exceptional cases that you can safely infer prediction based on the variable alone! In practice, however, this may not exist amongst risk scoring variables you collected.

In the figure, r-values are very small. This is quite common in skewed datasets. If the interested targets have very high skew in values and there are no clear-cut influential variables, they tend to show very low r-values.

4. Hotspot and Exception Analysis

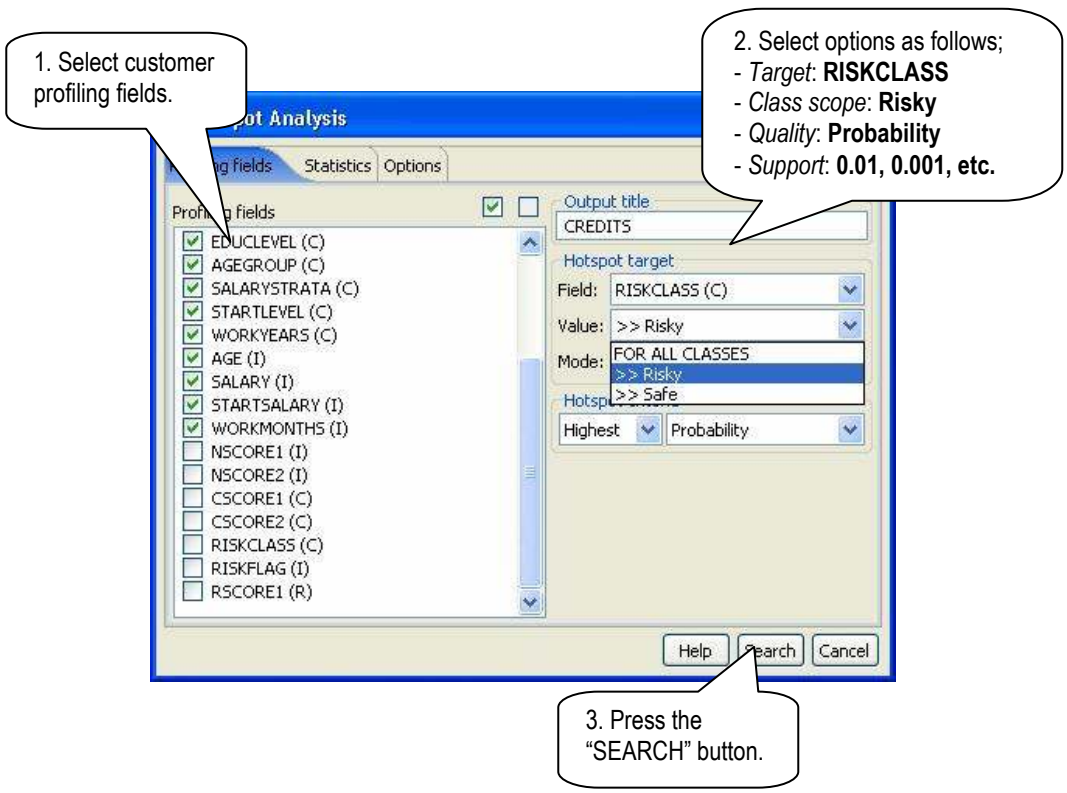
Hotspot analysis is a powerful tool that can be used to identify customer segments prone to most or least credit defaults or insurance claims. It's a very effective customer profiling tool. Hotspot analysis can identify exceptional customer segments as well. For risk management, customer segments having highest credit defaults or insurance claims ratio will be of highest interest.

The Pareto Principle

The Pareto principle states that in many events, 80~90% of effects comes from 10~20% causes. This can be applied to risk management. Most of insurance claims and credit defaults may come from small portions of customer segments. Identifying the segments can reveal useful information for developing risk scoring models. Hotspot profiling tools can be used for the purpose.

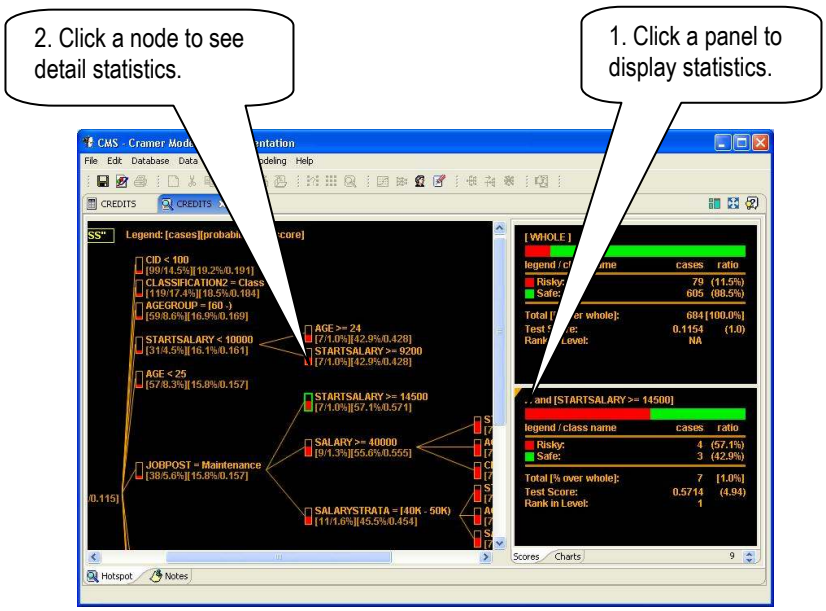
Configuring Hotspots

To identify hotspot customer segments, perform the steps described in the following figure;



Analyzing Results

Result of hotspot analysis will be displayed in the following window. The left mini window panel shows customer segments having highest/lowest probability of being claimed or defaulted. It is noted that the diagram is a overlapping drill-down tree. At the top left is the root which represents the whole customer population. The second layer is customer segments with highest /lowest claim/default history. Nodes may represent overlapping customer sub-segments, meaning that the same customers may be in multiple segments of the same layer. The next layer is sub-segments of corresponding segments having highest /lowest claim/default history within the corresponding parent segments. The right mini windows show detail statistics of nodes selected in the tree. To see statistics of a node, click the front part of the node.



At first attempts, results of hotspot analysis can be messy. Or it can show exceptional cases or outliers. If result is messy, try again but exclude the variables that make result messy. In addition, “Support” level can be useful in filtering messy segments. In addition, be careful not to use too high support level. They may filter out some interesting segments.

5. Segmentation and Probability Scoring

The principle of this scoring technique is predictive segmentation and statistical probabilistic reasoning. Customers can be divided into segments in such a way that can boost either the proportion of "Risky" customers or the proportion of "Safe" customers. Customer risk scores are measured in terms of ratio of "Risky" customers in corresponding customer segments. For example, if a customer belongs to a segment with 23% past insurance claims history, the score of the customer can be assigned to, say, 0.23. This ratio will indicate relative risk level amongst customers.

Decision tree is an excellent tool for segmenting customers for scoring purposes. Decision tree divides customers into smaller sub-segments recursively. At each segment, splitting is made in a way that boosts proportions of either risky customers or safe customers in each resulting sub segment. If further split does not improve or leads to segments with less than the support-level, segmentation stops.

How decision tree splits segments?

At each segment, decision tree selects a variable that can boost the proportion of either "Risky" or "Safe" customers. This is done based on one of the following mathematical criteria;

- Cramer coefficients.
- Entropy and entropy gain ratio.
- GINI diversity index.
- Chi-square statistic and probability.
- Expected accuracy or expected probability.
- Twoing.
- Manual selection (for arbitrary segmentation analysis).

Decision tree computes merits based on the chosen criteria. For example, if you choose Cramer, decision tree computes Cramer coefficients for all variables included in modeling. Then, the variable with the highest Cramer coefficient will be chosen as the splitting variable. If the variable is categorical, the segment will be further divided for each categorical value. If it is numerical, normally it will be divided into two sub-segments, based on a boundary in the middle. Different decision tree tools primarily support different criteria.

Generally speaking, the best criterion is Cramer. It tends to produce most compact trees without sacrificing general accuracy. It is noted that other splitting criteria tend to prefer segmentation that produce a larger number of sub-segments. Cramer avoids this bias using degree-of-freedom naturally. Both in theory and in practice, Cramer decision tree is the best one to use.

What's wrong with classification?

Classification is a predictive modeling approach where categorical class values are assigned to records (or customers), e.g., "Risky", "Safe", etc. Decision tree was originally developed as a classification method. In decision tree, statistical distribution

of categorical values is used in determining class values. For examples, if a decision tree (terminal) node contains less than 50% "Risky" customers, all the customers falling into the node are classified as "Safe". This will be true even if 49% of the customers of the segment have insurance claim or loan default history. In other words, classification will classify them as "Safe", despite they are in a high-risk segment. It should be noted that 49% insurance claims or loan defaults means extreme risk!

There are suggested tricks that alleviate this problem. For example, boosting artificially increases the number of risky customer records in training datasets. Assume that a training dataset contains the following records;

Risky	Safe
10%	90%

Duplicating "Risky" records, boosting increases the ratio of "Risky" records, say, to 50% as follows;

Risky	Safe
50%	50%

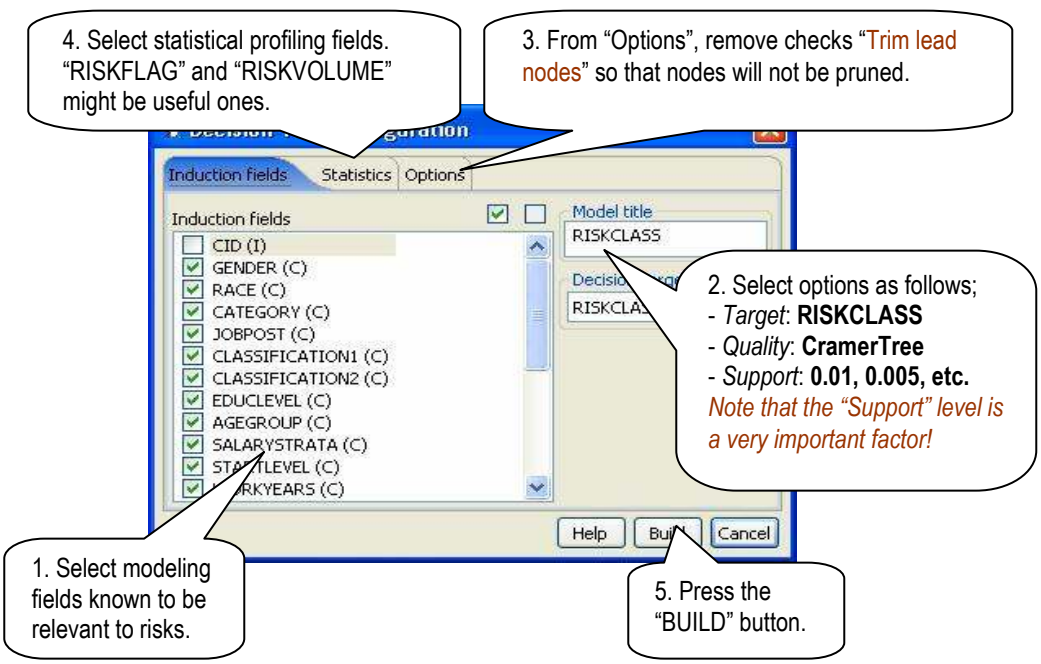
Without boosting, classification may predict "Risky" as "Risky" at about 5% ~ 15% accuracy. Boosting may improve to, say, 30%. This is not a reliable accurate system you can use to discriminate customers! In addition, boosting distorts the true representation of risk distribution, and thus it loses legitimacy as a fair approach. Therefore, we avoid classification altogether. We will develop scoring models based on predictive segmentation and statistical probability.

5.1 Developing Decision Tree Models

Enough has been explained about decision tree. In a nutshell, risk scoring modeling can be developed as follows;

1. Develop decision tree taking "RISKCLASS" as the target classification variable.
2. Each segment will have ratio of "Risky" customers. Use this ratio as the relative score of customers who belong to the corresponding segments.

To develop a decision tree, configure model as described in the following figure;



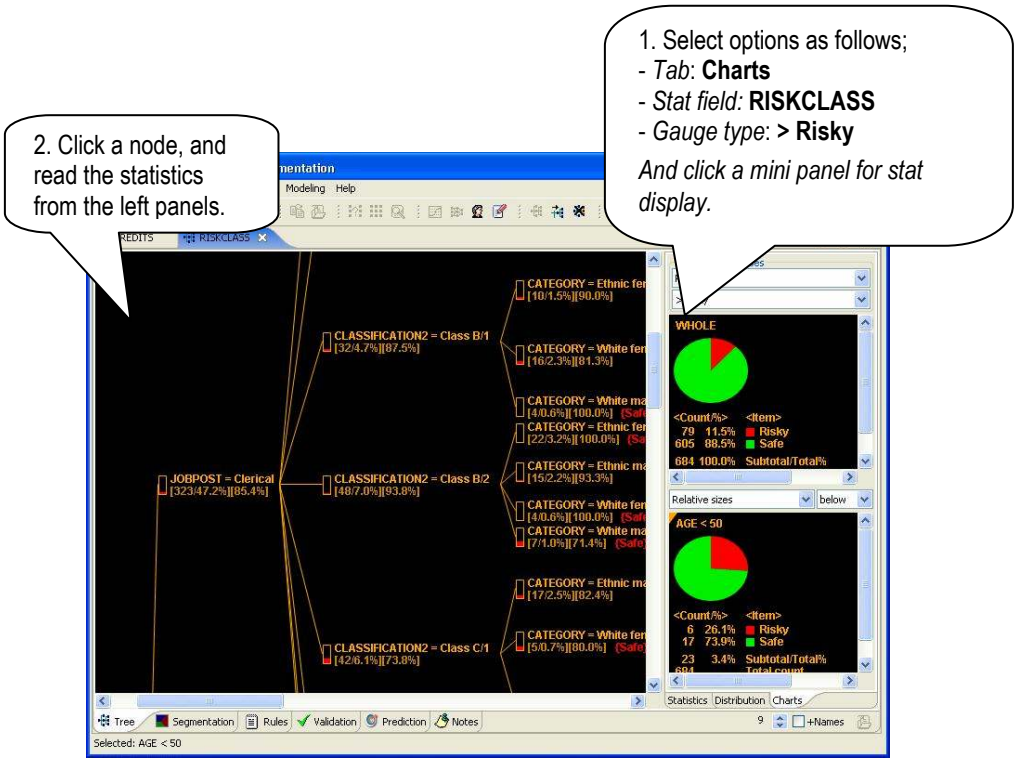
Note that in "Step 4" supplemental fields provide statistical segment analysis. The target field "RISKCLASS" is automatically selected. You may add additional selection to the list. In addition, if you want to prune segments manually after tree construction, select "Auto-manual" from the "Growing" option and specify enough "Depth:" of resulting trees.

How to avoid overfitting in decision tree?

Overfitting is a phenomenon where predictive models learn training datasets too detail, i.e., "overly fit" to the training data. The consequence is that models may not predict accurately on other datasets, which means that they may not accurately predict on future events. Once a tree is developed, you need to verify with a number of datasets to test whether it consistently performs well. If testing result is patchy, you may try with different "Support:" level. Note that "Support:" controls the level of support in each tree nodes. The higher, trees will become more general, and thus may work better on test dataset. If too high, on the other hand, meaningful trees may not be produced.

Analyzing Decision Tree

Result of decision tree is displayed in the following window. To analyze segments, follow the steps described in the figure;



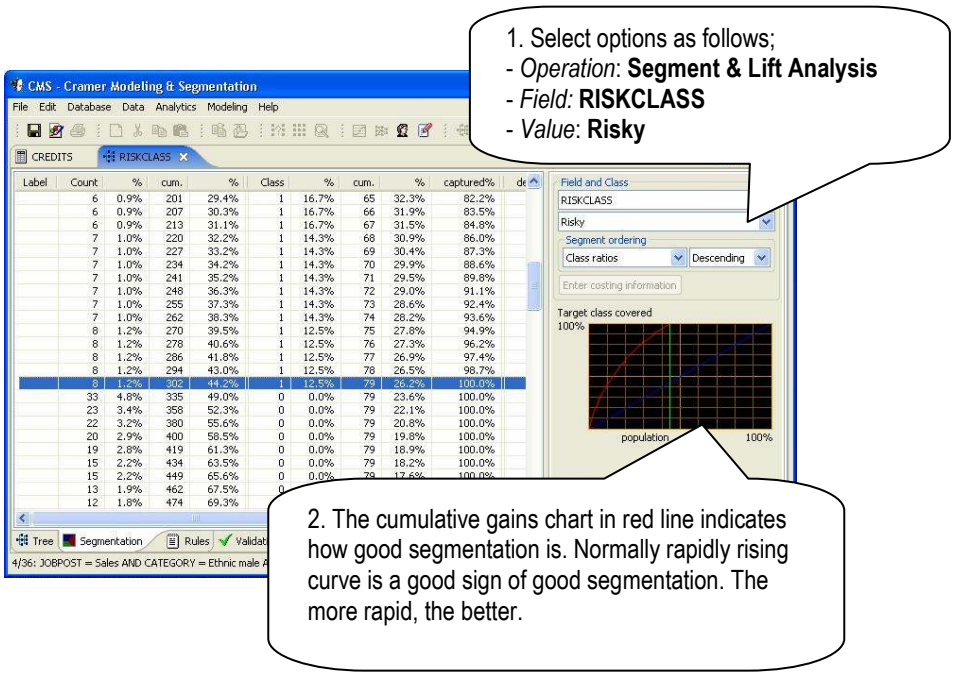
The red part of node rectangles represents risky customer proportions. You will notice that terminal nodes are biased towards either risky customers or non-risky customers. Risk level of a customer is determined by the ratio of risky customers of the node that the customer belongs to.

To classify a customer into a segment, follow the decision tree from the root based on the attributes of the customer. Normally, a terminal node is reached. The ratio of risky customers of the terminal node will be taken as the score of the customer. When there is a NULL value or a value other than in the node, the node will be treated as the terminal node and the ratio of that node will be used as the score.

5.2 Validation of Models

Validation of models is very important to assure that models can predict accurately on future events. As we are based on statistical probability modeling, validation methods used in classification modeling techniques do not work. We still need to find a better way to deal with than the ones described here. For the time being, we will do limited validation procedures.

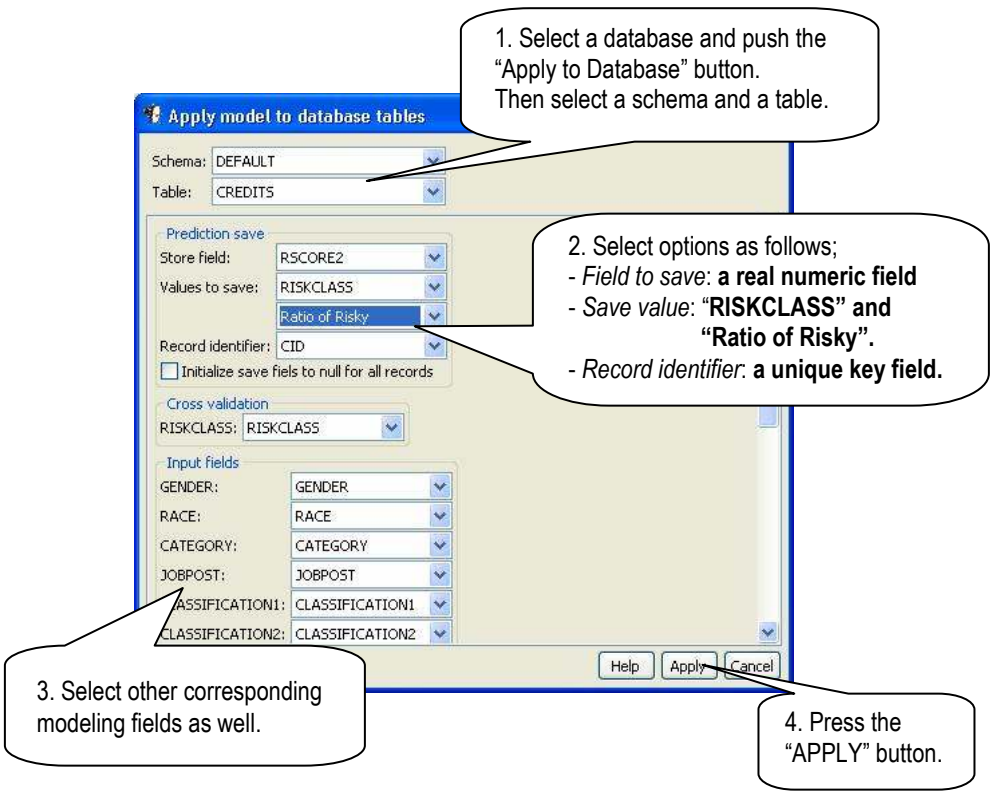
First, we can determine how good decision tree segmentation is using the cumulative gains chart as shown in the following figure;



In the cumulative gains chart, the red curve indicates how good segmentation is. The blue diagonal line indicates gains from random models. Population is represented horizontally, from 0% (at the left end) to 100% (at the right end). Percentages of captured risky customers are shown vertically. A rapidly rising curve is a good indication that risky customers are concentrated in some segments (appearing earlier in the table), provided that it is not a consequence of overfitting. This in turn implies a good predictive segmentation with high accuracy. You may develop a number of decision trees using different sets of modeling fields, and/or using different splitting criteria, say, entropy, twoing, etc., and compare cumulative gains charts. And choose the ones with the most rapidly rising models for deployment.

5.3 Applying to Customer database

To apply the decision tree to a database table, perform the steps described in the following figure. You will need a numeric field that can store REAL values with decimal points, e.g., 0.27347, etc. It is recommended to create a number of scratch fields on customer tables. They can be used in variety of validation works.



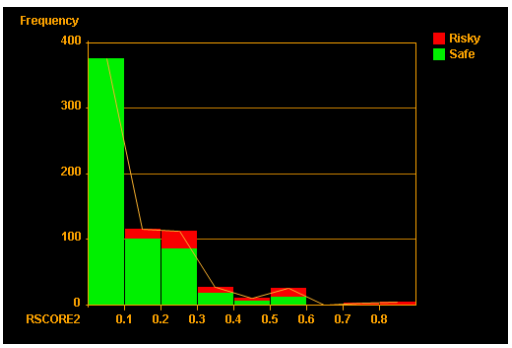
This reads database customer records one-by-one, and evaluates using the decision tree. Result values will be stored on the "Field to save".

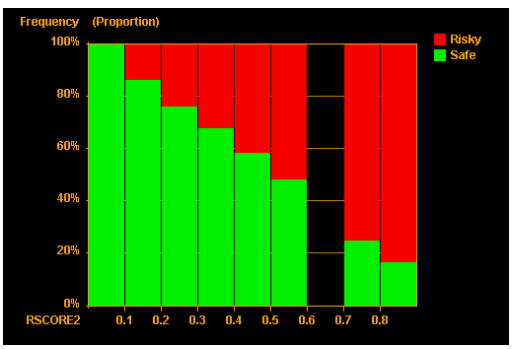
5.4 Analyzing Score Distribution

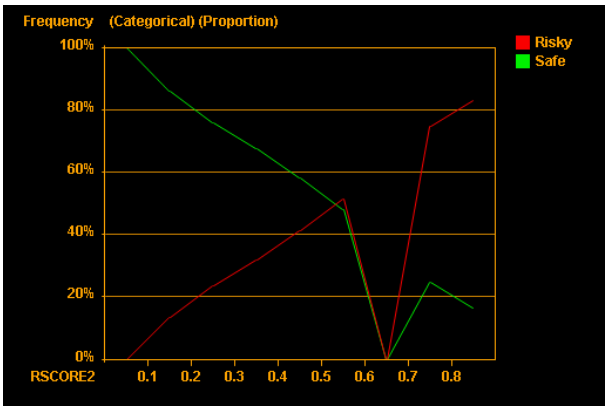
Once you apply decision tree to a database table, you can analyze probability-score distribution using "dimensional histograms". Dimensional histograms are available in CMSR. Select "Histograms" from the "Database" menu. Then select options as follows;

- **X-axis**: the field you saved the score.
- **Category**: "RISKCLASS"

This will produce the following histograms;







The top left figure shows overall distribution of probability scores. Although it's not obvious in the figure, the top right figure clearly indicates that higher the score, the higher probability of being risky (in red colors). The bottom figure shows relative distribution of both risky and safe customer categories.

The figures indicate that there is a room to improve in the model. You may try smaller "Support" levels in the decision tree so that finer segmentation can be produced. However, too low support may lead to overfitting. Alternatively, you may try neural network scoring methods described in the next chapter.

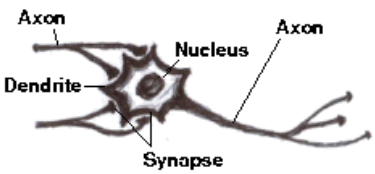
6. Neural Network Modeling

In the previous chapter, we have described a risk scoring method based on decision tree. The method provides coarse and general scoring that may not be precise. In this section, we describe a neural network modeling method that may provide much finer accuracy.

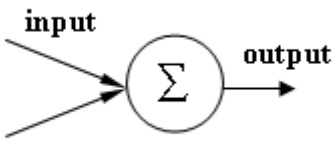
Predictive neural network is a very powerful predictive modeling technique. Neural network is derived from animal nerve systems (like human brains). The heart of the technique is neural network (or network for short). Neural networks can learn to perform variety of predictive tasks. For our interest, it can be trained to predict risk scores.

Neurons in Biological Neural Network

The core of neural network is nodes. Neural network nodes correspond to neurons of nerve systems. Anatomy of neurons (or neuro cells) is shown at the left figure. Nucleus contains DNA of neuro cells. Axons connect neurons and deliver neuro signals to other neurons. Dendrites receive signals passed through synapses. Synapse is the narrow gap between axon terminals and dendrites.



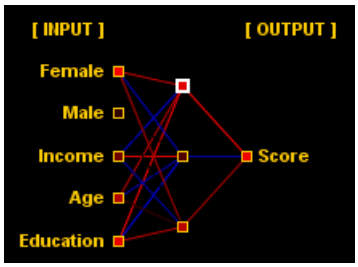
Neurons receive signals from other neurons and combine them. Combined signals are delivered to other neurons which may receive different level of signals as strength of connected axons will differ. The right figure illustrates this. Indeed, the core function of neurons is very simple!



What is Artificial Neural Network?

The core function of neurons is simple. That is, neurons combine input signals and pass to other neurons. When this simple function is organized into a network of neurons, it renders a very powerful predictive system. Artificial (or computer) version of neurons is referred to as **nodes**. Artificial neural networks consist of layers of nodes and links between nodes.

The following figure is an example of artificial neural network. (You may assume it as a credit scoring or insurance scoring predictive model which predicts score for credit/insurance applicants, based on gender, income, age and education.) The first layer is **input layer**. Nodes of input layer represent input fields or values of categorical input fields. In the figure, there are five input nodes: "Female", "Male", "Income", "Age", and "Education". The last layer is **output layer**. Nodes of output layer represent either prediction values or predicted class names. In the figure, there is a single node for "Score". The rest of layers are called **hidden layers** (or middle or internal layers). There may be zero or more hidden layers, normally a single hidden layer. The figure has three hidden nodes in a single hidden layer.



How neural network predicts?

Links in the network represent axons of biological counterparts. To simulate the fact that each axon of neurons reacts differently, each link is assigned with a different weight. Weights of links are the essence of neural networks. With weights, networks make prediction as follows. First, known values of input fields are presented to the nodes of input layer. Then, values are propagated towards the nodes of the output layer. In this process, values are multiplied with weights, summed and, then, applied to a non-linear function. Note that this is exactly how neurons combine input signals. Weights are designed in such a way that for given input patterns, values of the output layer reflect the values of actual outcome.

How neural network is trained?

Links of networks are represented by weight. Weights are computed by a method known as **back-propagation**. Computation of weights is called network training. Neural Networks are trained by "showing exemplary data and making corrections repeatedly", until networks fully learn patterns hidden inside data and are able to predict accurately. More specifically, a customer record is presented to a network along with the past result of the customer. If network predict wrongly the past result value, network will be corrected appropriately so that that the error will be reduced next time.

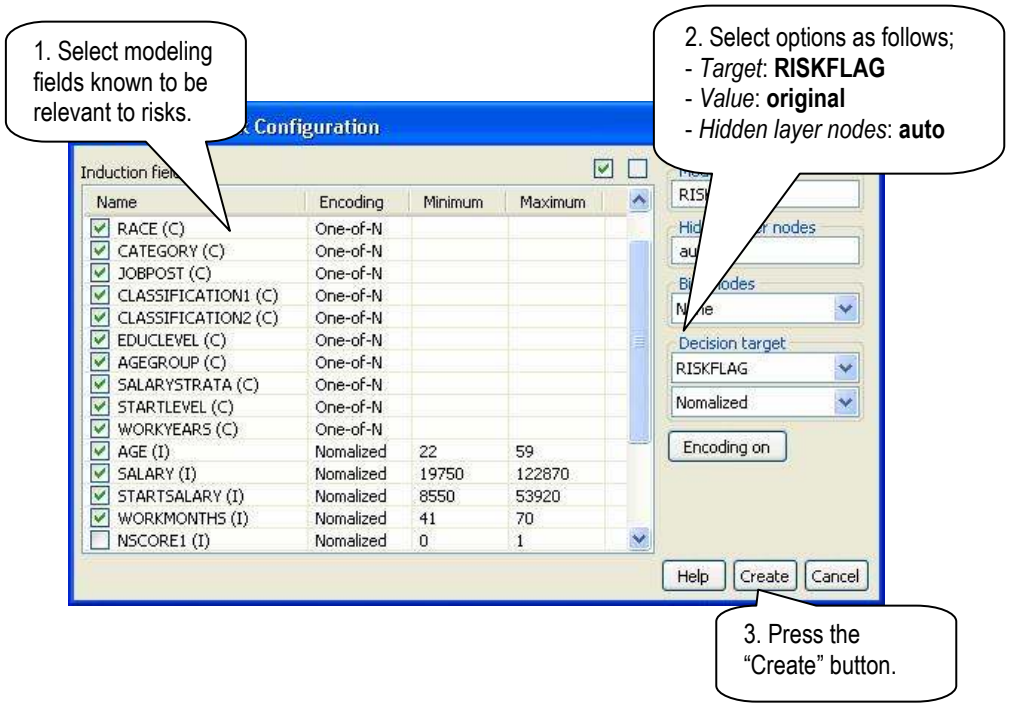
Network training is performed by repeating the following procedure. For each input training data record;

1. Present input data values to nodes of the input layer.
2. Propagate the presented input values towards the output layer (forward process).
3. Compare with the values of output nodes to the actual values of the data.
4. Correct the differences of outcome and propagate towards the input layer (backward process).

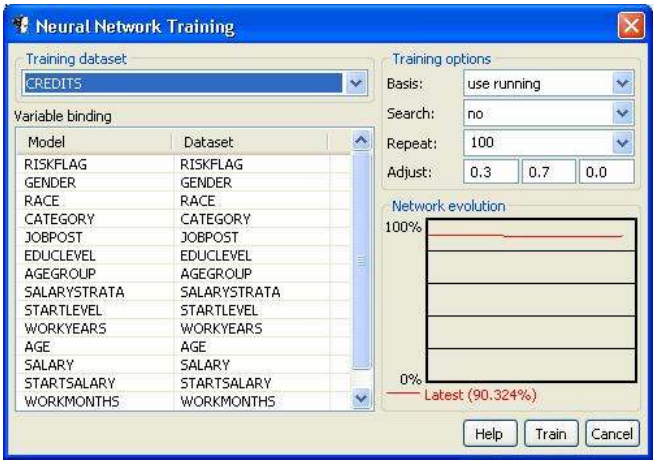
An application of entire training data records is called as “epoch”. Normally, training data records are applied many many epochs before network is actually used. Network training is a repetitive process. In the beginning, networks are trained coarsely. Then, they are refined by repeated application of input data. After networks reach a certain maturity level, they are used or deployed for value prediction.

6.1 Developing Neural Network Models

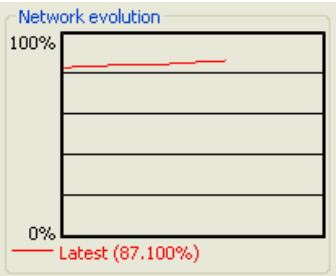
Neural network models are developed in a similar way as in decision tree. The first step is to configure a neural network and start training with an input training dataset. To develop a neural network, configure a model as described in the following figure;



Note that we select "RISKFLAG" which contains 0.0 and 1.0 as values. This contrasts with the decision tree method in which we select "RISKCLASS". Once configuration is created, a neural network is trained repetitively from the following dialog panel. Select training options and train the network;



A major difference from the decision tree is that training of neural network is iterative. It can be repeated any number of times. It is noted that in decision tree, once you select configuration, models are developed in a single phase. On the other hand, neural network training is repeated until it can reach a satisfactory level. The important question will be how do we know that network is fully or well trained? There are no automatic methods for this important task. You need to determine using the network evolution chart. The network evolution chart, as shown below, shows changes of accuracy level in latest iterations.



When network training is saturated and further training does not produce any improvement, the chart will show a flat red line indicating accuracy level is not improving.

However, don't be fooled with temporary stagnations in improvement. Neural network often remains stagnant over thousands of iterations, then suddenly reap to higher accuracy level. Train many many many times with patience!

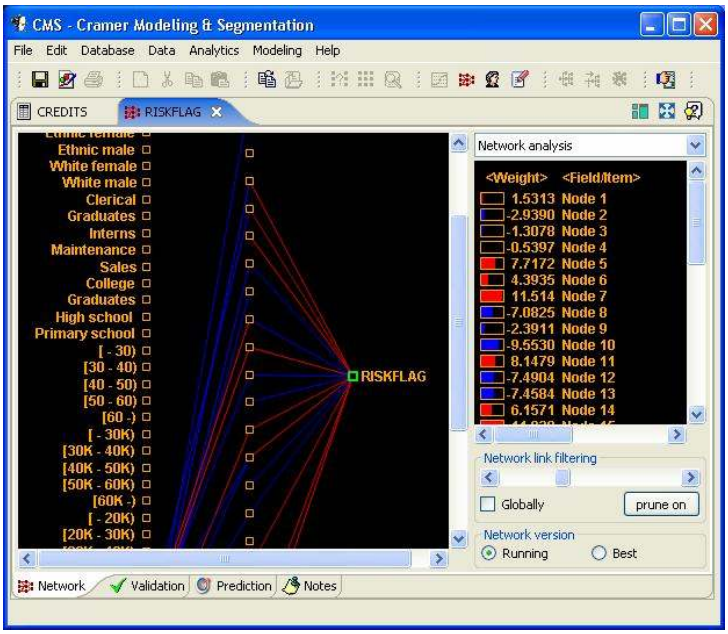
How to avoid overfitting in neural network?

Overfitting is a phenomenon where predictive models learn training datasets too detail, i.e., "overly fit" to the training data. The consequence is that models may not predict accurately on other datasets, which means that they may not accurately predict on future events. Once a network is developed, you need to verify with a number of datasets to check whether it consistently performs well. If the test result is patchy, the model may not work well on future data!

In neural network, there is a way you can control overfitting. Fewer network nodes imply lower learning capacity. By reducing the numbers of nodes, you can reduce the level of overfitting. Numbers of nodes can be controlled with "Hidden layer nodes" specification. You might try with no hidden layers first, i.e., "Hidden layer nodes" = "" (empty string). Then try "auto" configured network and compare them. From "auto" configured numbers, find a suitable number for "Hidden layer nodes". It is noted that models with "no hidden layers" represent the most general networks. Bagging with other models having suitable hidden nodes might be a good approach.

Analyzing Network

Networks can be analyzed visually from the following main panel;



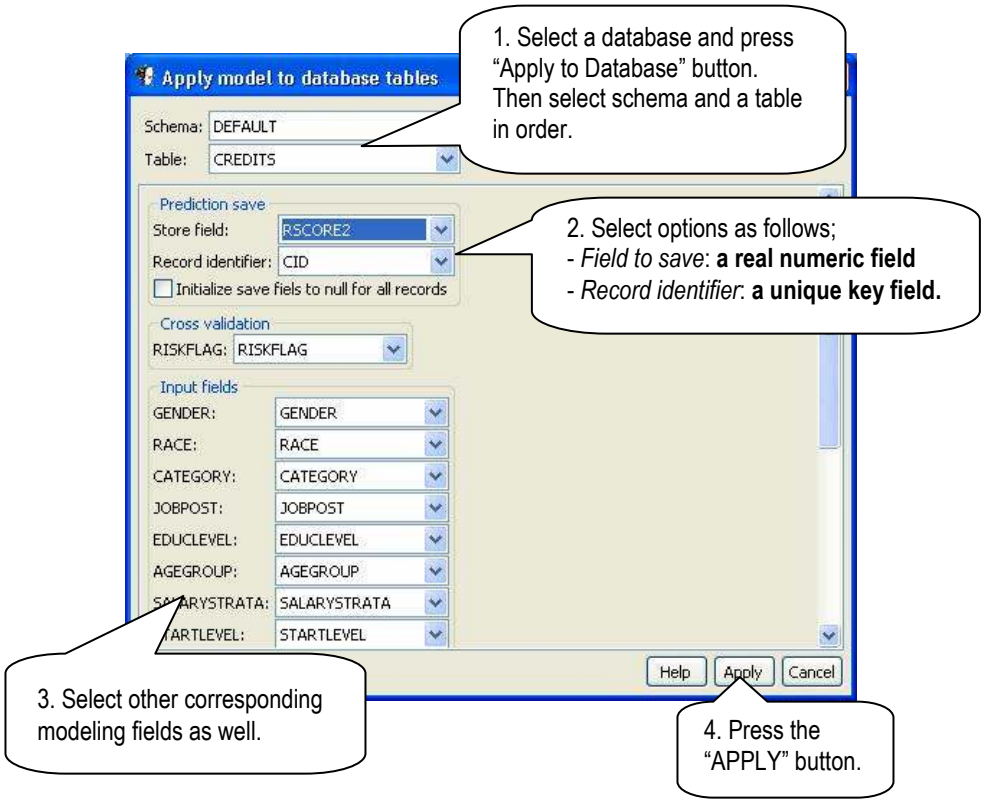
Links show connection weight between nodes. Links in red color indicate positive feeds, while links in blue indicate negative feeds. The mini panel at the right shows details of input connections of a node selected. You can choose a node by clicking the node with the mouse.

Pruning Nodes (optional)

After initial training, you might want to trim internal nodes that do not have significant roles in the network. Removing such nodes can make networks learn cleaner patterns and reduce overfitting. To prune nodes, press the “prune on” button and click the nodes to remove. When finished, click the “prune on” button again. It is important to note that when nodes are pruned, the network needs to be further re-trained using “use running” of “Training” – “Basis:” options.

6.2 Applying to Customer database

To apply the neural network to a database table, perform the steps described in the following figure. You will need a numeric field that can store REAL values with decimal points, e.g., 0.3674, etc. It is recommended to create a number of scratch fields on customer tables. They can be used in variety of validation works.



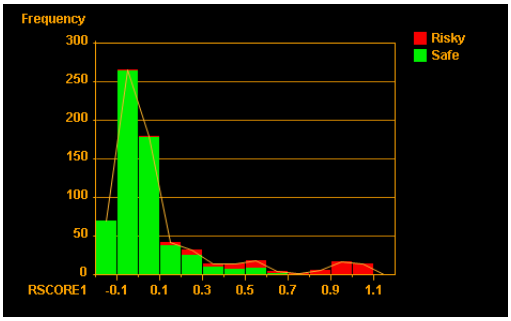
This reads database customer records one-by-one, and evaluates using the neural network. Result values will be stored on the "Field to save".

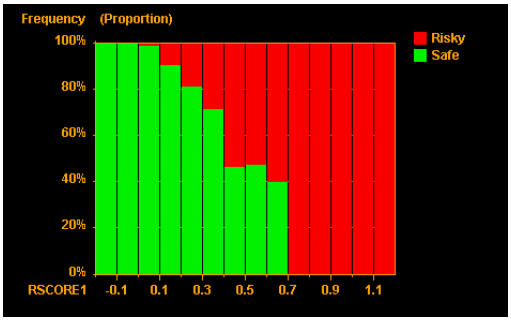
6.3 Analyzing Score Distribution

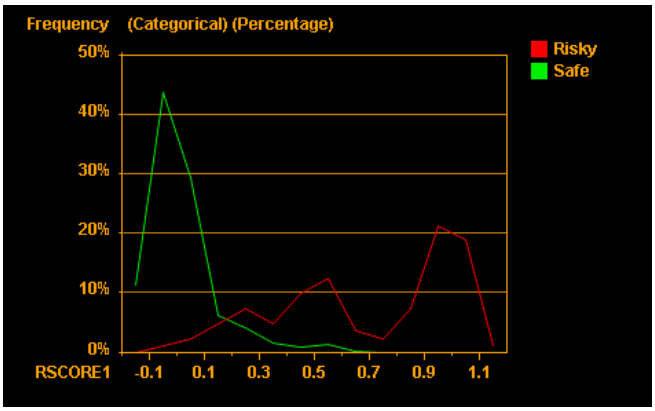
Once you apply neural network to a database table, you can analyze score distribution using “dimensional histograms”. Dimensional histograms are available in CMSR. Select “Histograms” of the “Database” menu. Then select options as follows;

- **X-axis**: the field you saved the score.
- **Category**: “RISKCLASS”

This will produce the following histograms;







The top left figure shows overall distribution. Although it’s not obvious in the figure, the top right figure clearly shows that risky customers are at higher score ranges. Notice the marked improvement from the one using decision tree! The bottom figure shows relative distribution of both risky and safe customer categories. Risky customers are concentrated at around 1.0, while safe customers at around 0.0. *The top right chart shows intuitive visualization of scores and risk. It can be used in model user documentations.*

It is important to note that this wonderful result may be an indication of overfitting. This was trained using a relatively very small number of training data records. In practice, you need to use a large dataset and perform rigorous tests to avoid overfitting.

6.4 Modeling Risk Amounts

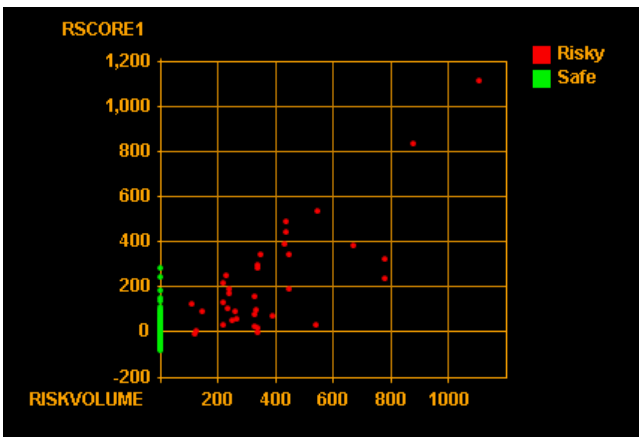
In the previous sections, we have described procedures developing risk scoring models using “RISKFLAG”. Note that the field contains values either 0 or 1. In this section, we describe how the procedures can be adapted to predict actual values, i.e., insurance claim amounts or credit default amounts.

To develop a model for risk amounts, perform the procedures described in sections “Developing Neural Network Models” and “Applying to Customer database”, with a couple of exceptions. First, at “Step 2” of section “Developing Neural Network Models”, use “RISKVOLUME” in place of “RISKFLAG”. Second, analyzing methods in section “Applying to Customer database” should be changed as follows.

Once you apply the model to a database table, the table will hold two types of amount information: actual amounts and predicted amounts. You can analyze model’s performance with the scatter plot. Select “Histograms” of the “Database” menu. Then select options as follows;

- **Independent**: the field that holds actual values
- **Dependent**: the field you saved predicted values.
- **Category**: “RISKCLASS”.

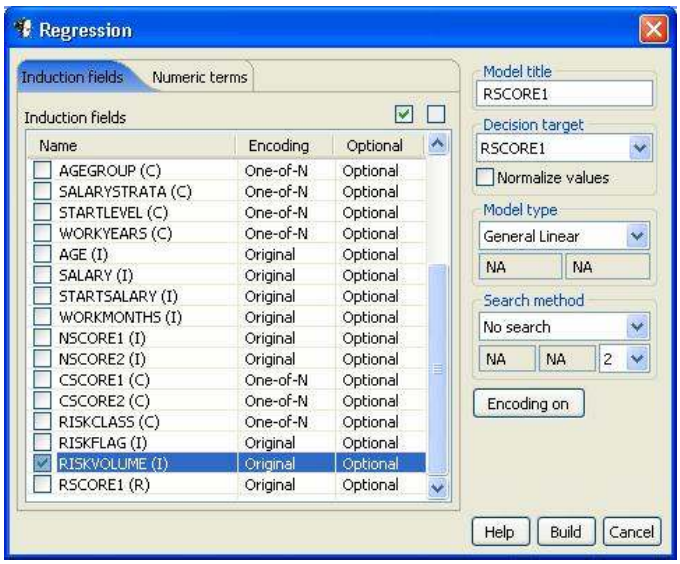
The following is an example scatter plot. Note that this chart is plaugible if data is small!



The left is a scatter plot for actual and predicted values. Perfect ones will be clustered right on 45-degree diagonal lines crossing the point (0, 0). In good models, distribution should be clustered around 45-degree diagonal lines. The figure indicates that the model is not a perfect one, but still not a bad one! When predicted values (= “RSCORE1”) are over 200, most of them are “risky”. The problematic part is when predicted values are below 200, since it can be either “Risky” or “Safe”. However most of them will be safe!

Quantifying Model Fitness

The quality of models can be quantified using linear regression on actual values and predicted values, that is to say, using correlation coefficients. To do regression, you may need to re-import the table which has the predicted values by a model. From the following regression dialog, select original value fields as induction fields and predicted values as the target.



Result can be found from ANOVA tables. By far the most important information is "R-square" values. R-square value 1.0 means perfect models, while zero means perfect rubbish models. The figure indicates "R-square = [0.6678]". By comparing R-square values, you can determine model superiority.

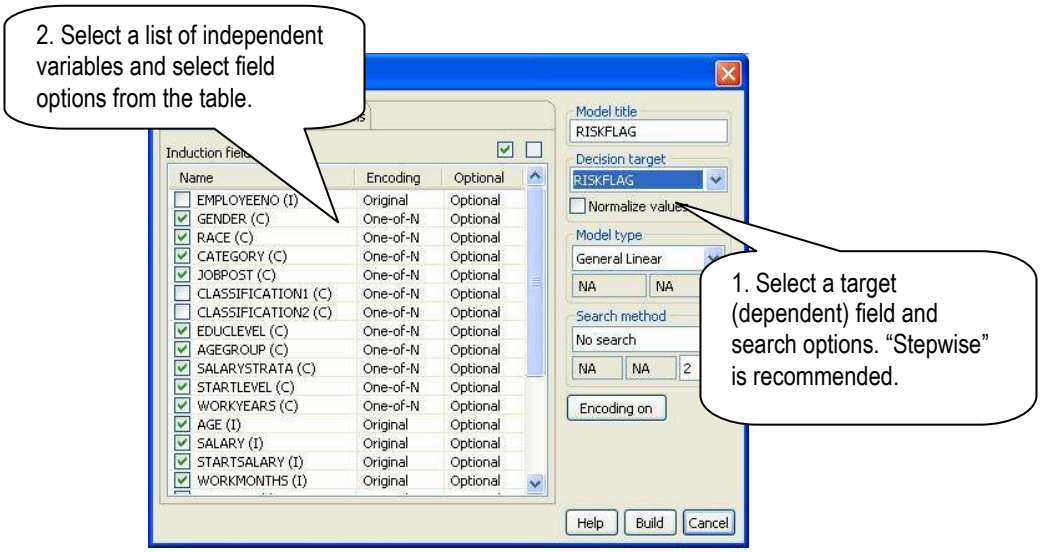
As usual, you need to perform validation on a number of different test datasets. If your models perform well consistently, you may assume that they will perform on others as well.

7. Regression Modeling

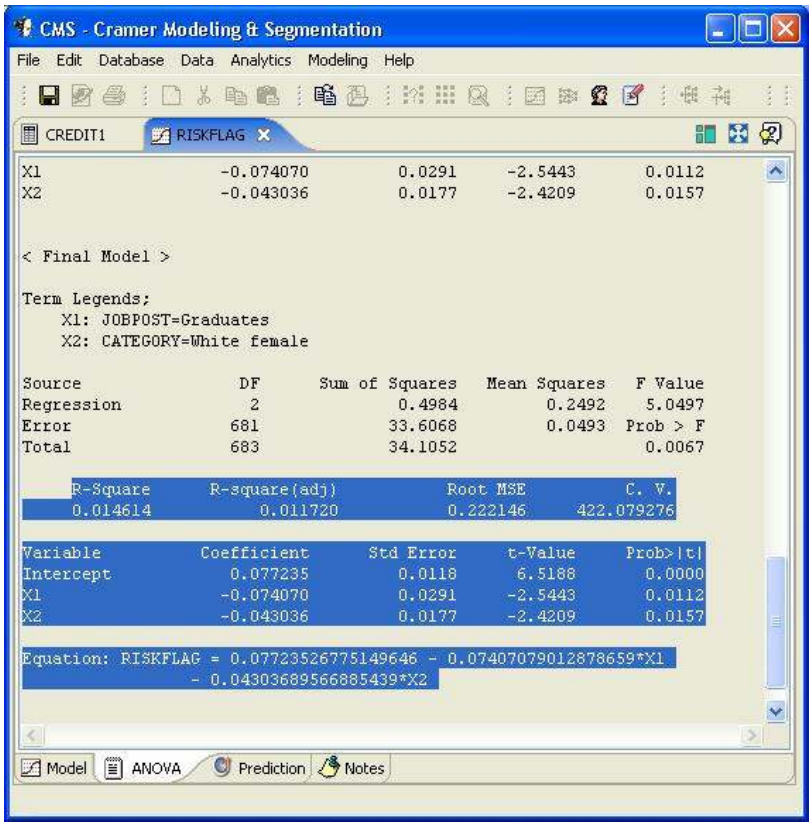
In the previous two chapters, we have described two scoring methods based on decision tree and neural network. Regression can supplement the techniques. Regression is a mathematical modeling technique which works on numerical variables. That is, both target and factor variables must be numerical. Categorical variables can be transformed into numerical variables by creating each category a numerical variable. “Gender”, for example, can be transformed into two numerical variables: one for “male”, the other for “female”. If a person is a female, the “male” field will be assigned with “0.0” and the “female” will be assigned with “1.0”.

Conventionally, regression has been used widely. However, it lacks the versatility of the advanced methods described in the previous chapters. Regression can work well if models can be developed with a smaller number of factor variable, say, one to several factor variables. In addition, if target values are extremely skewed, regression may not work at all. Note that in regression, factor variables are called independent variables. Target variables are called dependent variables. Regression is closely related to the variable relevancy analysis described in Chapter 3. Only those variables with high correlation coefficients are used in regression. Regression models may be developed to predict either risk amounts or 0.0~1.0 risk levels (as in the neural network method).

CMSR supports regression tools. Select “Regression” of the “Modeling” menu. Perform the steps described in the following figure;



Once computation is completed, the following ANOVA report panel will appear. The part highlighted in blue shows correlation coefficients and regression formula. To understand other information shown in the report, you may need to refer to statistical text books.



Regression can be very useful for developing segment-specific models. For example, in motor vehicle insurance, once certain categorical factors are removed by segmentation, the rest may be modeled as regression functions using “age of vehicles” and “mileage driven”, as combined repair costs will raise according to the factors. This could yield very accurate prediction for the segments. Segmentation can be performed using hotspot analysis and decision tree described in previous chapters.

8. Putting Them Together

In the previous chapters, three risk scoring methods are described. In this chapter, we describe how multiple predictive models can be integrated and incorporated into a single model. There are a number of reasons that we need much more advanced techniques than flat predictive models described in the previous chapters. The following is a list of methods that can improve risk scoring models.

Combining multiple models

A predictive model is developed from a single dataset using a single technique. Singular models tend to show biases one way another. To overcome shortcomings of singular models, multiple models, that may use different techniques and/or different sets of modeling fields, are developed. Use of multiple models is referred to as **bagging**. Multiple models can be developed in various ways using different information and techniques as follows;

- **Scope of models**: whole population or special segments.
- **Modeling methods**: neural network, decision tree, and mathematical models.
- **Modeling variables**: use different sets of modeling variables.
- **Model configurations**: use different model configurations.

By combing these factors, you can create robust risk scoring models. Most common approaches combining models include;

- **Pessimistic**: This will use the highest values using the *maximum* of combining scores.
- **Optimistic**: This will use the lowest values using the *minimum* of combining scores.
- **Fair**: This will try to make it fair using the *average* of combining scores.

Risk-level labeling

Models developed using decision tree, regression and neural network can predict statistical probability or scores. Based on statistical probability or scores, risk may be labeled into different levels of risk groups. This is called as *classification*. For example;

- "Low risk": less 1% risk.
- "Medium risk": less than 10% risk.
- "High risk": less than 70% risk.
- "Very high risk": over 70% risk.

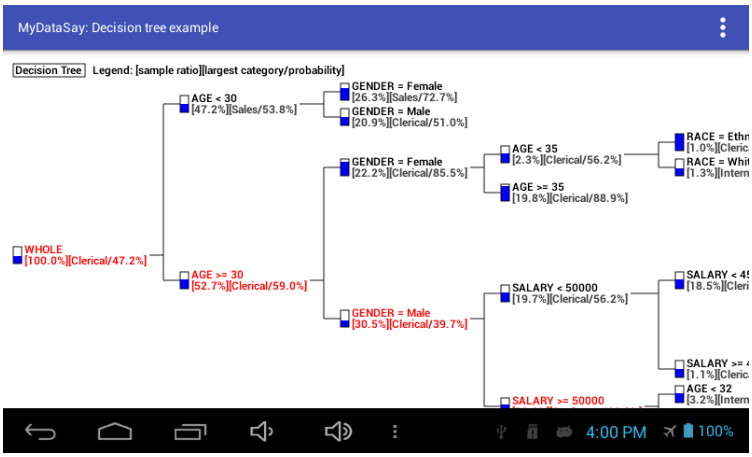
This can be easily implemented using Rules-based Modeling described in the next section. Examples (3 and 4 and 5) of labeling are given in the examples section.

9. Deployment of Models

CMSR provides an environment where you can develop and apply predictive models to your data directly. However, the tools are not suitable for non-technical end-users such as insurance actuaries and credit managers. You need to deploy them with customized interfaces to suit your end-users. Typical deployment of models is described in the following sections.

9.1 MyDataSay Android App

MyDataSay is an Android application that can be run on Android devices such as phones and tablets. MyDataSay supports predictive models such as neural network, decision tree and regression, and SOM (=neural clustering) developed from CMSR Studio. In addition, RME models are supported. The following figure shows a decision tree model on MyDataSay;

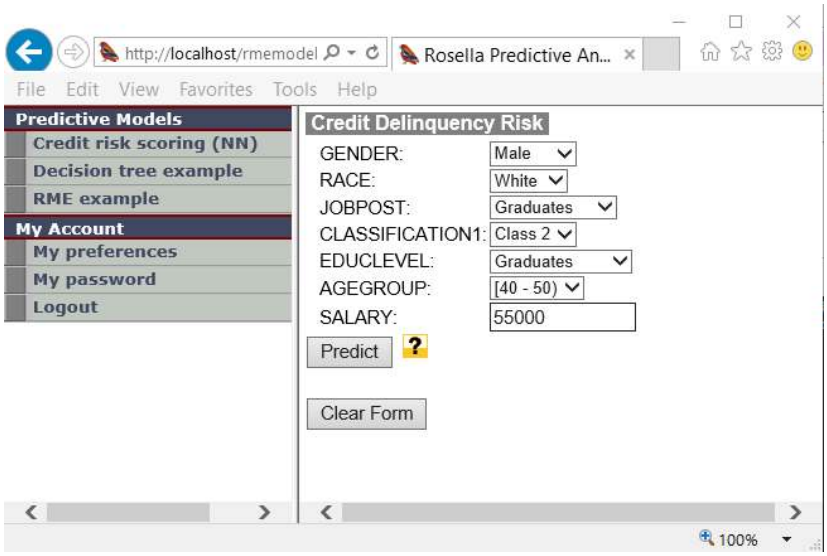


For details for MyDataSay deployment, please read the guide "MODEL-DEVELOPER-GUIDE.txt" available in the "MyDataSay.zip" install file. The "MyDataSay.zip" file can be downloaded from the following link;

http://www.roselladb.com/mydatasay.htm

9.2 Web-based Model Deployment

Rosella BI server is a simple and fast way of delivering risk models to end-users over the web. It's the easiest way to deliver models to a large number of end-users efficiently. Rosella BI server supports html-based implementation. The following examples show user-interfaces. In the left frame, a user selects a model. Then enters input data at the right frame and press the "predict" button.



For details of web-based deployment, read "Predictive Model Deployment Guide for BI Server" (RME-Web-Setup.pdf).

Business applications can call predictive models through HTTP requests with model input data in JSON format. For more details, read "HTTP/JSON Setup Guide" (HTTP-JSON-Setup.pdf).

In addition, BI server can incorporate database using JSP programs. For details, refer "JSP Program Setup Guide" (JSP-Setup-Guide.pdf).

Above mentioned documents can be found from the following BI server directory. If you use other J2EE server, replace "%TOMCAT_HOME%" with yours;

```
%TOMCAT_HOME%\webapps\rosellabi\WEB-INF
```

Appendix I. Customer Churn Analysis

Customer retention is very important because acquiring a new customer is far more expensive than keeping an existing one. This is even more true if the market is saturated or saturating. Retention is important to businesses. The most important customer retention strategy is to identify customers who are likely to leave (potentially to rival providers). Once they are identified, customer retention programs can be developed and actions can be taken to prevent churning. As it turned out, analytic churn management methods are the same as those of risk management! The following data mining techniques (customer churn analysis) can be used in identifying customer groups with high churn risk;

- **Churner variable relevancy**: Analyze which variables are indicators for potential churns, *as described in Chapter 3*.
- **Churner profiling**: Develop profiles of risky customer groups based on demographic, geographic, psychographic attributes and service usage patterns, *as described in Chapter 4*.
- **Customer churn scoring**: Build predictive models that can predict likelihood of defection and perform segmentation, *as described in Chapter 5 ~ Chapter 9*.

How customer surveys can help to identify churns

Customer surveys are very important means for identifying potential problems in your services. Most common reasons for customer defection may include inadequacy in service and high cost. Ask customers to rate your services along with their demographic and psychographic profiles. Analyzing surveys can reveal customer groups who are not happy with your services. In addition, survey information can be very important churn predictors. If survey information is available, include it in your churn detection modeling!

Data preparation for churn detection

Data for churn detection modeling can be prepared as described in Chapter "2. Data Preparation", with one exception for outcome variables. Datasets should consist of all customers including the ones who have churned already. Data for churn analysis is prepared with the following outcome fields;

- **"CHURNCLASS" :** Customers are categorized "**Churned**" and "**Staying**".
- **"CHURNFLAG" :** Churn flag value is 1.0 for churned customers. Other current customers will be assigned to 0.0.

Index

A

ANOVA 33
applying to database 20, 28
artificial neural network 23
axons 22

B

back propagation 24
bagging 34
behavioral variables 3
bogus values 5

C

categorical data 2
CHURNCLASS 37
CHURNFLAG 37
classification 15, 34
cleaning data 5
connecting database 7
correlation analysis 11
Cramer 15
credit scores 1
credit scoring 1

D

date and time 2
decision tree 15
demographic variables 3
dendrites 22
dependent variables 32
dimensional histograms 21, 29
distribution analysis 10

E

epoch 24
evolution chart 26
exception analysis 13

F

financial variables 3

G

geographic variables 3
GINI 15

H

health information 3
hidden layers 23
hotspot analysis 13

I

identifiers 2
importing data 9
independent variables 32
input layer 23
insurance scores 1
insurance scoring 1

J

JDBC 7

M

missing values 5
model deployment 35
model integration 34
model superiority 31
multiple models 34
MyDataSay 35

N

network training 24
neural network 22
neurons 22
nodes 23
nucleus 22
null 5
numerical data 2

O

ODBC 8
outcome variables 4
outliers 5
output layer 23
overfitting 6, 17, 26

P

Pareto principle 13
probability scoring 15
psychographic variables 3

Q

quality of models 31

R

regression.....31, 32
risk management.....1
risk scoring 1
RISKCLASS.....4, 17
RISKFLAG.....4, 25
risk-level labeling 34
RISKVOLUME.....4
R-square.....31

S

scatter plot 30
score distribution 21, 29
segmentation.....15
synapse 22
synonymous names.....5

T

test datasets.....6
textual information 2
training datasets.....6
Twoing.....15

V

validation 19
variable relevancy.....10

W

web deployment.....36