# Table of Contents

- **Introduction** ................................................................. 1  
- **Building Credit Models In-House** .................................. 2  
- **Building Credit Models with SAS® Enterprise Miner™** .......... 2  
- **SAS® Enterprise Miner™ Process Flow Templates** ............... 3  
- **The Larger Credit Scoring Process** ................................. 3  
- **Choosing the Right Model** ............................................. 4  
  - Scorecards ................................................................. 4  
  - Decision Trees .......................................................... 5  
  - Neural Networks ......................................................... 5  
- **Case Study** .................................................................. 6  
  - Scenario ................................................................. 6  
  - SAS® Enterprise Miner™ Process Flow ............................... 6  
  - Development Sample ..................................................... 7  
  - Classing ................................................................. 8  
  - Logistic Regression ...................................................... 10  
  - Score-Points Scaling .................................................... 11  
  - Adverse Characteristics ............................................... 12  
  - Scorecard Assessment .................................................. 13  
  - Decision Tree Model .................................................... 15  
  - Model Comparison ...................................................... 16  
  - Reject Inference ........................................................ 18  
- **Summary** .................................................................... 19  
- **Recommended Reading** ............................................... 19
Introduction

Over the past 30 years of growing demand, stronger competition and advances in computer technology, the traditional methods of making credit decisions that relied mostly on human judgment have been replaced by methods that use statistical models. Today, statistical models are used not only for deciding whether to accept an applicant (application scoring), but also to predict the likelihood of defaults among customers who have already been accepted (behavioral scoring) and to predict 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. 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 to optimize the approval rate for credit applications. Application scores enable the organization to choose an optimal cutoff 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 selected 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 for determining custom 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 enable the organization 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 also are 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 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 access of outsiders 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™ is SAS® solution for data mining. It is used across many industries to answer a variety of business questions, and it has been extended with specific functionality for credit scoring that is described in more detail in the case study section below. 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 so that analysts can interact with data and models to fully use their domain expertise, i.e., use the software as a steering wheel and not as an autopilot. 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.

Credit Scoring for SAS Enterprise Miner adds these specific nodes to the SAS Enterprise Miner toolset support scorecard development:

- The Interactive Grouping node facilitates the classing of characteristics into attributes, which is essential to building a credit scorecard. While performing an initial automatic grouping that can provide optimal splits, this node enables the user to regroup the variables through an interactive interface. Additionally, it has the capability of variable screening/selection.
• The Scorecard node follows the Interactive Grouping node and uses the grouped variables as inputs in a logistic regression model. It also scales the regression parameters to compute score points and the resulting scorecard. Finally, the Scorecard node performs score and characteristic (variable) analysis that helps in understanding the scorecard and aids in crafting score-based strategies.

• The Reject Inference node offers three standard industry-accepted methods for inferring the performance of the reject applicant data by using the model built on accepted applicants.

• The Credit Exchange node enables the use of scorecards in the SAS credit risk solution.

**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 below 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.

Collecting performance data occurs at the beginning and 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.
The power of SAS software 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. This is a subject-oriented, integrated, time-variant and nonvolatile repository of information 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 such question, a list of possible answers (called attributes). For example, one such 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 cutoff 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 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. Also, the scorecard is 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 a decision that is made on the basis of a scorecard to the applicant. It is possible to disclose groups of characteristics where the applicant has potential to improve the score and to do so in broad enough terms not to risk the manipulation of 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 of employment. Then, credit might be granted only to those who have exceeded 24 months of employment, because only in that subsegment 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 also are easy to understand, perhaps even more so than a decision rule that is based on a total score 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 there is such a thing as a decision rule that is too easy to understand and thereby invites fraud. 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 then must 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 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 be created later as required.

SAS® Enterprise Miner™ Process Flow
SAS Enterprise Miner software is used for building the scorecard. It enables the analyst to access a comprehensive collection of analytical tools through a graphical user interface, and provides a workspace onto which nodes (tool icons) are dropped from a tools palette. Nodes are then connected to create 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.

Figure 1 shows the process flow for modeling on the “accepts” data and scoring the “rejects” data. All components of the flow are discussed in more detail in the sections below. The flow begins with reading in the development sample (CS_ACCEPTS). 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 – “CSAccepts” data.
Development Sample

The development sample (input data set) is a balanced sample consisting of 1,500 “good” and 1,500 “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. (Note: The Scorecard node in Credit Scoring for SAS® Enterprise Miner™ can process indeterminate target values and include them in all Gains table calculations). 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 accepted 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 is assigned to a bad application, using the frequency variable in the data set. Thereafter, all nodes in the process flow diagram treat the sample as if it consisted of 1,500 good applications and 1,500 bad applications. The bad rate is 3.23 percent in our development sample. A Data Partition node, run with the default partition of 40 percent training, 30 percent validation and 30 percent test, then splits a 30 percent validation data set away from the development sample. Models will be compared later based on this validation data set.

Figure 2: Variable list – a development sample.
Classing

Classing is the process of automatically and/or interactively binning and grouping interval, nominal or ordinal input variables 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 and 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 the section on 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 correspond to high risk; high positive values correspond to low risk. (See Equation 1 and Figure 3.)

$$\ln\left(\frac{\text{Distr Good}_i}{\text{Distr Bad}_i}\right)$$ for each group $i$ of a characteristic.

Equation 1: Weight of Evidence.

Because an attribute’s number of points in the scorecard is proportional to its Weight of Evidence (see the section on 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 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. (See Equation 2.)

$$\sum_{i=1}^{L} (\text{Distr Good}_i - \text{Distr Bad}_i) \cdot \ln\left(\frac{\text{Distr Good}_i}{\text{Distr Bad}_i}\right)$$, where $L$ is the number of attributes (levels) of the characteristic.

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 overpredicting, 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 table on the bottom lists the Weights of Evidence and event rates for each group. When the variable “Age” is selected, a splitting algorithm automatically suggests a grouping, which then can be modified manually in a variety of ways. Whenever a change is made, the statistics that describe the current grouping are updated. Those statistics include the distribution of the attributes, the Weight of Evidence and bad rate per attribute.

You define customized interactive groupings in the coarse detail level. 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 moved freely 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 for the scorecard to appear plausible. Some analysts will always include only those characteristics where a sensible re-grouping can achieve this. Others may consider a smooth variation sufficiently plausible and would include a nonmonotone 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 at Job,” “EC Card Holder,” “Customer Status,” “Income,” “Credit Cards” 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 nonmonotone characteristic.

![Figure 3: Interactive grouping results — classing for the interval variable “Age.”](image-url)
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.

<table>
<thead>
<tr>
<th>Grouped: Age</th>
<th>Weight of Evidence: Age</th>
<th>Grouped: Income</th>
<th>Weight of Evidence: Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>0.3726752853</td>
<td>1.0</td>
<td>0.5194093675</td>
</tr>
<tr>
<td>4.0</td>
<td>0.3726752853</td>
<td>2.0</td>
<td>-0.693147181</td>
</tr>
<tr>
<td>3.0</td>
<td>0.0264916154</td>
<td>1.0</td>
<td>0.5194093675</td>
</tr>
<tr>
<td>4.0</td>
<td>0.3726752853</td>
<td>3.0</td>
<td>-0.350925966</td>
</tr>
<tr>
<td>4.0</td>
<td>0.3726752853</td>
<td>1.0</td>
<td>0.5194093675</td>
</tr>
<tr>
<td>1.0</td>
<td>-1077558879</td>
<td>3.0</td>
<td>-0.350925966</td>
</tr>
<tr>
<td>4.0</td>
<td>0.3726752853</td>
<td>1.0</td>
<td>0.51940936675</td>
</tr>
<tr>
<td>5.0</td>
<td>1.0414538748</td>
<td>2.0</td>
<td>-0.693147181</td>
</tr>
<tr>
<td>3.0</td>
<td>0.0264916154</td>
<td>3.0</td>
<td>-0.350925966</td>
</tr>
<tr>
<td>4.0</td>
<td>0.3726752853</td>
<td>4.0</td>
<td>0.3722394605</td>
</tr>
</tbody>
</table>

Table 1: Weight of Evidence coding example.

A variety of further selection methods (forward, backward and stepwise) can be used in the Scorecard node to eliminate redundant characteristics. In our case, we use a simple regression. Figure 4 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.
Score-Points Scaling

For each attribute, its Weight of Evidence and the regression coefficient of its characteristic can 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 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 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 Properties panel of the Scorecard node (see Figure 5), where it can be easily parameterized. The resulting scorecard is output as a table and is shown in Figure 6. 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.

\[
Score = \sum_{i=1}^{n} \left( -\text{woe}_i \ast \beta_i + \frac{\text{a}}{n} \ast \text{factor} + \frac{\text{offset}}{n} \right)
\]

*Equation 3: Scaling.*

\[
600 = \log(50) \ast \text{factor} + \text{offset}
\]

\[
620 = \log(100) \ast \text{factor} + \text{offset}
\]

*Equation 4: Scaling example.*
Scaling Options

- Intercept Based Scorecard: No
- Reverse Scorecard: No
- Odds: 50.0
- Scorecard Points: 600.0
- Points to Double Odds: 20.0

Figure 5: Scorecard scaling options.

![Scorecard](image)

Figure 6: Scorecard.

**Adverse Characteristics**

Adverse codes can be calculated based on either Neutral Score or Weighted Average. A report of these adverse codes can be generated showing the top reasons for denial for each applicant and the number of times each reason is cited across the data set.

<table>
<thead>
<tr>
<th>Input Variable</th>
<th>Adverse Reason 1 Count</th>
<th>Adverse Reason 2 Count</th>
<th>Adverse Reason 3 Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGE</td>
<td>356</td>
<td>96</td>
<td>91</td>
</tr>
<tr>
<td>CARDS</td>
<td>402</td>
<td>126</td>
<td>23</td>
</tr>
<tr>
<td>EC_CARD</td>
<td>178</td>
<td>45</td>
<td>38</td>
</tr>
<tr>
<td>INCOME</td>
<td>68</td>
<td>223</td>
<td>87</td>
</tr>
<tr>
<td>PERS_H</td>
<td>4</td>
<td>78</td>
<td>146</td>
</tr>
<tr>
<td>STATUS</td>
<td>4</td>
<td>119</td>
<td>260</td>
</tr>
<tr>
<td>TMJOB1</td>
<td>188</td>
<td>157</td>
<td>103</td>
</tr>
</tbody>
</table>

Figure 7: Adverse codes overall.
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 then are used to evaluate the discriminatory power of the scorecard. These charts analyze the scorecard’s ability to separate the good cases from the bad cases by their scores. Measures include the Kolmogorov-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.

Figure 9: The KS chart shows the difference between the cumulative distribution of the “goods” and the “bads.” The maximum value is the KS statistic.
Figure 10: 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 cutoff values. The more the chart bends to the top left, the better.

Figure 11: The Captured Bad chart shows the proportion of all “bads” concentrated in each decile.

In application scoring, trade-off charts are used that show how the approval rate and the bad rate in the accepted applications depend on the cutoff score. Good scorecards enable the choice of a cutoff that corresponds to a relatively high approval rate and a relatively low bad rate.
Figure 12: The Trade-Off chart displays approval rate and bad rate against cutoff score for the current model. Horizontal lines show values of the previously used model and cutoff 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. 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.

Figure 13: Actual against predicted odds.

**Decision Tree Model**

After going 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 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 subsegments (leaves) are merged again (pruned) in order to optimize the tree.

Figure 14: Decision tree results.

Figure 14 shows the resulting tree structure for our case. The most important variable is age, with applicants younger than 27.5 having an elevated credit risk. Furthermore, among the young, those with incomes less than $4,500 have the highest risk of all applicants.

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 15 shows the ROC charts of both models on the validation data. 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. Therefore, we 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.
Figure 15: Model comparison results (ROC and lift charts).
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 who would have been accepted and those who 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 from 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 cutoff 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 cutoff for reject inference would be an arbitrary decision that biases the augmented data set. Since the cutoff for eventually accepting or rejecting an application follows from cost-revenue trade-off considerations that are based on the final scorecard, choosing a cutoff based on a preliminary scorecard seems unfounded.

Figure 16 shows a modeling flow that includes a reject inference process.

*Figure 16: Process flow diagram – “rejects” data.*
Summary

This paper has illustrated how Credit Scoring for SAS Enterprise Miner software is used to build credit scoring models for the retail credit industry. In the introduction, it discussed the benefits of performing credit scoring and the advantages of building credit scoring models in-house using 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, beginning 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 readers will come to appreciate the efficiency gains that SAS Enterprise Miner software brings to the modeling process. Efficiency is gained by completely 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. If changes or variations need to be introduced, the overall process is already defined and doesn’t have to be started 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 organization to implement its traditional way of working but also to experiment with new approaches and compare the results. The environment is flexible and open enough so the analyst can interact with the data and models and bring in his or her domain expertise.

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

Recommended Reading


About SAS

SAS is the leader in business analytics software and services, and the largest independent vendor in the business intelligence market. Through innovative solutions, SAS helps customers at more than 60,000 sites improve performance and deliver value by making better decisions faster. Since 1976 SAS has been giving customers around the world THE POWER TO KNOW®. For more information on SAS® Business Analytics software and services, visit sas.com.