Building Credit Scoring Models with SAS® Enterprise Miner

The Power to Know®

A SAS White Paper

Table of Contents

Introduction

Over the past 30 years, growing demand, stronger competition and advances in computer technology have meant that the traditional methods of making credit decisions that relied mostly on human judgment have been replaced by methods that employ statistical models. Statistical models today are used not only for deciding whether or not to accept an applicant (application scoring), but also for predicting the likelihood of defaults among customers who have already been accepted (behavioral scoring) and for predicting the likely amount of debt that the lender can expect to recover (collection scoring).

The term ***credit scoring*** can be defined on several conceptual levels. Most fundamentally, credit scoring means applying a statistical model to assign a risk score to a credit application or to an existing credit account. On a higher level, credit scoring also means the process of developing such a statistical model from historical data. On yet a higher level, the term also refers to monitoring the accuracy of one or many such statistical models and monitoring the effect that score-based decisions have on key business performance indicators.

Credit scoring is performed because it provides a number of important business benefits, all of them based on the ability to quickly and efficiently obtain fact-based and accurate predictions of the credit risk of individual applicants or customers. For example, in application scoring, credit scores are used for optimizing the approval rate of credit applications.

Application scores enable an organization to choose an optimal cut-off score for acceptance, such that market share can be gained while retaining maximum profitability. The approval process and the marketing of credit products can be streamlined based on credit scores. For example, high-risk applications can be given to more experienced staff, or pre-approved credit products can be offered to select low-risk customers via various channels, including direct marketing and the Web.

Credit scores, both of prospects and existing customers, are essential in the customization of credit products. They are used to determine customer credit limits, down payments, deposits and interest rates.

Behavioral credit scores of existing customers are used in the early detection of high-risk accounts, and they enable organizations to perform targeted interventions, such as proactively offering debt restructuring. Behavioral credit scores also form the basis for more accurate calculations of the total consumer credit risk exposure, which can result in a reduction of bad debt provision.

Other benefits of credit scoring include an improved targeting of audits at high-risk accounts, thereby optimizing the workload of the auditing staff. Resources spent on debt collection can be optimized by targeting collection activities at accounts with a high collection score. Collection scores are also used for determining the accurate value of a debt book before it is sold to a collection agency. Finally, credit scores serve to assess the quality of portfolios intended for acquisition and to compare the quality of business from different channels, regions and suppliers.

Building credit models in-house

While under certain circumstances it is appropriate to buy “ready-made” generic credit models from outside vendors or to have credit models developed by outside consultants for a specific purpose, maintaining a practice for building credit models in-house offers several advantages.

Most directly, it enables the lending organization to profit from economies of scale when many models need to be built. It also enables lenders to afford a greater number of segment-specific models for a greater variety of purposes.

Building a solid, internal skill base of its own also makes it easier for the organization to remain consistent in the interpretation of model results and reports and to use a consistent modeling methodology across the whole range of customer-related scores. This results in a reduced turnaround time for the integration of new models, thereby freeing resources to respond more swiftly to new business questions with creative new models and strategies.

Finally, in-house modeling competency is needed to verify the accuracy and to analyze the strengths and weaknesses of acquired credit models, to reduce outsider access to strategic information and to retain competitive advantage by building up company-specific best practices.

Building credit models with SAS® Enterprise Miner

SAS Enterprise Miner software is SAS’ solution for data mining. It is used across many industries to answer a variety of business questions and has been extended with specific functionality for credit scoring that is described in more detail in the case study section.

Building credit models with SAS Enterprise Miner offers a number of benefits. It enables the analyst to access a comprehensive collection of data mining tools through a graphical user interface and to create process flow diagrams that structure and document the flow of analytical activities. The various nodes that make up the process flow are designed such that the analyst can interact with data and models to bring in fully the domain expertise—i.e., use the software as a steering wheel and not as an auto-pilot. SAS Enterprise Miner is ideal for testing new ideas and experimenting with new modeling approaches in an efficient and controlled manner. This includes the creation and comparison of various scorecard, decision tree and neural network models, to name just a few.

SAS® Enterprise Miner process flow templates

SAS Enterprise Miner process flow diagrams can serve as templates for implementing industry or company standards and best practices. Such templates not only reduce the development time for new models, but also ensure consistency and an efficient transfer of ability to new employees.

The process flow that is used in the case study in this paper is available from SAS and can serve as a basic credit scoring template. It enables the analyst to build a scorecard model that assigns score points to customer attributes, to use the Interactive Grouping node to class and select characteristics automatically and/or interactively using Weights of Evidence and Information Value measures, and to normalize score points to conform to company or industry standards. As an alternative model type, the template builds a decision tree.

The larger credit scoring process

Modeling is the process of creating a scoring rule from a set of examples. In order for modeling to be effective, it has to be integrated into a larger process. Let's look at application scoring. On the input side, before the modeling step, the set of example applications must be prepared. On the output side, after the modeling, the scoring rule has to be executed on a set of new applications, so that credit granting decisions can be made. The collection of performance data is at the beginning and at the end of the credit scoring process. Before a set of example applications can be prepared, performance data has to be collected so that applications can be tagged as "*good*" or "*bad*." After new applications have been scored and decided upon, the performance of the accepted accounts again must be tracked and reports created. By doing so, the scoring rules can be validated and possibly substituted, the acceptance policy finely tuned and the current risk exposure calculated.

SAS' power to access and transform data on a huge variety of systems ensures that modeling with SAS Enterprise Miner smoothly integrates into the larger credit scoring process. SAS software is the ideal tool for building a risk data warehouse.[1] This is a subject-oriented, integrated, time-variant and nonvolatile repository of information[2] that serves as the integration hub for all risk management-related decision-support processes, including scorecard monitoring reports and risk exposure calculations.

SAS Enterprise Miner creates portable scoring code that can be executed on a large variety of host systems. For example, the scoring code can be used for scoring a large customer segment centrally in batches, or it can be integrated into applications that score individual applicants in branch offices.

Choosing the right model

With SAS Enterprise Miner, it is possible to create a variety of model types, such as scorecards, decision trees or neural networks. When you evaluate which model type is best suited for achieving your goals, you may want to consider criteria such as the ease of applying the model, the ease of understanding it and the ease of justifying it.

At the same time, for each particular model of whatever type, it is important to assess its predictive performance, such as the accuracy of the scores that the model assigns to the applications. A variety of business-relevant quality measures are used for this. The best model will be determined both by the purpose for which the model will be used and by the structure of the data set on which it is validated.

Scorecards

The traditional form of a credit scoring model is a scorecard. This is a table that contains a number of questions that an applicant is asked (called characteristics) and, for each question, a list of possible answers (called attributes). For example, one characteristic may be the age of the applicant, and the attributes for this characteristic are then a number of age ranges into which an applicant can fall. For each answer, the applicant receives a certain number of points—more if the attribute is one of low risk, fewer if the risk is higher. If the application's total score exceeds a specified cut-off number of points, it is recommended for acceptance.

Such a scorecard model, apart from being a long-established method in the industry, still has several advantages when compared with more recent "data mining" types of models, such as decision trees or neural networks. To begin with, a scorecard is easy to apply.

If needed, the scorecard can be evaluated on a sheet of paper in the presence of the applicant. The scorecard is also easy to understand. The number of points for one answer doesn't depend on any of the other answers, and across the range of possible answers for one question, the number of points usually increases in a simple way (often monotonically, or even linearly). Therefore, it is often easy to justify to the applicant a decision that is made on the basis of a scorecard. It is possible to disclose groups of characteristics where the applicant has a potential for improving the score and to do so in broad enough terms not to risk manipulated future applications.

Decision trees

On the other hand, a decision tree may outperform a scorecard in terms of predictive accuracy, because unlike the scorecard, it detects and exploits interactions between characteristics. In a decision tree model, each answer that an applicant gives determines what question is asked next. If the age of an applicant is, for example, greater than 50, the model may suggest granting a credit without any further questions, because the average *bad* rate of that segment of applications is sufficiently low. If, on the other extreme, the age of the applicant is below 25, the model may suggest asking about time on the job next. Then, credit might be granted only to those that have exceeded 24 months of employment, because only in that sub-segment of younger adults is the average *bad* rate sufficiently low.

Thus, a decision tree model consists of a set of "if … then … else" rules that are still quite straightforward to apply. The decision rules are also easy to understand, perhaps even more so than a decision rule that is based on a total score that is made up of many components.

However, a decision rule from a tree model, while easy to apply and understand, may be hard to justify for applications that lie on the border between two segments. There will be cases where an applicant will, for example, say: “If I had only been two months older, I would have received credit without further questions, but now I am asked for additional securities. That is unfair.” That applicant may also be tempted to make a false statement about his or her age in the next application, or simply go elsewhere for financial services.

Even if a decision tree is not used directly for scoring, this model type still adds value in a number of ways. The identification of clearly defined segments of applicants with a particularly high or low risk can give dramatic new insight into the risk structure of the entire customer population. Decision trees are also used in scorecard monitoring, where they identify segments of applications where the scorecard underperforms.

Neural networks

With the decision tree, we could see that there is such a thing as a decision rule that is too easy to understand and thereby invites fraud. Ironically speaking, there is no danger of this happening with a neural network. Neural networks are extremely flexible models that combine characteristics in a variety of ways. Their predictive accuracy can be far superior to scorecards and they don’t suffer from sharp “splits” as decision trees sometimes do.

However, it is virtually impossible to explain or understand the score that is produced for a particular application in any simple way. It can be difficult to justify a decision that is made on the basis of a neural network model. In some countries, it may even be a legal requirement to be able to explain a decision and such a justification must then be produced with additional methods. A neural network of superior predictive power is therefore best suited for certain behavioral or collection scoring purposes, where the average accuracy of the prediction is more important than the insight into the score for each particular case. Neural network models cannot be applied manually like scorecards or simple decision trees, but require software to score the application. However, their use is just as simple as that of the other model types.

Case study

Scenario

An international financial services organization entered the consumer credit market in a large western European country two years ago. So far, it has been operating with the use of a generic scorecard for application scoring, but now has collected enough performance data to create its own custom scorecard. The company has been offering various types of consumer loans via various channels and the first custom scorecard will be applicable to applicants from all channels. Channel-specific scorecards may later be created as required.

SAS® Enterprise Miner process flow

SAS Enterprise Miner software is used for building the scorecard. SAS Enterprise Miner enables the analyst to access a comprehensive collection of analytical tools through a graphical user interface. It provides a workspace onto which nodes (tool-icons) are dropped from a tools palette. Nodes are then connected to form process flow diagrams (PFDs) that structure and document the flow of analytical activities that are carried out. The SEMMA concept (Sample, Explore, Modify, Model and Assess) serves as a guideline for creating process flows and nodes are grouped accordingly in the tools palette.[3,4]

Figure 1 shows the process flow for modeling on the *accepts* data. All components of the flow are discussed in more detail in the sections below. The flow begins with reading in the development sample. After using the Data Partition node to split off part of the sample for later validation, the flow divides into a scorecard branch consisting of the Interactive Grouping node and Scorecard node and a decision tree branch consisting of the Decision Tree node. The quality of the scorecard and the tree are then compared on the validation data with the Model Comparison node.

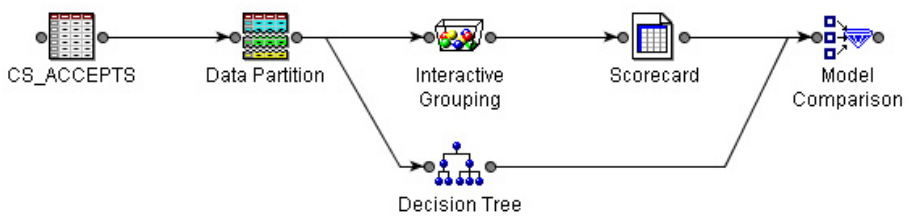

Figure 1: Process flow diagram – "accepts" data.

Development sample

The development sample (input data set) is a balanced sample consisting of 1500 *good* and 1500 *bad* accepted applicants. "*Bad*" has been defined as having been 90 days past due once. Everyone not "*bad*" is "*good*," so there are no "indeterminates." A separate data set contains the data on *rejects*.

The modeling process, especially when the validation charts are involved, requires information about the actual *good/bad* proportion in the *accept* population. Sampling weights are used here for simulating that proportion. A weight of 30 is assigned to a *good* application and a weight of 1 to a *bad* one. Thereafter all nodes in the process flow diagram treat the sample as if it consisted of 45,000 *good* applications and 1,500 *bad* applications. *Figure 3* shows the distribution of *good/bad* after the application of sampling weights. The *bad* rate is 3.23 percent.

A Data Partition node then splits a 30 percent validation data set away from the development sample. Models will later be compared based on this validation data set.

Name	Label	Role	Level
AGE	Age	Input	Interval
BUREAU	Credit Bureau R	Input	Nominal
CAR	Type of Vehicle	Input	Nominal
CARDS	Credit Cards	Input	Nominal
CASH	Requested cash	Input	Interval
CHILDREN	Num of Children	Input	Nominal
DIV	Large region	Input	Binary
EC_CARD	EC_card holders	Input	Binary
FINLOAN	Num finished Lo	Input	Binary
GB	Good/Bad	Target	Binary
INC	Salary	Rejected	Nominal
INC1	Salary+ec_card	Rejected	Nominal
INCOME	Income	Input	Interval
LOANS	Num of running	Input	Nominal
LOCATION	Location of Cred	Input	Binary
NAT	Nationality	Input	Nominal
NMBLOAN	Num Mybank Lo	Input	Nominal
PERS_H	Num in Househo	Input	Nominal
PRODUCT	Type of Busines	Input	Nominal
PROF	Profession	Input	Nominal
REGN	Region	Input	Nominal
RESID	Residence Type	Input	Binary
STATUS	Status	Input	Nominal
TEL	Telephone	Input	Nominal
TITLE	Title	Input	Binary
TMADD	Time at Address	Input	Interval
TMJOB1	Time at Job	Input	Interval
freq		Frequency	Binary

Figure 2: Variable list – development sample.

Classing

Classing is the process of automatically and/or interactively binning and grouping interval, nominal or ordinal input variables in order to:

- Manage the number of attributes per characteristic.
- Improve the predictive power of the characteristic.
- Select predictive characteristics.
- Make the Weights of Evidence—and thereby the number of points in the scorecard—vary smoothly or even linearly across the attributes.

The number of points that an attribute is worth in a scorecard is determined by two factors:

- The risk of the attribute relative to the other attributes of the same characteristic.
- The relative contribution of the characteristic to the overall score.

The relative risk of the attribute is determined by its “Weight of Evidence.” The contribution of the characteristic is determined by its coefficient in a logistic regression (see section Logistic Regression below).

The Weight of Evidence of an attribute is defined as the logarithm of the ratio of the proportion of *goods* in the attribute over the proportion of *bads* in the attribute. High negative values therefore correspond to high risk, high positive values correspond to low risk. See *Equation 1* and the middle right of *Figure 3.*

$$WeightOfEvidence_{attribute} = \log \frac{p^{good}_{attribute}}{p^{bad}_{attribute}},$$

where

$$p^{good}_{attribute} = \frac{\#goods_{attribute}}{\#goods}$$

$$p^{bad}_{attribute} = \frac{\#bads_{attribute}}{\#bads}$$

Equation 1: Weight of evidence.

Since an attribute's number of points in the scorecard is proportional to its Weight of Evidence (see section Score Points Scaling below), the classing process determines how many points an attribute is worth relative to the other attributes of the same characteristic.

After classing has defined the attributes of a characteristic, the characteristic's predictive power (i.e., its ability to separate high risks from low risks) can be assessed with the so-called Information Value measure. This will aid the selection of characteristics for inclusion in the scorecard. The Information Value is the weighted sum of the Weights of Evidence of the characteristic's attributes. The sum is weighted by the difference between the proportion of *goods* and the proportion of *bads* in the respective attribute.

$$InformationValue = \sum \left(p^{good}_{attribute} - p^{bad}_{attribute}\right) * WeightOfEvidence_{attribute}$$

Equation 2: Information value.

The Information Value should be greater than 0.02 for a characteristic to be considered for inclusion in the scorecard. Information Values lower than 0.1 can be considered weak, smaller than 0.3 medium and smaller than 0.5 strong. If the Information Value is greater than 0.5, the characteristic may be over-predicting, meaning that it is in some form trivially related to the *good/bad* information.

Classing in SAS Enterprise Miner takes place in the Interactive Grouping node. This node has been specifically developed for credit scoring applications. *Figure 3* shows a screenshot for the grouping of the interval-scaled input variable "age." The chart on the top left shows the ranges that define the grouping overlaid on a histogram of the distribution. When the variable "age" is selected, a splitting algorithm automatically suggests a grouping, which then can be modified in a variety of ways manually. Whenever a change is made, the statistics that describe the current grouping are updated.

Those statistics include the distribution of the attributes (bottom left) and the Weight of Evidence and *bad* rate per attribute (middle right). These numbers can also be read from a table (bottom right). The Information Value is also updated as the grouping is modified. The grouping of nominal and ordinal variables is performed similarly, respecting the specific differences that are implied by the measurement levels. For example, a group of ordinal values can only be merged with a neighboring group, whereas nominal values can be freely moved between groups.

There is no single criterion that indicates when a grouping can be considered satisfactory. A linear or at least a monotone increase or decrease of the Weights of Evidence is often what is desired in order for the scorecard to appear plausible. Some analysts will always include only those characteristics where a sensible regrouping can achieve this. Others may consider a smooth variation sufficiently plausible and would include a non-monotone characteristic such as “income,” where risk is high for both high and low incomes, but low for medium incomes, provided the Information Value is high enough.

In our case, we chose the characteristics “Age,” “Time on the Job,” “EC Card Holder,” “Customer Status,” “Income” and “Number of Persons in the Household.” All of these have an Information Value greater than 0.1. For some of the variables, the suggested groupings were manually modified to smooth the Weights of Evidence charts. The “Income” characteristic was intentionally included as a non-monotone characteristic.

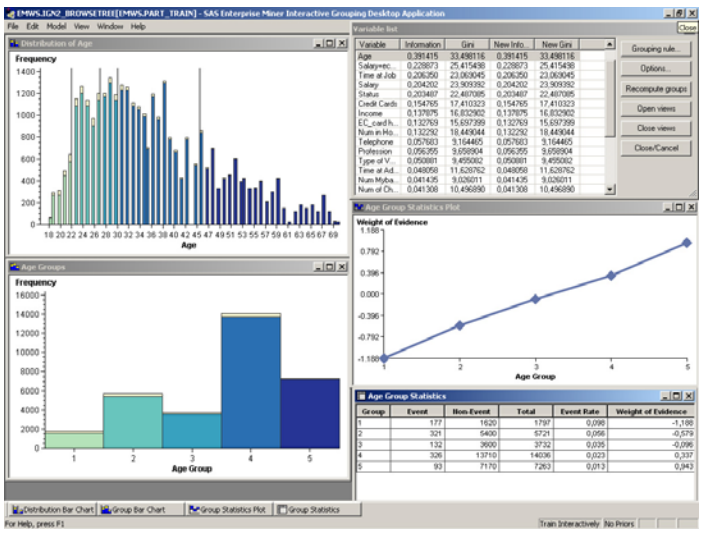

Figure 3: Interactive grouping node – grouping an interval variable logistic regression.

After the relative risk across attributes of the same characteristic has been quantified, a logistic regression analysis now determines how to weigh the characteristics against each other. The Scorecard node receives one input variable for each characteristic. This variable contains as values the Weights of Evidence of the characteristic’s attributes (see *Table 1* for an example of Weight of Evidence coding). Note that Weight of Evidence coding is different from dummy variable coding, in that single attributes are not weighted against each other independently, whereas whole characteristics are weighted, thereby preserving the relative risk structure of the attributes as determined in the classing stage[5].

Customer ID	Age	Age – Group	Age – Weight of Evidence
1	20	1: until 22	-1.1
2	31	2: 27 until 35	0.2
3	49	3: 44—58	0.9

Table 1: Weight of evidence coding.

A variety of further selection methods (forward, backward, stepwise) can be used in the Scorecard node to eliminate redundant characteristics. In our case, we use a simple regression. *Table 2* shows the values of the regression coefficients. In the following step, these values are multiplied with the Weights of Evidence of the attributes to form the basis for the score points in the scorecard.

Variable	Intercept	Age	EC Card	Income	Status	Time at Job
Parameter Estimate	-3.40237	-0.66699	-0.75488	-0.29083	-0.378789	-0.60891

Table 2: Parameter estimates.

Score points scaling

For each attribute, its Weight of Evidence and the regression coefficient of its characteristic could now be multiplied to give the score points of the attribute. An applicant's total score would then be proportional to the logarithm of the predicted *bad/good* odds of that applicant.

However, score points are commonly scaled linearly to take more friendly (integer) values and to conform to industry or company standards. We scale the points such that a total score of 600 points corresponds to *good/bad* odds of 50 to 1 and that an increase of the score of 20 points corresponds to a doubling of the *good/bad* odds. For the derivation of the scaling rule that transforms the score points of each attribute see *Equations 3 and 4*. The scaling rule is implemented in the Scorecard node (see *Figure 1*), where it can be easily parameterized. The resulting scorecard is output as a table and is shown in *Table 3*.

Note how the score points of the various characteristics cover different ranges. The score points develop smoothly and—with the exception of the "Income" variable—also monotonically across the attributes.

$$
\begin{aligned}
score &= \log(odds) * factor + offset = \\
&\left(-\sum_{t=1}^{n} (woe_1 * \beta_1) + \alpha \right) * factor + offset = \\
&\left(-\sum_{t=1}^{n} \left(woe_1 * \beta_1 + \frac{\alpha}{n} \right) \right) * factor + offset = \\
&\sum_{t=1}^{n} \left(-\left(woe_1 * \beta_1 + \frac{\alpha}{n} \right) * factor + \frac{offset}{n} \right)
\end{aligned}
$$

Equation 3: Score points scaling.

$600 = \log(50) * factor + offset$
$620 = \log(100) * factor + offset$

$factor = 20/\log(2)$
$offset = 600 - factor * \log(50)$

Equation 4: Calculation of scaling parameters.

Characteristic Name	Attribute	Scorecard Points
AGE	1	73
AGE	2	85
AGE	3	99
AGE	4	107
AGE	5	116
AGE	6	128
EC_CARD	1	93
EC_CARD	2	109
INCOME	1	106
INCOME	2	92
INCOME	3	88
INCOME	4	92
INCOME	5	97
INCOME	6	101
PERS_H	1	100
PERS_H	2	96
STATUS	1	89
STATUS	2	98
STATUS	3	103
TMJOB1	1	86
TMJOB1	2	99
TMJOB1	3	119

Table 3: Scorecard.

Scorecard assessment

The Scorecard node produces various charts and measures that help assess the quality of the scorecard. As a first insight into the usefulness of the scorecard, a score distribution chart shows the range of the score, which score bands are most frequent, if the distribution is approximately normal and if outliers exist.

Various measures and charts are then used to evaluate the discriminatory power of the scorecard. These charts analyze the scorecard's ability to separate the *good* from the *bad* cases by their scores. Measures include the Kolmorogov-Smirnoff (KS) statistic, the Gini coefficient and the area under the ROC chart (AUROC). Corresponding to these measures, the KS chart, the Captured Bad chart and the ROC chart are inspected (see *Figures 4 – 6*).

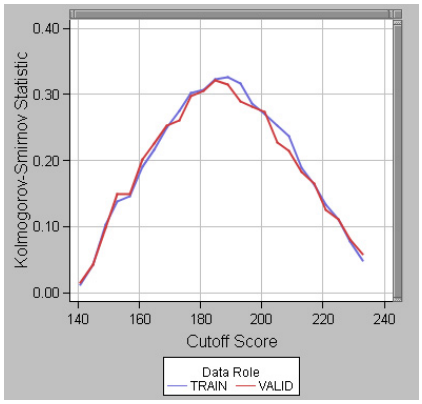

Figure 4: The KS chart shows the difference between the cumulative distributions of "goods" and "bads." The maximum value is the well-known KS statistic.

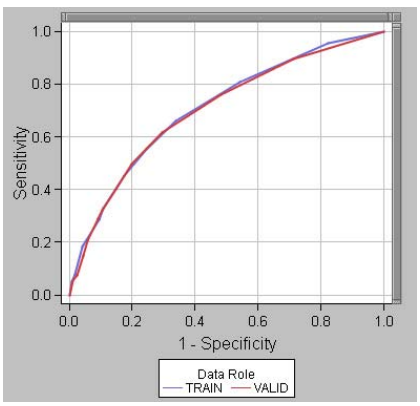

Figure 5: The ROC chart shows how well the model is able to be specific (catch only "bads") and sensitive (catch all "bads") simultaneously. Sensitivity and 1-Specificity are displayed for various cut-off values. The more the chart bends to the top left, the better.

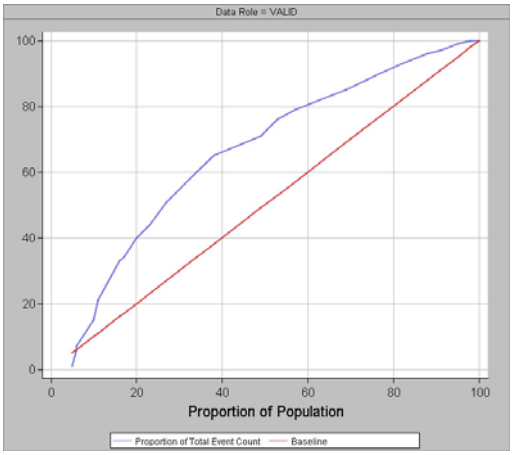

Figure 6: The Captured Bad Chart shows the proportion of all "bads" concentrated in each decile of applicants, ordered by score from worst to best. The more the chart bends to the top left, the better. The area under the chart corresponds to the Gini statistic.

In application scoring, trade-off charts are used that show how the approval rate and the *bad* rate in the *accepts* depend on the cut-off score. Good scorecards enable the choice of a cut-off that corresponds to a relatively high approval rate and a relatively low *bad* rate.

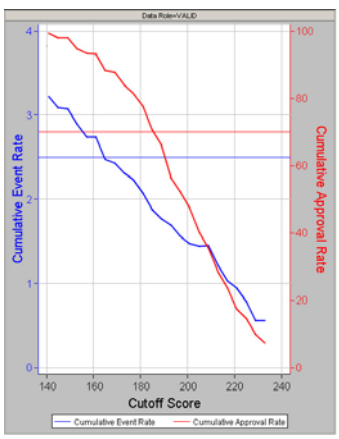

Figure 7: The Trade-off chart displays approval rate and bad rate against cut-off score for the current model. Horizontal lines show values of the previously used model and cut-off and are used to show the expected benefit from updating the model.

Finally, an empirical odds plot is used to evaluate the calibration of the scorecard.

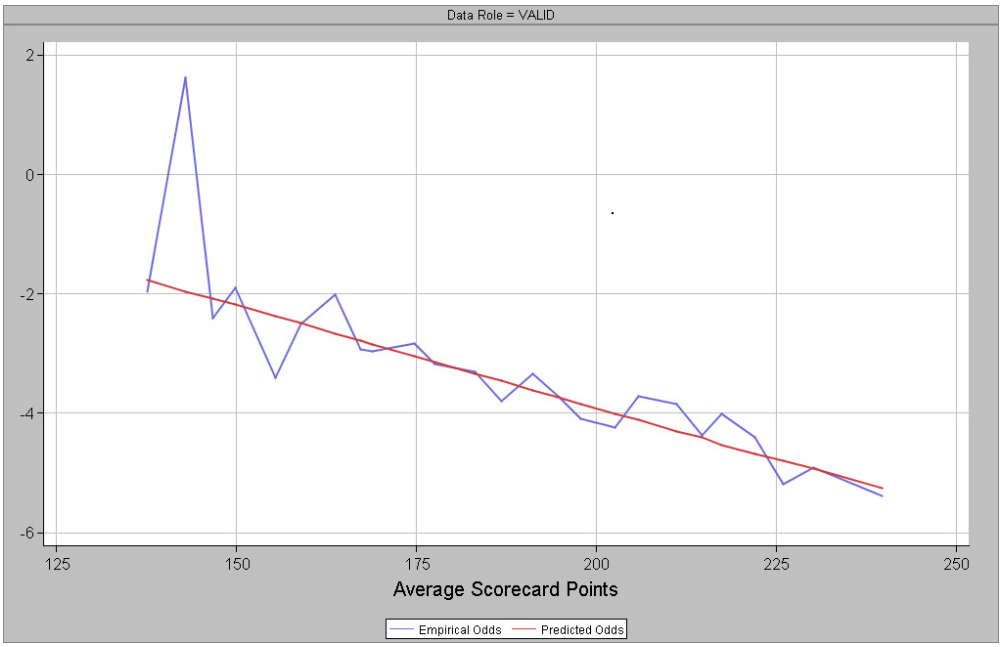

Figure 8: Actual against predicted odds.

The chart plots the actual odds values as they are found in the validation data against score band. This chart is overlaid with a chart of the values that are predicted by the scorecard. The chart thus determines those score bands where the scorecard is or is not sufficiently accurate.

Decision tree

After having gone through the process of building a scorecard model, let's take a step back and build a decision tree model instead. As previously discussed, this model type can often have superior performance, because it exploits interactions between variables. It defines segments of extremely high or extremely low *bad* rates and can thus give surprising new insights into the risk structure of the population.

The decision tree algorithm is called from the Tree node. It begins by splitting the input variables into groups in much in the same way as the Interactive Grouping node does

However, then it goes on to choose the most predictive variable and to group the applicants into segments according to the split of that variable. It then continues recursively in each segment of applicants to split the variables, choose the most predictive one and group the applicants. This process continues until no further partitioning seems useful. Finally, some of the terminal sub-segments (leaves) are merged back together again (pruned) in order to optimize the tree.

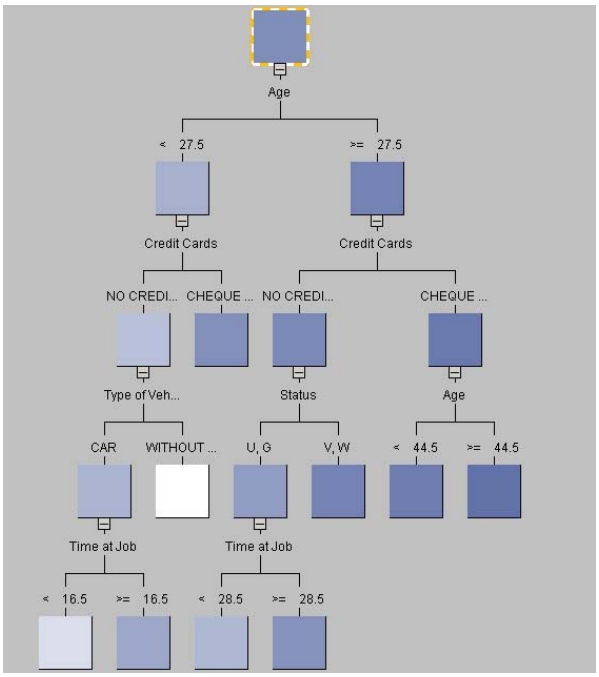

Figure 9: Decision tree.

Figure 9 shows the resulting tree structure for our case. The most important variable is age, with applicants younger than 28 having an elevated credit risk. Furthermore, among the young, those with no credit cards and no own vehicle have the highest risk of all applicants. For those with a car, the time on the job becomes important. In the group of older applicants, customer status delivers important information.

Model comparison

After building both a scorecard and a decision tree model, we now want to compare the quality of the two models. The model comparison node is used for that purpose. *Figure 10* shows the ROC charts of both models on the validation data.

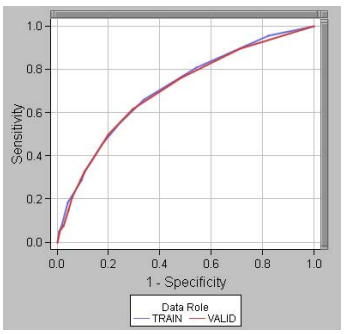

Figure 10: ROC chart.

According to this chart, the difference in quality of the two models is minimal. This is also confirmed by comparing fit statistics, such as KS, Gini and AUROC. The greater flexibility of a decision tree model has not shown to be important in this case. We therefore prefer the scorecard as our model of choice, as it is used in practice today and because the tabular format is particularly easy to understand and explain.

Reject inference

The application scoring models we have built so far, even though we have done everything correctly, still suffer from a fundamental bias. They have been built based on a population that is structurally different from the population to which they are supposed to be applied. All of the example applications in the development sample are applications that have been accepted by the old generic scorecard that has been in place during the past two years.

This is so because only for those accepted applications is it possible to evaluate their performance and to define a *good/bad* variable. However, the through-the-door population that is supposed to be scored is composed of all applicants, those that would have been accepted and those that would have been rejected by the old scorecard. Note that this is only a problem for application scoring, not for behavioral scoring.

As a partial remedy to this fundamental bias, it is common practice to go through a process of reject inference. The idea of this approach is to score the data that is retained of the rejected applications with the model that is built on the accepted applications. Next, *rejects* are classified as inferred *goods* or inferred *bads* and are added to the *accepts* data set that contains the actual *good* and *bad*. This augmented data set then serves as the input data set of a second modeling run. In the case of a scorecard model, this involves the readjustment of the classing and the recalculation of the regression coefficients.

Instead of using a hard cut-off score for classifying the *rejects*, one can instead also add the same *reject* twice, once as an inferred *good* and once as an inferred *bad*, but adjust the corresponding sampling weight by multiplying it with the predicted probability of being *good* or *bad* respectively. The rationale behind this approach is that a hard cut-off for reject inference would be an arbitrary decision that biases the augmented data set. Since the cut-off for eventually accepting or rejecting an application follows from cost-revenue trade-off considerations that are based on the final scorecard, choosing a cut-off based on a preliminary scorecard seems unfounded. *Figure 11* shows a modeling flow that includes a reject inference process.

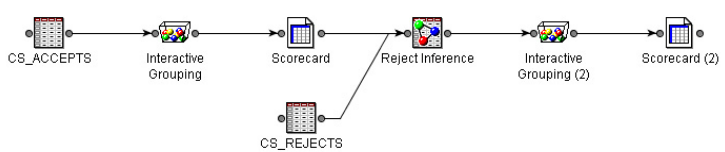

Figure 11: Process flow diagram – "rejects" data.

Summary

This paper has illustrated how SAS Enterprise Miner software is used to build credit scoring models. In the introduction, it discussed the benefits of performing credit scoring and the advantages of building credit scoring models in-house and with SAS Enterprise Miner. It went on to discuss the advantages and disadvantages of three important model types, the scorecard, the decision tree and the neural network. Finally, it presented a case study where an application scoring model is built with SAS Enterprise Miner, starting with reading the development sample, through classing and selecting characteristics, fitting a regression model, calculating score points, assessing scorecard quality (in comparison to a decision tree model built on the same sample) and going through a reject inference process to arrive at a model for scoring the new customer applicant population.

The study has been presented in the hope that the reader will come to appreciate the efficiency gains that SAS Enterprise Miner software brings to the modeling process. Efficiency is gained by automating the modeling process, but even more so by providing the analyst with a graphical user interface that structures, connects and documents the flow of activities that are carried out.

Thus, if changes or variations need to be introduced, the overall process is already defined, and one doesn't have to start from scratch. (For example, if a similar analysis needs to be carried out on different data, for a different purpose and by a new analyst, the process is still easy to apply.) Process flows enable the company to implement its traditional way of working but also to experiment with new approaches and to compare the results. The environment is flexible and open enough to enable the analyst to interact with the data and the models and, therefore, to bring in fully his or her domain expertise.

In summary, one should think of SAS Enterprise Miner more as a steering wheel than as an autopilot. Because SAS Enterprise Miner is only one part of the whole SAS system for information delivery, there will be no bottleneck when deploying individual models or when automating the whole score creation and application process in an enterprisewide solution.

References

1. SAS Institute Inc. (1996) *SAS' Rapid Warehousing Methodology: A SAS White Paper*, Cary, NC: SAS Institute Inc.

2. Inmon, W. H. (1993) *Building the Data Warehouse*, New York: John Wiley & Sons, Inc.

3. SAS Institute Inc. (1997) *From Data to Business Advantage: Data Mining, SEMMA Methodology, and the SAS System: A SAS White Paper*, Cary, NC: SAS Institute Inc.

4. SAS Institute Inc. (1998) *Finding the Solution to Data Mining: A Map of the Features and Components of SAS® Enterprise Miner™ Software: A SAS White Paper*, Cary, NC: SAS Institute Inc.

5. Dummy variable based scorecard development is also provided by SAS Enterprise Miner, but is not described here.

Recommended reading

Credit Scoring

Hand, D. J. and Henley, W.E. (1997) *Journal of the Royal Statistical Society* A, 160, Part 3, 523–541

Lewis, E. M. (1992) *An Introduction to Credit Scoring*, Fair, Isaac and Co., Inc., Library of Congress Catalog Number: 90-92258

L. C. Thomas, Jonathan Crook, David Edelman, Lyn Thomas (2002) *Credit Scoring & Its Applications* (Siam Monographs on Mathematical Modeling and Computation)

Lyn C. Thomas, David B. Edelman, Jonathan N. Crook, L. C. Thomas (2004) *Readings in Credit Scoring: Foundations, Developments, and Aims* (Oxford Finance S.)

Elizabeth Mays (2003) *Credit Scoring For Risk Managers: The Handbook For Lenders*

Naeem Siddiqi (2006) *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring* (SAS Institute Inc.)

Data mining

Adriaans, Pieter and Dolf Zantinge (1996) *Data Mining*, Harlow, England: Addison Wesley

Berry, Michael J. A. and Gordon Linoff (1997) *Data Mining Techniques*, New York: John Wiley & Sons, Inc.

SAS Institute Inc. (1997) *Business Intelligence Systems and Data Mining: A SAS White Paper*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1997) *From Data to Business Advantage: Data Mining, SEMMA Methodology, and the SAS System: A SAS White Paper*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1998) *Finding the Solution to Data Mining: A Map of the Features and Components of SAS® Enterprise Miner™ Software: A SAS White Paper*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1999) *Cross-Selling in the Financial Industry: Solving Business Problems Using SAS Enterprise Miner Software*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1999) *Data Mining and the Case for Sampling: Solving Business Problems Using SAS Enterprise Miner Software*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1999) *Building Consumer Credit Scoring Models with Enterprise Miner*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1999) *Using Data Mining for Fraud Detection: Solving Business Problems Using SAS Enterprise Miner Software*, Cary, NC: SAS Institute Inc

Weiss, Sholom M. and Nitin Indurkhya (1998) *Predictive Data Mining: A Practical Guide*, San Francisco, California: Morgan Kaufmann Publishers, Inc.

Data warehousing

Berson, Alex and Stephen J. Smith (Contributor) (1997) *Data Warehousing, Data Mining and OLAP*, New York: McGraw Hill

Inmon, W. H. (1993) *Building the Data Warehouse*, New York: John Wiley & Sons, Inc.

SAS Institute Inc. (1995) *Building a SAS Data Warehouse: A SAS White Paper*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1996) *SAS Institute's Rapid Warehousing Methodology: A SAS White Paper*, Cary, NC: SAS Institute Inc

Singh, Harry (1998) *Data Warehousing Concepts, Technologies, Implementations, and Management*, Upper Saddle River, New Jersey: Prentice-Hall, Inc

World Headquarters
and SAS Americas
SAS Campus Drive
Cary, NC 27513 USA
Tel: (1) 919 677 8000
Fax: (1) 919 677 4444
U.S. & Canada sales:
(1) 800 727 0025

SAS International
PO Box 10 53 40
Neuenheimer Landsr. 28-30
D-69043 Heidelberg, Germany
Tel: (49) 6221 4160
Fax: (49) 6221 474850

www.sas.com

100423US_367424.0106