SESUG Paper 251-2019

Validating a Probability of Default model using SAS®Enterprise Miner Scorecard

George Rezek, TCF Bank

ABSTRACT

Despite the prevalence of Enterprise Miner (EM) with Scorecard, Probability of Default (PD) models at TCF Bank are more often developed and validated using products outside of EM. While there is no shortage of demos pointing out features of using EM, this presentation focuses on stumbling blocks that are obvious to those familiar with the product, but often cause potential users to give up on EM citing time considerations. The Interactive Grouping Node is often mentioned as reason enough to use EM, because of the ease it brings to the binning process. Relevant statistical output, graphs and data are automatically generated—some of the controls for these are easier to find than others; code nodes allow interaction with base SAS and entire EM diagrams can be converted to SAS code and run in simulations.

INTRODUCTION

While there are many articles on the internet concerning many aspects of the use and results of Enterprise Miner / Scorecard, I had trouble finding a simple overall guide to setting up a model. This paper attempts to help the user get started without getting into the finer points and implications of parameters and settings using a simple case.

This example is a PD model validated without a validation dataset available. How to convert the process diagram to SAS code that can be run in Base SAS is also covered.

ACCESSING AND DEFINING DATA

The EM's first time user, or even intermittent users of EM, can be at a loss as to how to set up a simple diagram with the necessary parameters correctly specified. While it is possible to import data from .csv and Excel files, I simplify by assuming the input data to be in a SAS dataset already. To define the library to the EM project (*Figure 1*),

- Highlight the project name in the top panel at the left.
- Click on the three dots to the right of "Project Start Code" in the properties section
- Enter the libname statement in the panel that opens to the right and close the panel.

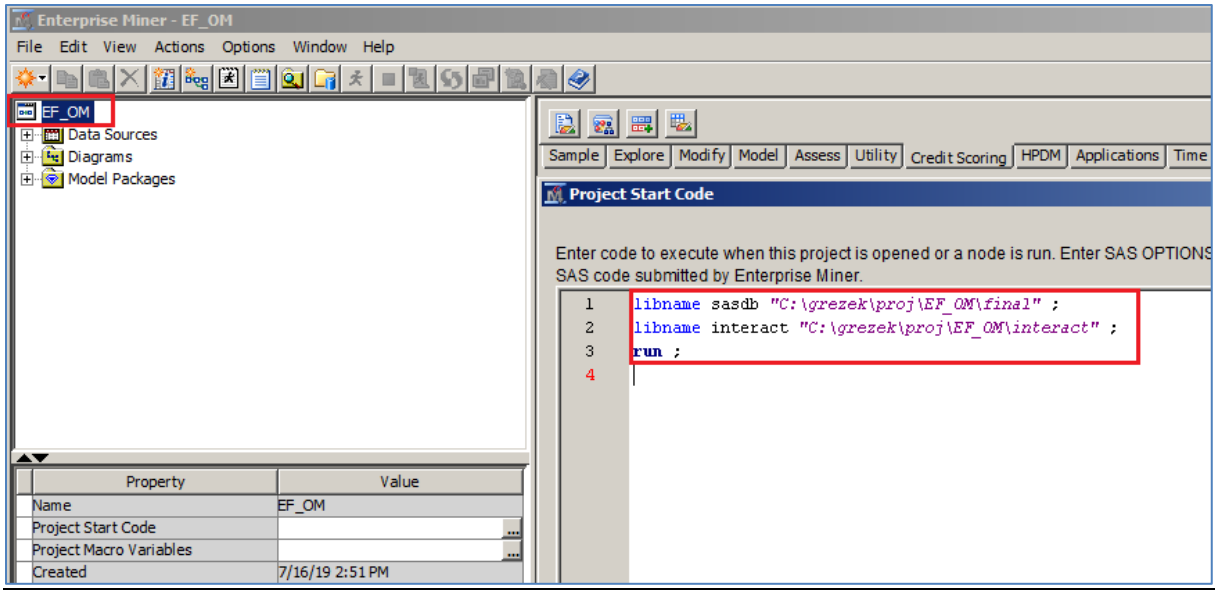


Figure 1. Assigning libraries to the project

Note that I've also added a library, "interact," to store the interactive grouping library (I wanted to keep the datasets of binnings separate). These libraries will be available to all diagrams within the project.

To add a dataset to project (*Figure 2*):

- Highlight "Data Sources" just below "EF_OM" Click on "Create data Source"
- Click "next" on SAS Table
- Browse to your SAS library created in Project Start Code (sasdb)
- Select the dataset you want to add, "final_test"

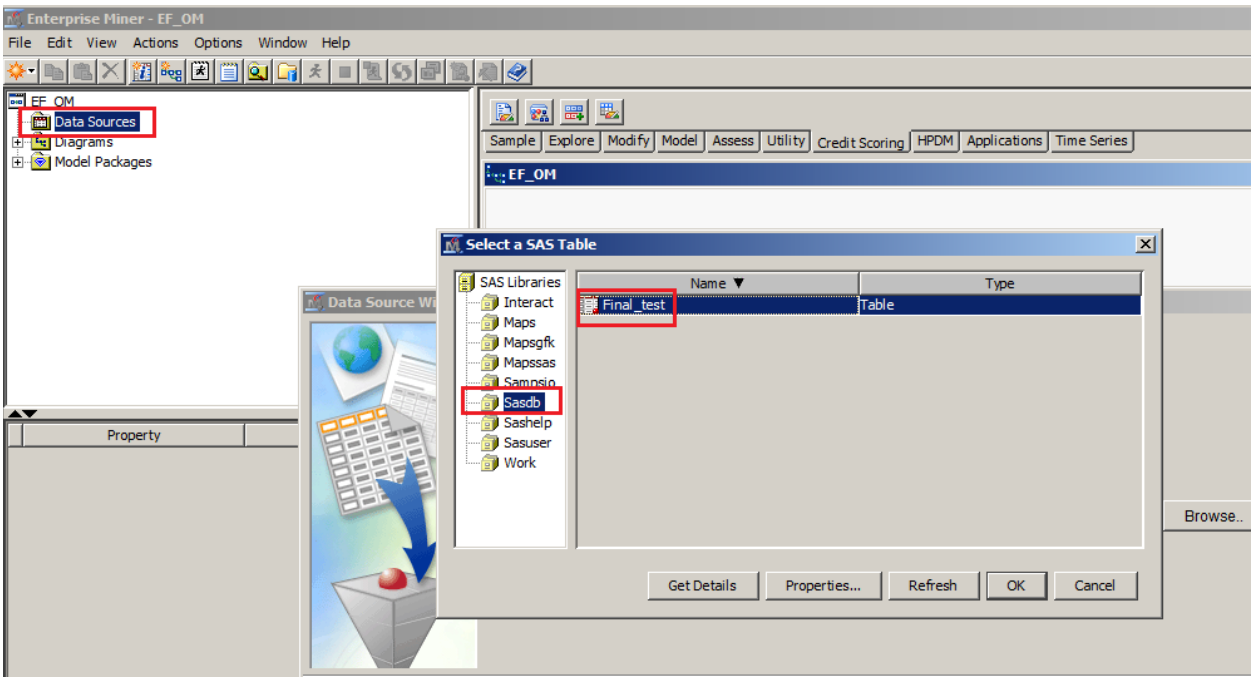


Figure 2. Adding a dataset to the project

Selecting "OK," "Next," "Next," "Next," brings us to the metadata display of the dataset (*Figure 3*). All data will typically appear with the role of "Input;" we need to assign the variables we are not going to include in the model to "Rejected." Depending on the number of variables in the dataset, it may be easier to highlight all the variables, changing the roles to "Rejected," then change the variables used in the model to "Input." We also have to change the role of the target variable, "new_def," to "Target" (and in this case to level "Binary.")

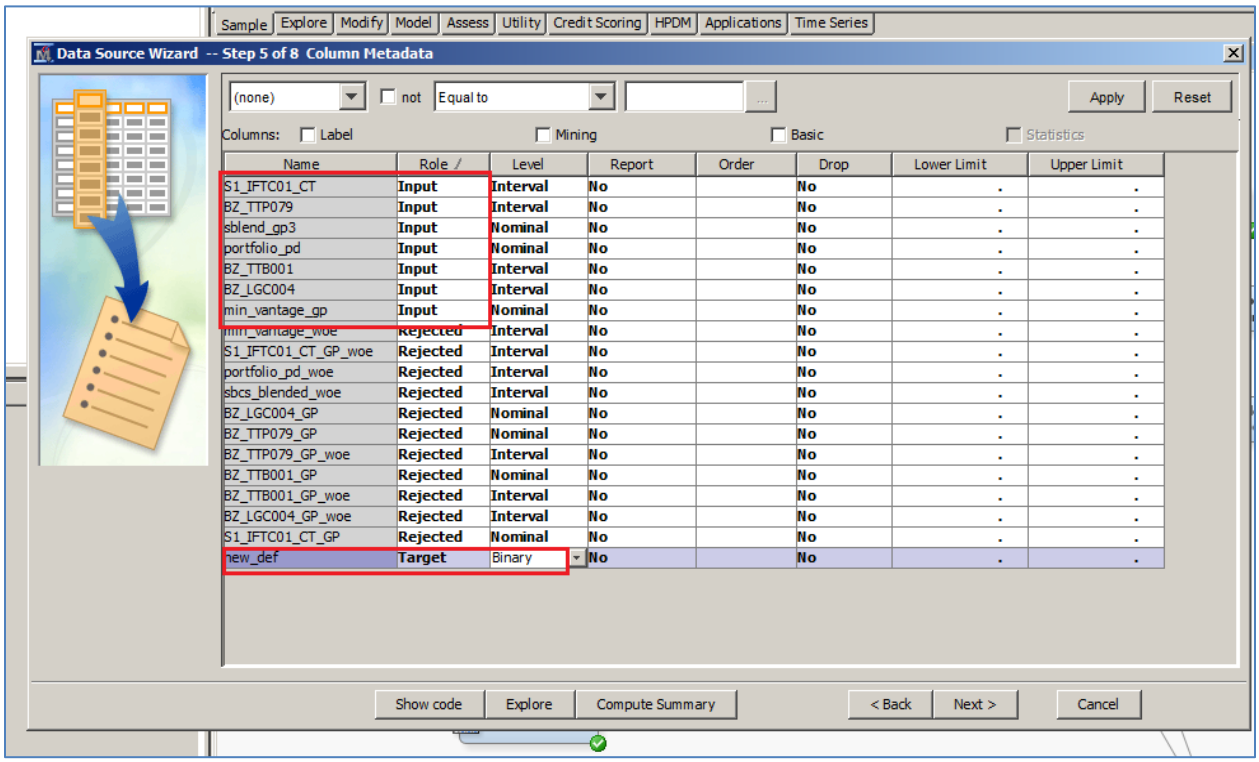


Figure 3. Defining input variables

Then "Next," "Next," and verify that this data will have the role of "Raw," that is for building the model (*Figure 4*).

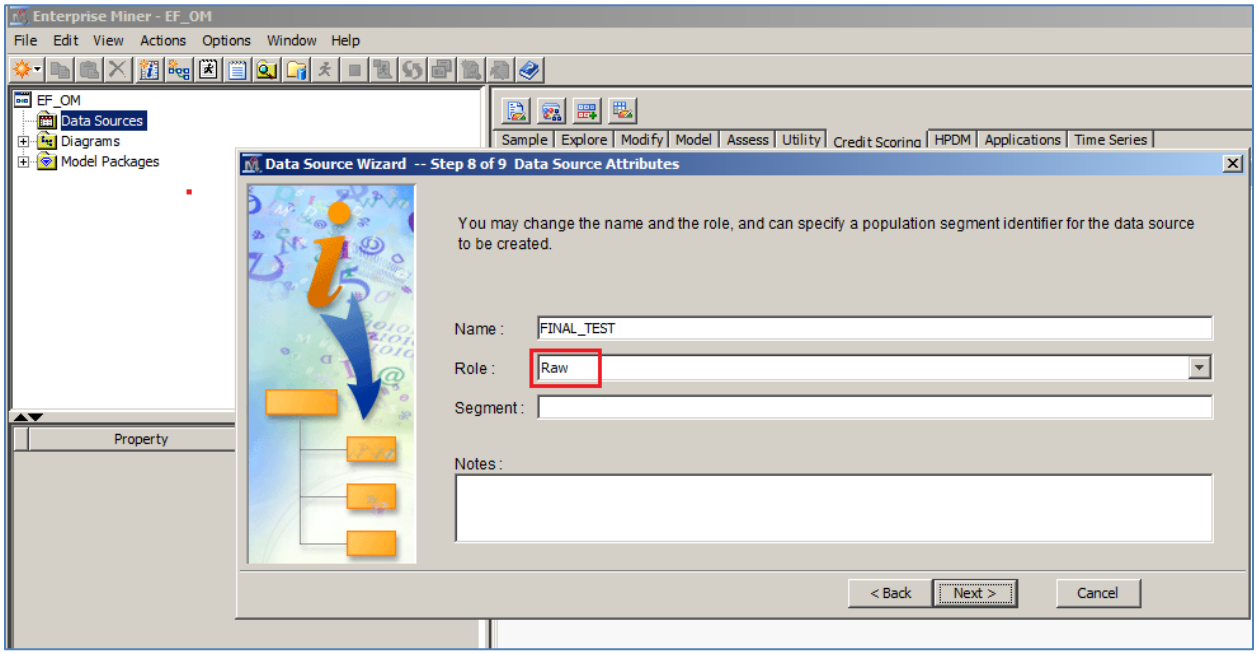


Figure 4. Assigning role of input dataset

Then "Next," "Finish."

The added dataset appears under Data Sources with the corresponding Property table appearing in the left panel (*Figure 5*).

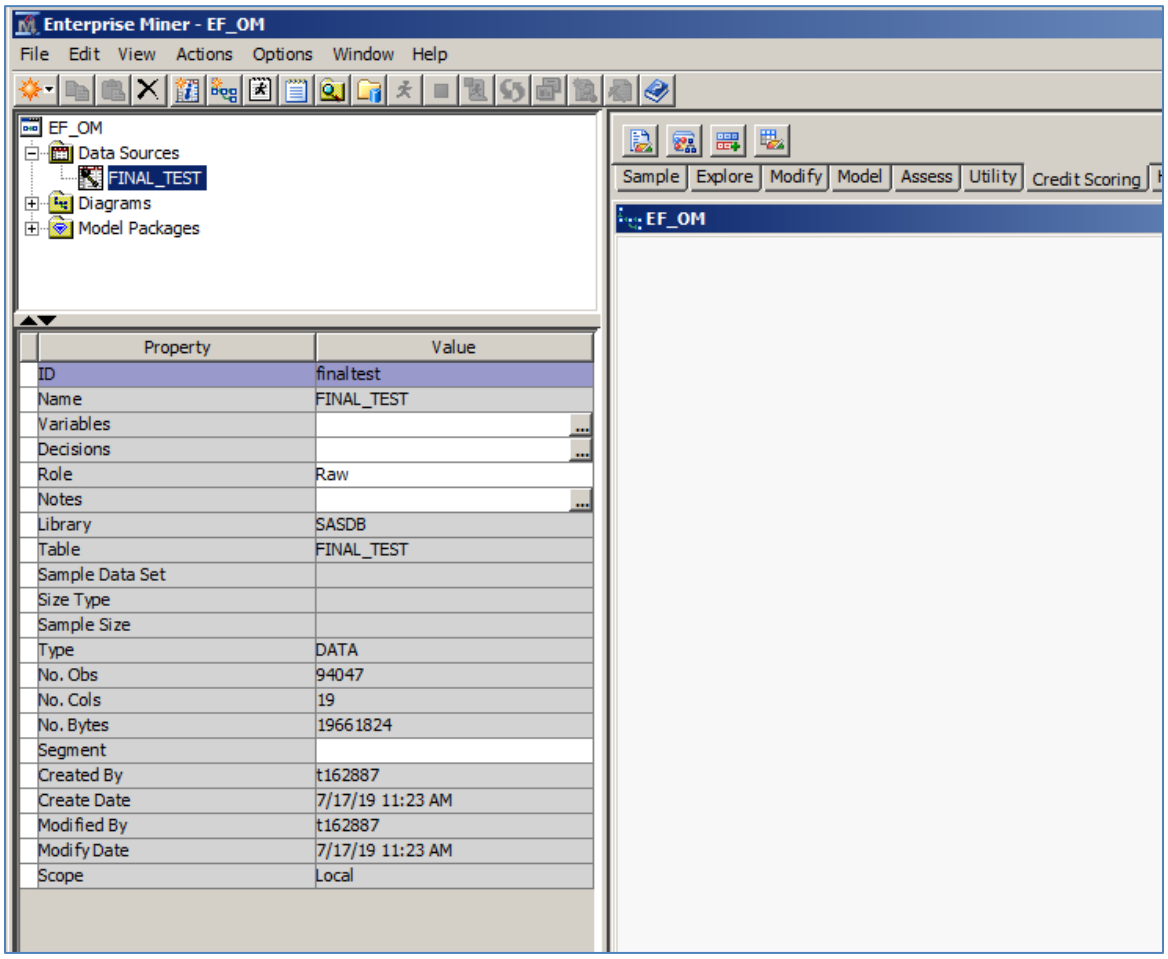


Figure 5. Properities of available dataset

To add a diagram to a project: Right click on "Diagrams" (*Figure 6*)

- Select "Create New Diagram"
- Enter new diagram name
- Click "OK"

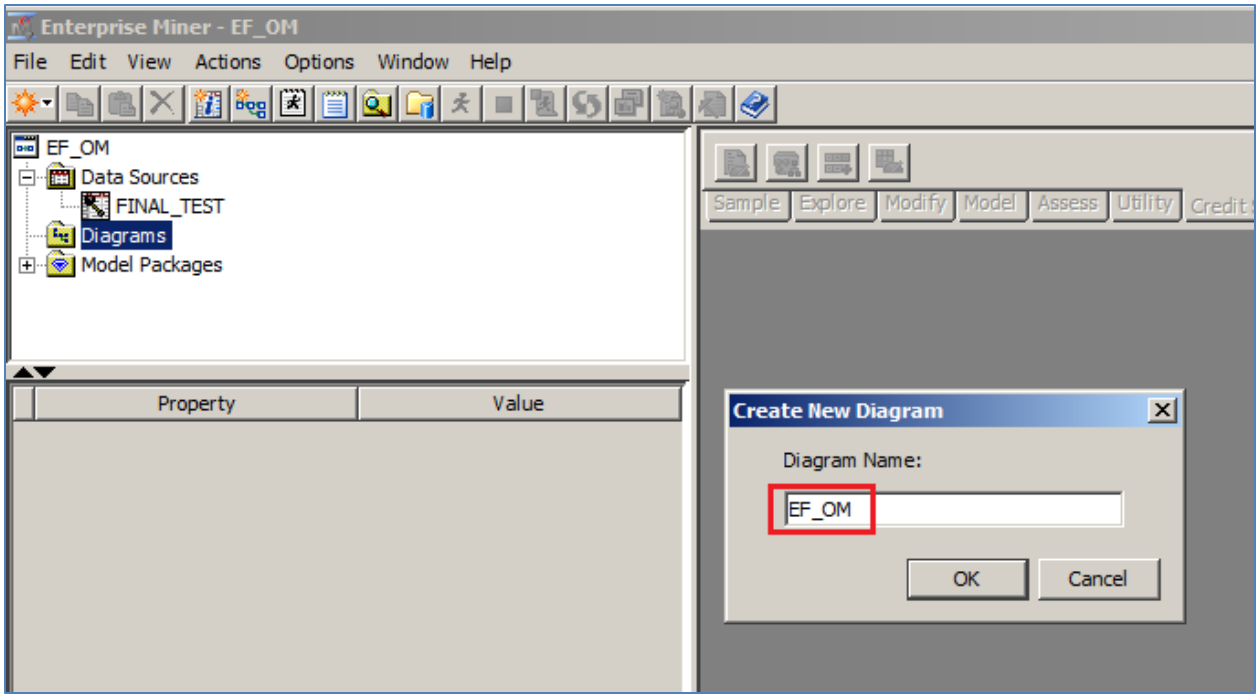


Figure 6. Assigning name to new diagram

Left click on the dataset, "Final_Test" in Data Sources and drag it to the diagram

THE BINNING PROCESS

In the diagram, EF_OM (*Figure 7*), the model process uses nodes in the "Credit Scoring" Tab. Hovering over the second item, a description of "Interactive Grouping" appears (elsewhere in EM under the "Modify" tab the node "Interactive Binning" appears, which is not the one we want). We drag "Interactive Grouping" to the diagram. We connect the dataset with the Interactive Grouping Node.

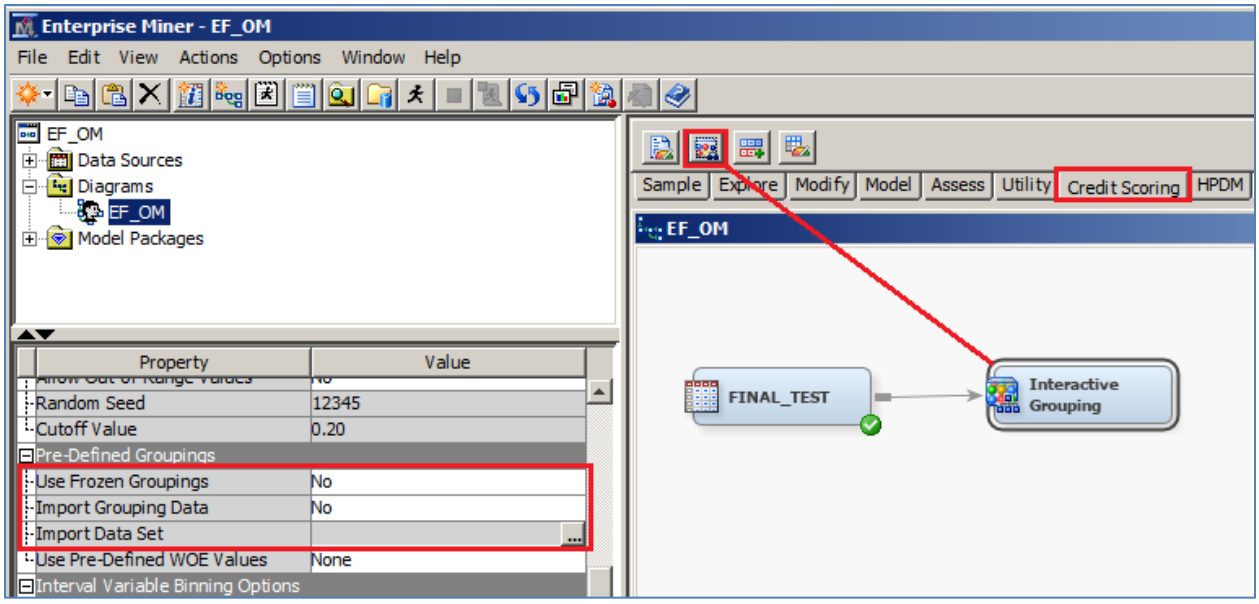


Figure 7. Adding Interactive Grouping Node to diagram

Selecting the Interactive Grouping node and scrolling down the properties window on the left we see quite a few parameters that can be set. Unless we have been given a SAS data set with the bins already defined, to replicate the developer's model, for our initial run we use the defaults (Use Frozen Groupings = "No" and Import Grouping Data = "No" as in *Figure 7*) by right-clicking on Interactive Grouping and selecting "run."

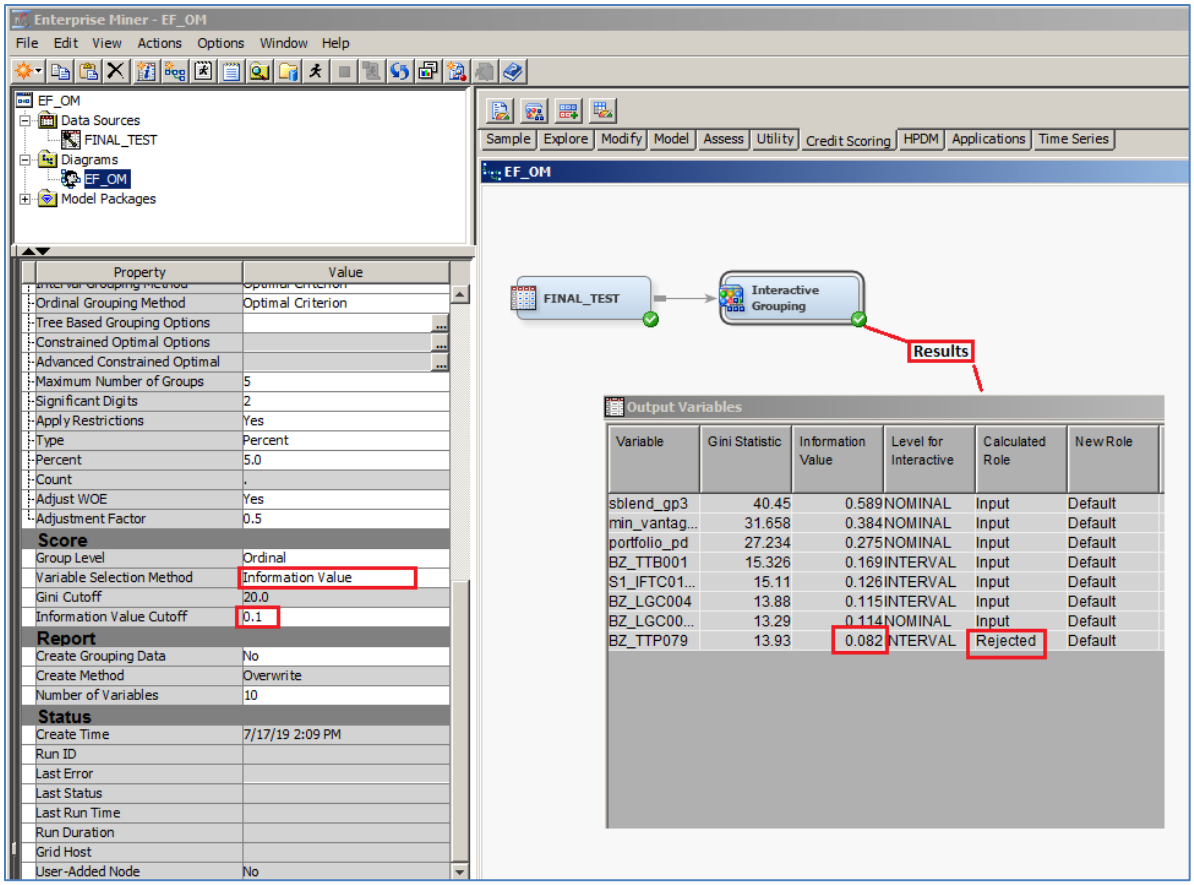


Figure 8. Including variables to the inputs

The green checks indicate that the nodes (and proceeding nodes) ran successfully (*Figure 8*). Clicking on "Results" in the window that opens up upon completion we notice in the Output Variables table that the variable, BZTTP079, was "Rejected." To control the acceptance level of a variable, we need to scroll down the "interactive Groupings" Property Panel until we see "Information Value Cutoff" (not to be confused with the "Cutoff Value" in the Internal Target Options) set to 0.1 by default. We can lower this value to, say, .02 which allows this variable to be included in the model.

After the Interactive Grouping has completed, we can see the details by selecting the Interactive Grouping node and left-clicking on the '…' next to the "Interactive Grouping" as we see in *Figure 9.* This dataset contains the binning information.

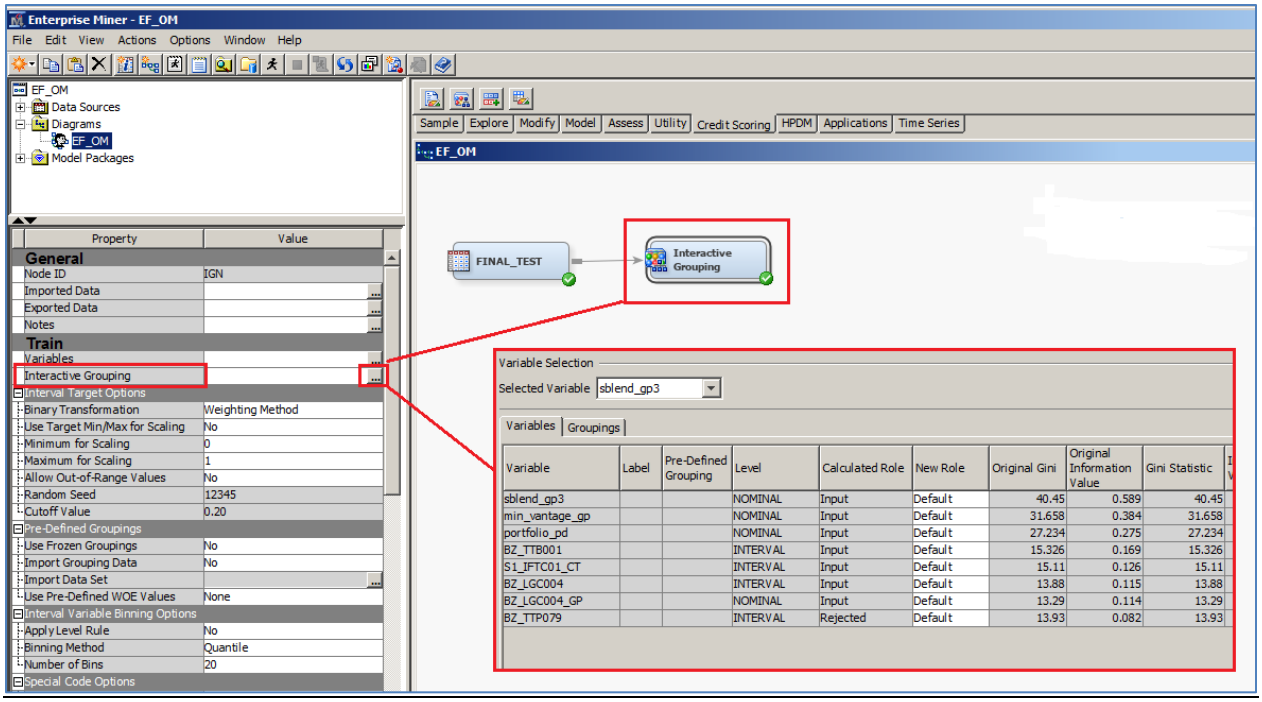


Figure 9. Input variable statistics

Selecting the "Groupings" tab and selecting a variable in the drop down list in *Figure 9*, we see that variable's bins (*Figure 10*). From this screen we can split the bin (selecting different cutoff values), combine groupings, reassign groups and observe the adjusted values in the columns Cutoff, Event Count, Non-Event Count, Total, Event Rate and WOE.

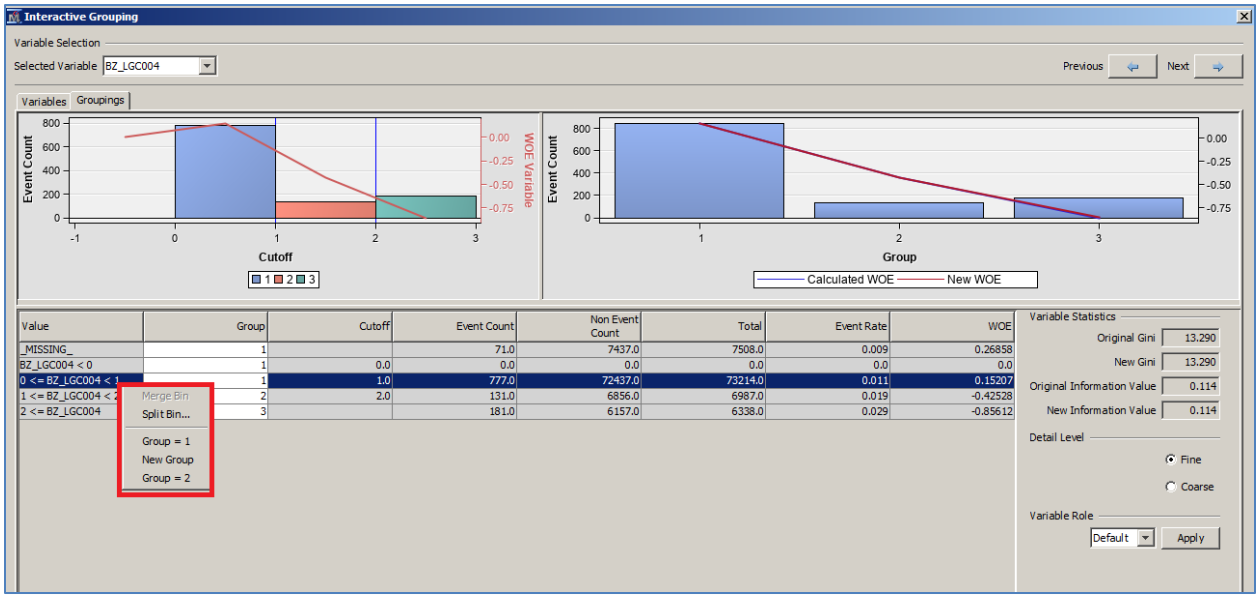


Figure 10. Variable characteristic groupings

The binning cutoff values are generated when "Create Grouping Data" in the section "Report" in the lower part of the Interactive Grouping panel is set to "Yes." A dataset, "ign_exportgroup" is found in the "EMWSx" directory under the "Workspaces" folder within the project folder.

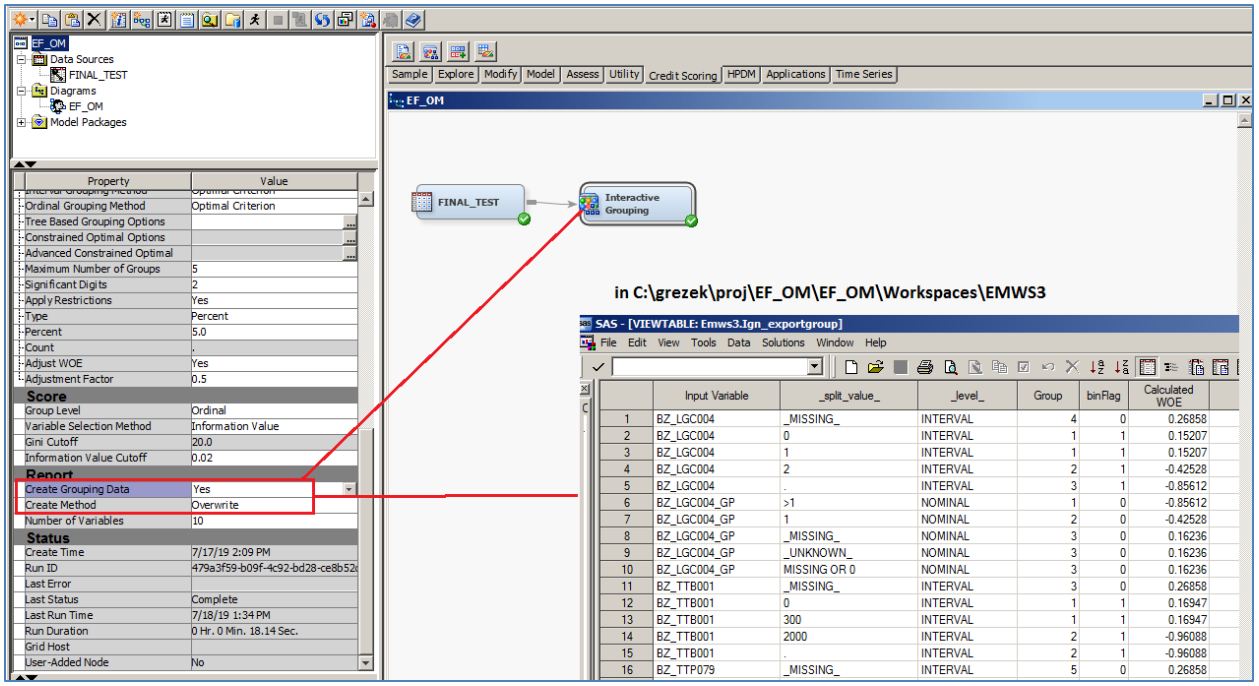


Figure 11. The Ign_exportgroup dataset of variable binnings

To save the bin definitions of the model, add a Code Node to the diagram connecting to the left side of the IGN node as we see in *Figure 12*. Save the IGN_EXPORTGROUP created by the diagram to the local library that is outside of the automatically created dataset. As inattention to panel values can result in the rerunning of the diagram replacing the cutoffs with system defaults in the Interactive Grouping, you may want execute this SAS code node periodically, rename the output dataset, to insure that you have a copy of the binnings.

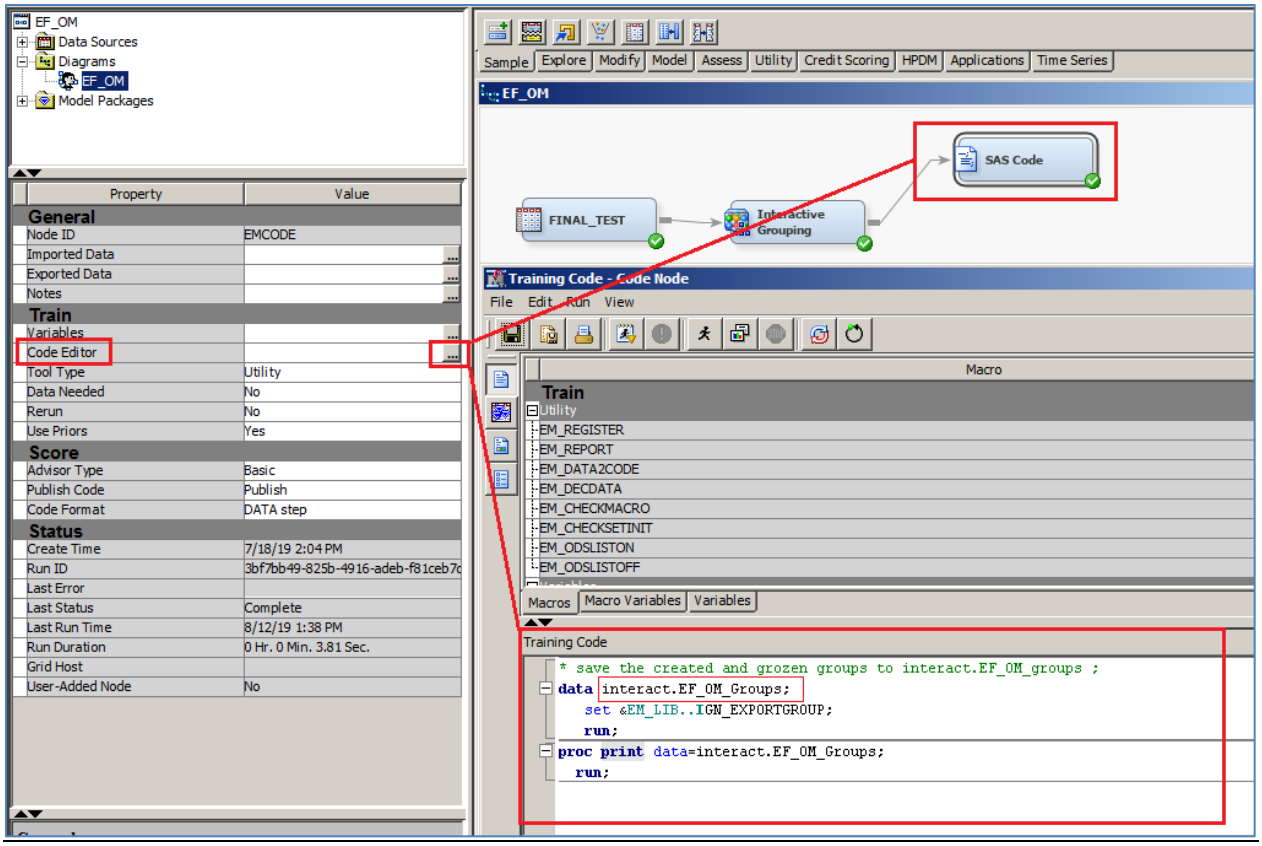
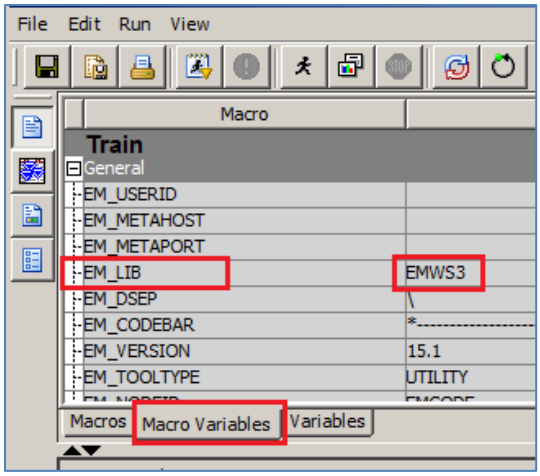


Figure 12. Creating dataset of variable bins from

In the SAS Code Node we have:

```
title 'save the created and frozen groups to interact.EF_OM_Groups';
data interact.EF_OM_Groups;
   set &EM_LIB..IGN_EXPORTGROUP;
run;
proc print data=interact. EF_OM_Groups';
run;
```



Click on the "Macro Variables" tab in the Code Node window and the resolution of the macro variable "EM_LIB" appears in the adjacent column.

Figure 13. Macro variables revealed

EMWS3 points to the subdirectory "C:\grezek\proj\EF_OM\EF_OM\Workspaces\EMWS3", that is automatically created by EM; ign_exportgroup is the SAS dataset that contains the saved bins.

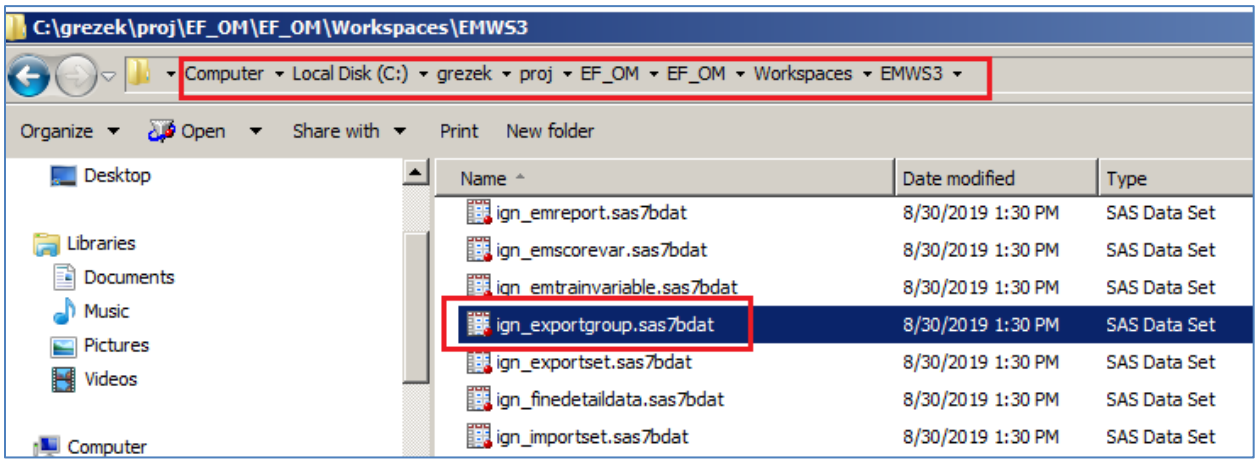


Figure 14. Dataset of bins created by the binning process

To read the saved binnings back into the process we can return to the Properties Tables of Interactive Grouping node, specify "No" for Create Grouping Data, and set "Use Frozen Data" to "No" and Import Data Set to "Yes", specifying the dataset, "INTERACT.EF_OM_GROUPS."

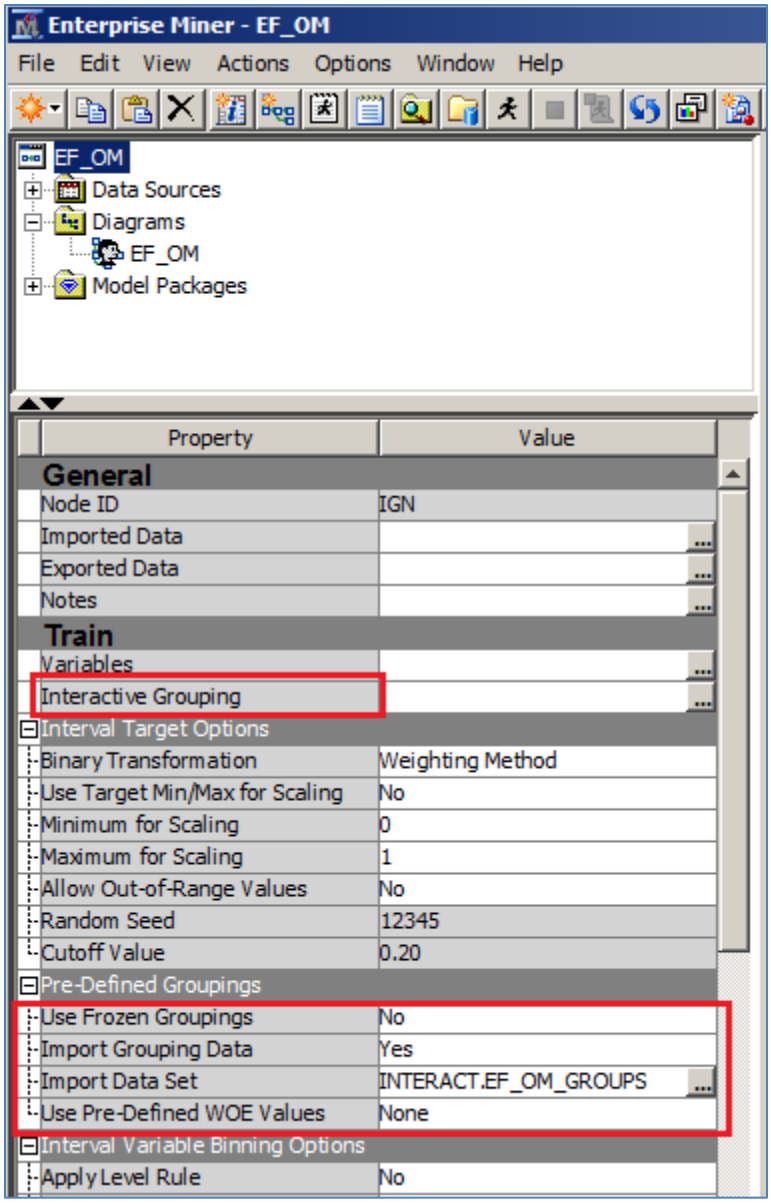


Figure 15 IGN properties to read binnings from SAS dataset

To read the saved binnings back into the process we can return to the Properties Tables of Interactive Grouping node, we can specify “No” for Create Grouping Data, and set “Use Frozen Data” to “No” and Import Data Set to “Yes”, specifying the dataset, “INTERACT.EF_OM_GROUPS.”

ROC Curve

Where sensitivity is the proportion of true positive responders (event = 1) that have a positive model result and specificity is the proportion of true negative responders (event =0) that have a negative model result. In this graph, *Figure 17*, the sensitivity is cumulative. The random line would be the result of model that does not differentiate between events 1 and 0. Intuitively a point below the random line would be in the area where the model is predicting an event = 1 where the true response was 0. The ROC or Area under the curve (AUR) is the area under the blue line.

ROC or area under the curve (AUR) is one measure of how well the model is in predicting defaults. One rule of thumb is:

ROC	Grade
90.0%	A
80.0%	B
70.0%	C
60.0%	D

Table 16. Grade equivalents of ROC percentages

KS is the maximum vertical distance between random line and the ROC curve. Gini can be calculated from ROC:

$$Gini = 2 \times ROC - 1.$$

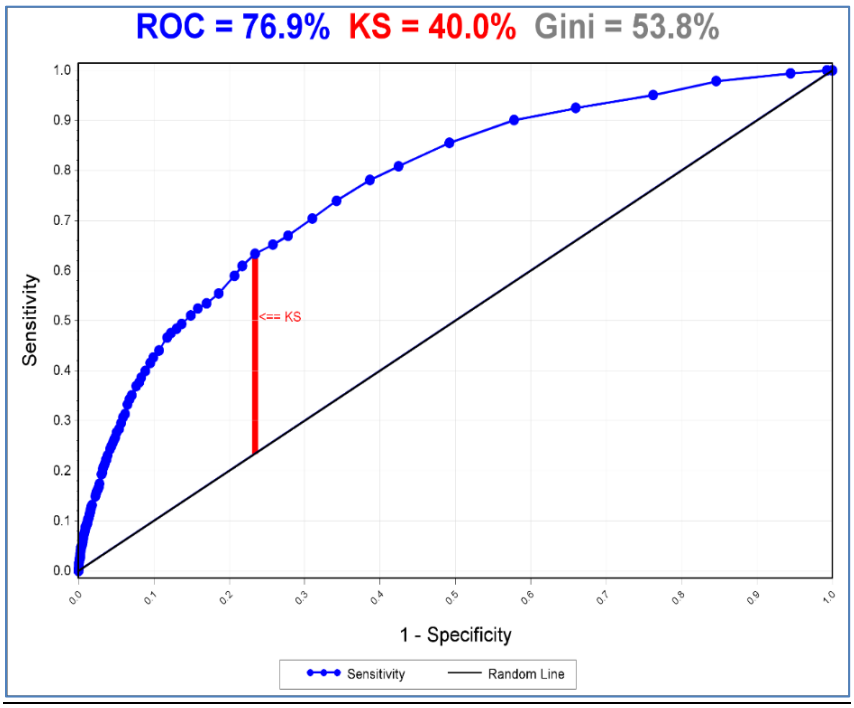


Figure 17. ROC Curve

SCORECARD REPLICATION

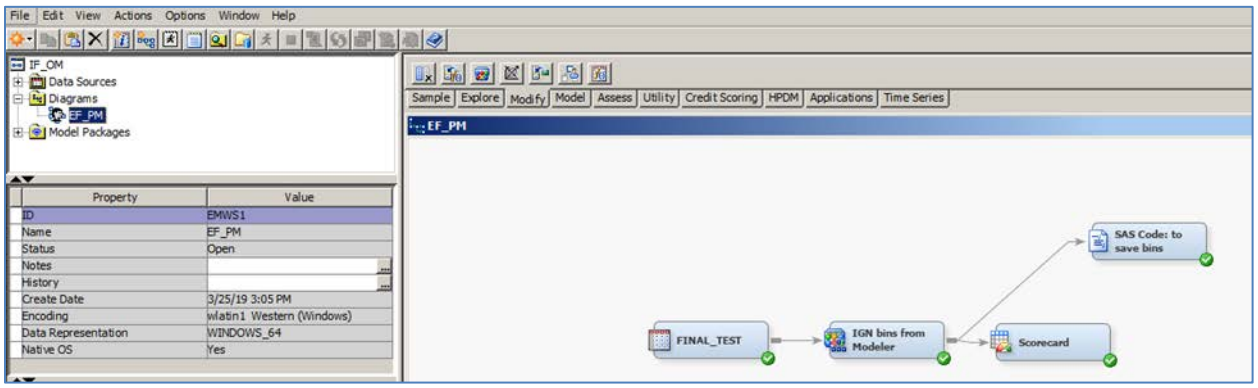


Figure 18. Enterprise Miner node to create SAS code for binning and scoring

We replicated the model in SAS's EM with the Scorecard module. We see in *Figure 18* that the nodes that are connected to build the model: the data input, "Final_test;" the Interactive Grouping Node, "IGN bins from Modeler;" the Scorecard Node; and the code node, "SAS Codes to save bins." The bins developed by the modeler were adopted in the Interactive Binning Node. By default, the Interactive Grouping Node re-estimates the bins optimizing by specified criteria, but here we have frozen the bins. The results are the same as those stated in the model documentation.

KS	0.400482
ROC	0.769132
Gini	0.538263

Table 19. Model statistics

FIVE-FOLD CROSS VALIDATION

We split the data into five parts making sure that each part has sufficient bads, defined as "new_def" in this input dataset. The code is further split into training and validation datasets.

Code to split up data into five parts:

```
libname sasdb "C:\grezek\proj\EF_OM\final" ;
proc sort data=sasdb.final_test out=sorted;
  by new_def ;
  run;
data make5;
   set sorted;
   fold=1+FLOOR(5*RAND('UNIFORM'));
   run;
title2 'OPTIONAL: check the percentages';
title3 'MAKE5';
proc freq data=make5;
   tables new_def / norow nocum;
   run;
proc freq data=make5;
   tables fold*new_def /  nocol nopercent nocum;
```

```
   run;
options mprint symbolgen macrogen ;
%macro fold_over ;
  %do i = 1 %to 5 ;
    data sasdb.mytrain_&i. sasdb.myvalidate_&i.;
      set make5;
      if fold=&i then output sasdb.myvalidate_&i.;
  else output sasdb.mytrain_&i;
  run;
    title2 'OPTIONAL: check the percentages';
    title3 "MYTRAIN I:  &i.";
  proc freq data=sasdb.mytrain_&i.;
    tables new_def / norow nocol;
    run;
    title3 "MYVALIDATE I:  &i.";
  proc freq data=sasdb.myvalidate_&i ;
    tables new_def / norow nocol;
    run;
  %end ;
run ;
%Mend ;
run ;
%fold_over ;
run ;
```

Table of fold by new_def

fold	new_def 0	new_def 1	Total
1	18518 98.75	235 1.25	18753
2	18692 98.88	212 1.12	18904
3	18343 98.62	256 1.38	18599
4	18525 98.69	246 1.31	18771
5	18809 98.89	211 1.11	19020
Total	92887	1160	94047

Figure 20. Distribution of goods and bads across the five folds

As we see in *Figure 20* the distribution of bads is fairly uniform, hovering between 1.25% and 1.11%, as are the total number of records between 18,500 and 19,020.

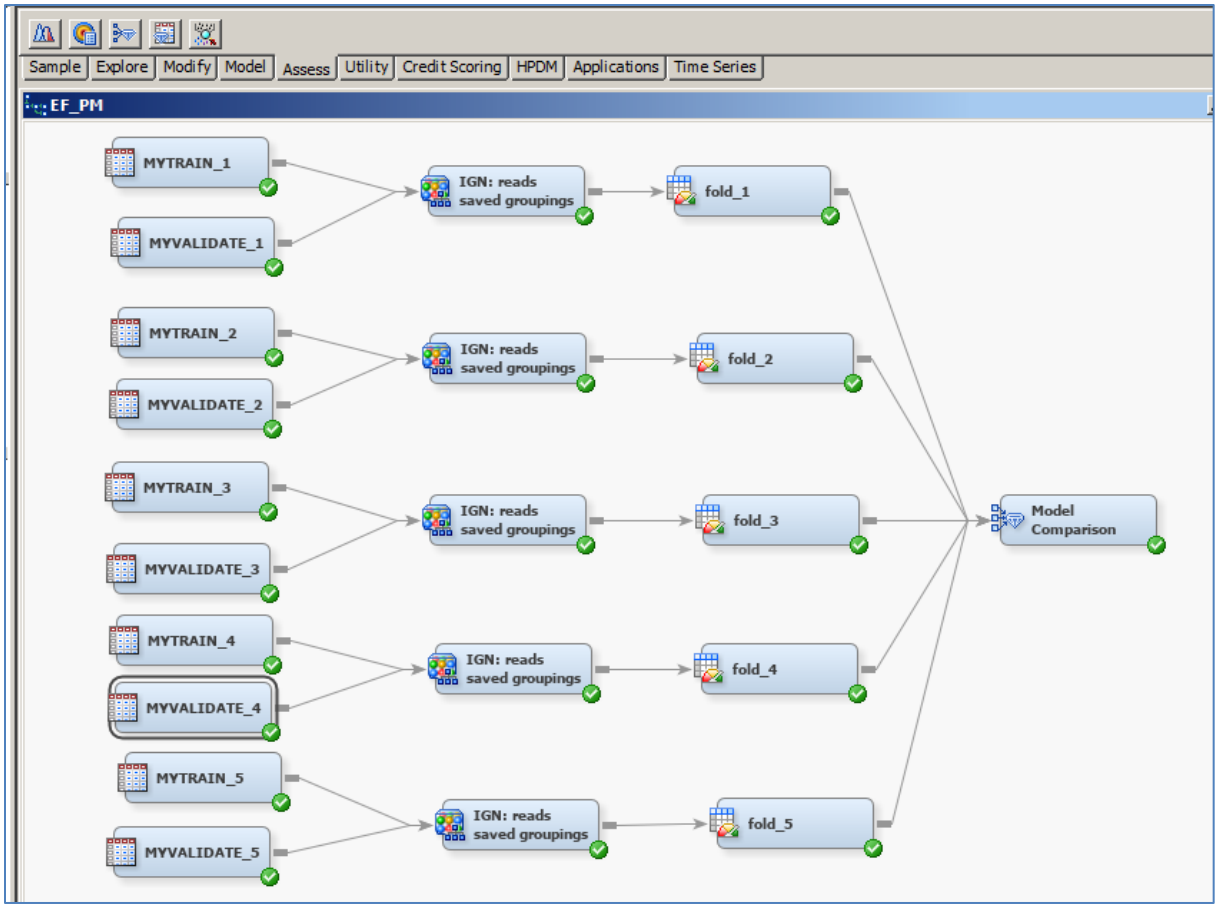


Figure 21. Five Fold Cross validation

Sample	Training			Validation		
	ROC	Gini	KS	ROC	Gini	KS
Fold 1	0.772	0.544	0.395	0.758	0.515	0.403
Fold 2	0.766	0.532	0.390	0.781	0.563	0.421
Fold 3	0.771	0.543	0.397	0.759	0.519	0.408
Fold 4	0.767	0.535	0.405	0.774	0.547	0.400
Fold 5	0.772	0.544	0.404	0.755	0.509	0.367
Average	0.770	0.540	0.398	0.765	0.531	0.400

Figure 22. Model performance statistics for Five Fold cross validation

The average for the training ROC at 77.7% and KS at 39.8% are very close to the final model at 76.9% and 40.0% respectively, as is the Gini of 54.0% to the final model's Gini of 53.8%. The validation numbers are very close to the training numbers as well.

Figure 23 below offers a graphic imagine reflecting the results of the *Table 22* above.

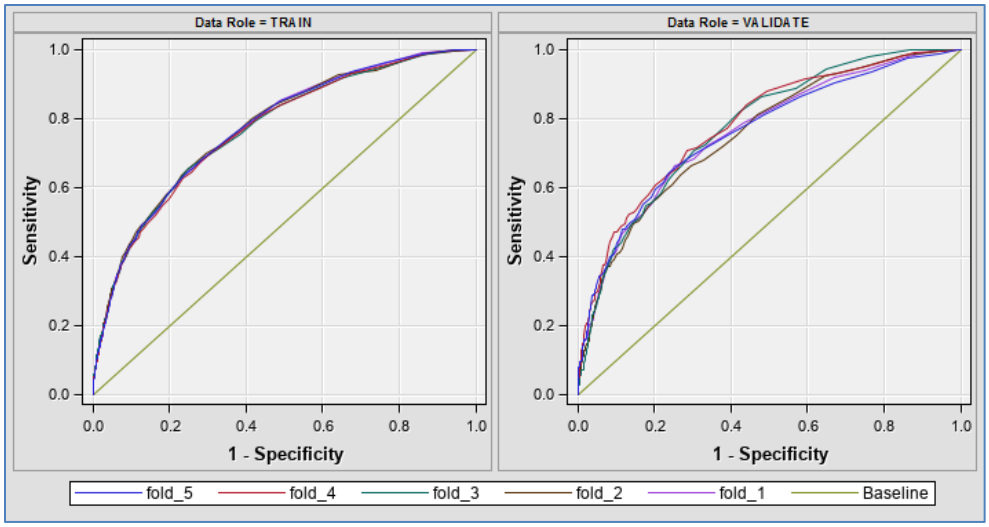


Figure 23. Graphs of Training and Validation ROC curves

SIMULATION

In order to run a simulation of the process MRM connected the nodes as seen in *Figure 24* below.

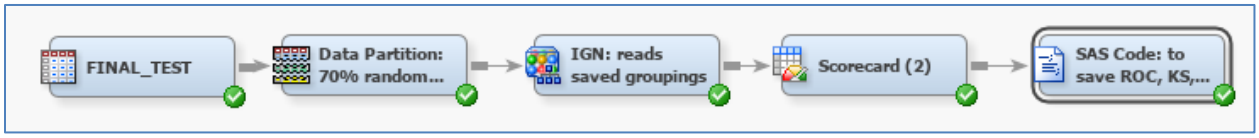


Figure 24. Enterprise Miner Nodes to create model process

The data is partitioned (randomly with replacement) at a 70% for training, 30% for validation; the Interactive node reads the fixed grouping that were saved at the model creation. The point is to run the process holding the parameters constant, varying only the data input. The code to save a single run is:

SAS Code to save AUR and KS from scorecard node:

```
title 'save _AUR_ Area Under ROC from the Scorecard node';

proc print data=&EM_LIB..scorecard2_strength;
  run;

data aur(keep = train_aur validate_aur)
      ks(keep = train_ks validate_ks) ;
  set &EM_LIB..scorecard2_strength ;
    if STAT = '_AUR_' then do ;
      train_aur = train ;
      validate_aur = validate ;
      output aur ;
     end ;
    if STAT = '_KS_' then do ;
      train_ks = train ;
      validate_ks = validate ;
      output ks ;
```

```
    end ;
   run ;
  data sasdb.myAUR ;
    merge aur ks ;
  run ;

proc print data=sasdb.myAUR;
  run;

proc print data=&EM_LIB..scorecard2_strength;
  run;

data sasdb.myest(keep = Intercept
WOE_BZ_LGC004
WOE_BZ_TTB001
WOE_BZ_TTP079
WOE_min_vantage_gp
WOE_sblend_gp3
WOE_portfolio_pd
WOE_S1_IFTC01_CT);
  set &EM_LIB..scorecard2_emestimate
(where = (_name_ = 'new_def') );
  run;

proc print data=sasdb.myest;
  run;
```

Right-clicking on the node, “SAS Codes to save ROC, KS” and checking off “Create Model

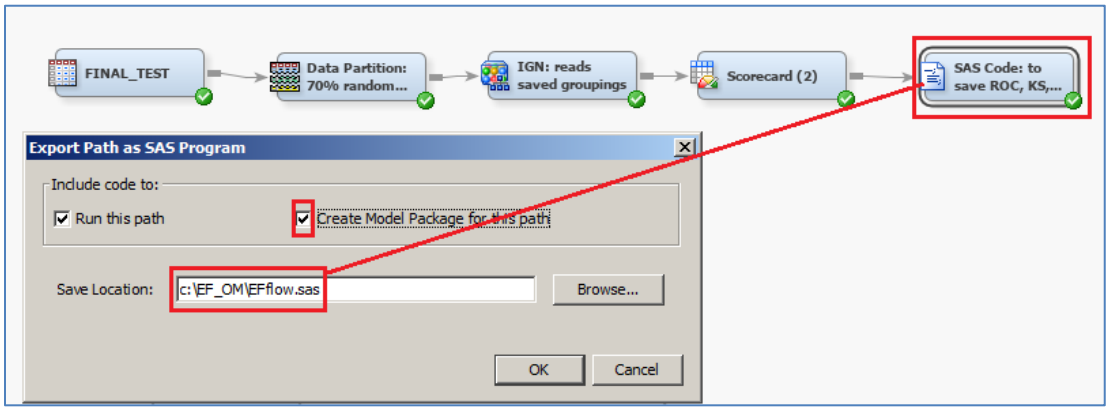


Figure 25. Enterprise Miner Node to create SAS code of diagram for simulation

Package for this path” opens up a window permitting us to store the SAS code equivalent of the preceding nodes. Running this batch code once gives us the current values of ROC, KS, etc. Including this code within a processing loop (“Rerun” was specified on the Input Node), “EFFLow.sas,” will allow the program to run from beginning to end, re-estimating the training/validation split giving us a simulation for the number of iterations specified. The code that determines the number of simulations (controlled by the macro “times” below) and append each result to datasets is:

```
options mprint symbolgen macrogen mlogic ;
libname sasdb "C:\grezek\proj\EF_OM\final" ;

%global times intera ;

data _null_ ;
  call symputx('times',100) ;
  call symputx('intera',1) ;
run ;

%macro spin ;
  %do spinit = 1 %to &times. ;


  %inc "C:\grezek\proj\EF_OM\code\EFFlow.sas" ;
  proc append data = sasdb.myaur base =
sasdb.myaurvalues_i&intera.t&times force ;
  proc append data = sasdb.myest base =
sasdb.myestvalues_i&intera.t&times force ;

  run ;
%end ;
run ;
%mend ;

%macro andag ;
  %do j = 1 %to &intera. ;
    %spin ;
```

```
  %end ;
run ;
%mend ;

run ;
%andag ;
run ;
```

is used to run the following code, EFFlow.sas, in a loop to create the simulations; this code is too long to include here, but is available upon request. In the code you will note that

- &times is the number of interations
- `sasdb.myaurvalues` collects the ROC statistics
- `sasdb.myest` collects the parameter estimates

At the end of each iteration we save the results for ROC and KS and append the data to `sasdb.myaurvalues`; also we captured the model parameters in `sasdb.myest` to see whether there was a shift in these values as illustrated below in *Figure 26*:

We ran simulations of 1000, which took 48 hours to complete, but noticed that the simulations converged at far fewer iterations. After a few trials, we judged that 100 iterations was reasonable. The result is a tight grouping for the model statistics.

TOLERANCE INTERVALS

To check whether the simulations were close to the predicted values, MRM estimated the tolerance intervals of the spread of values around the average[1]. Below is a graph (*Figure 26*) which gives an intuitive explanation of the difference between confidence intervals and tolerance intervals.

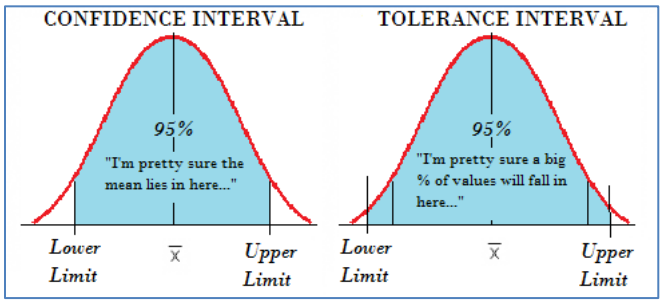


Figure 26. Graph of difference between Confidence Interval and Tolerance Interval.

In SAS/QC proc capability can be used to calculate the tolerance intervals using the code:

```
proc capability data=test;
  ods select Intervals3;
  var y;
```

[1] For a discussion of tolerance intervals see *Confidence Intervals vs Prediction Intervals vs Tolerance Intervals*, *https://statisticsbyjim.com/hypothesis-testing/confidence-prediction-tolerance-intervals/*

```
  intervals y / method=3;
run;
```

The datastep code to obtain the same result in base SAS:

```
%macro tolint(var,ds) ;
proc means data=&ds noprint;
  var &var;
  output out=out(drop=_type_ _freq_) n=n mean=mean std=std;
run;
data _null_ ;
  set out ;
  call symputx('n',n);
  call symputx('xbar',mean) ;
  call symputx('std',std) ;
run ;

data &var ;
  length var_name $ 12 ;
  set out ;
  var_name = "&var." ;
xbar = &&xbar;
s = &std;
n = &n;
p = 0.95;
alpha = 0.05;
df = n-1 ;
/* compute Lower & Upper tolerance limits using QC formula */
z = quantile("normal", (1+p)/2);
chi2 = quantile("chisq", alpha, n-1);
delta = s*z*(1 + 1/(2*df))*sqrt( (n-1)/chi2 );
Lower = xbar - delta;
Upper = xbar + delta;
run;

proc print data = &var;run;
run ;
%mend ;
run ;

data _null_ ;
  call symputx('var1','Train_aur') ;
  call symputx('var2','Validate_aur') ;
  call symputx('var3','Train_ks') ;
  call symputx('var4','Validate_ks') ;
run ;
```

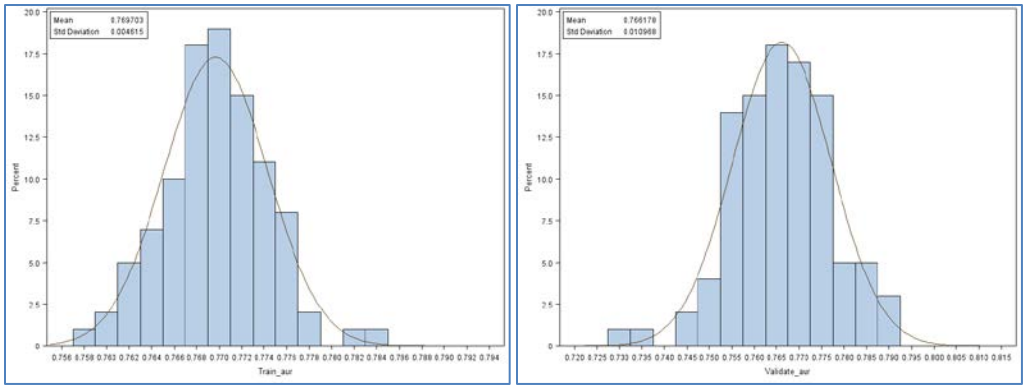
In *Table 27* below we see the tolerance intervals include the variable mean of the model statistics:

Statistic	N	p	alpha	df	Lower	mean	Upper	std
Train_aur	100	0.95	0.05	99	0.7594	0.7697	0.7800	0.0046
Validate_aur	100	0.95	0.05	99	0.7417	0.7662	0.7907	0.0110
Train_ks	100	0.95	0.05	99	0.3813	0.4023	0.4232	0.0094
Validate_ks	100	0.95	0.05	99	0.3519	0.4005	0.4492	0.0218

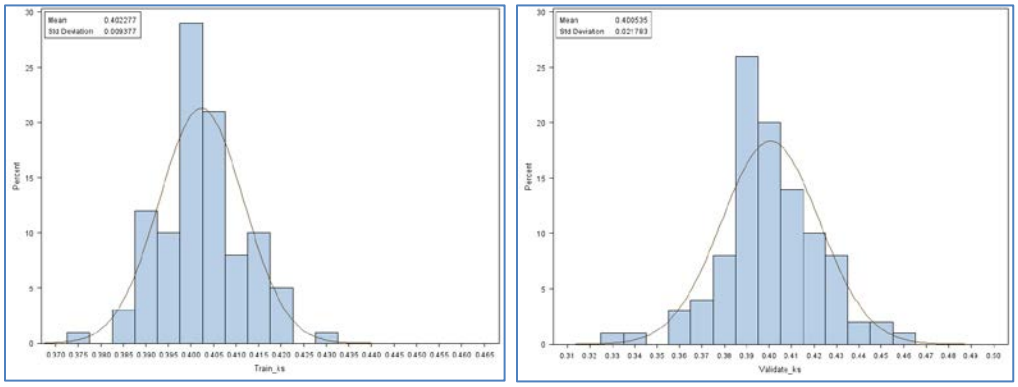
Table 27. 95%/95% Tolerance Interval

We can say for the two-sided test that at the 95% confidence level (1-alpha) that 95% of the population is within the tolerance level. We notice that the mean of the simulations is centered between the lower and upper limits of the tolerance intervals. The values for the corresponding standard deviations confirm the narrow spread of the estimates for the training of the ROC and KS (0.0046 and 0.0094 respectively). While the standard deviation of the validation ROC and KS are less tight at 0.0110 and 0.0218 respectively, the means are within the tolerance intervals and the distribution is still supportive of the model.

Below in *Figures 28-31.*, we see a tight grouping around the mean of the simulations for ROC and KS.



Figures 28-29. Distribution of ROC for train and validate data



Figures 30-31. Distribution of KS for train and validate data

The same analysis was performed on the model parameters, as we see in *Table 32* below we notice that the standard deviations are acceptable and that the means of the parameters are within the lower-upper tolerance intervals. The advantage of including seven variables, as opposed to fewer, is that the model is more robust in that model can withstand the variations in the data for a couple of variables without causing serious swings in model output.

Parameter	N	p	alpha	df	Lower	mean	Upper	std
Intercept	100	0.95	0.05	99	-4.3739	-4.3713	-4.3688	0.0011
BZ_LGC004	100	0.95	0.05	99	-0.3431	-0.2217	-0.1002	0.0544
BZ_TTB001	100	0.95	0.05	99	-0.6388	-0.5847	-0.5306	0.0242
BZ_TTP079	100	0.95	0.05	99	-0.5165	-0.4153	-0.3140	0.0453
S1_IFTC01_CT	100	0.95	0.05	99	-0.6581	-0.5980	-0.5380	0.0269
min_vantage_gp	100	0.95	0.05	99	-0.5105	-0.4470	-0.3834	0.0285
portfolio_pd	100	0.95	0.05	99	-0.8276	-0.8003	-0.7729	0.0123
sblend_gp3	100	0.95	0.05	99	-0.6659	-0.6188	-0.5717	0.0211

Table 32. Model parameters of tolerance level

The histograms in *Figures 33-40* indicate a relatively tight grouping around the means of the parameters.

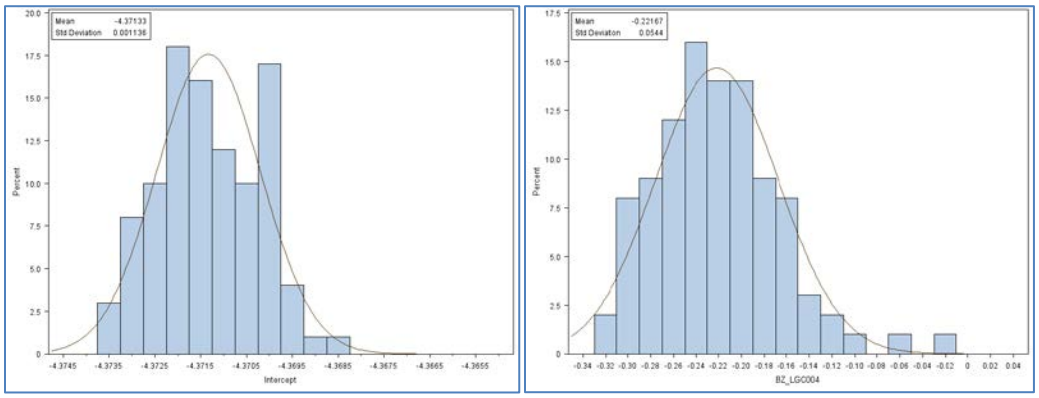
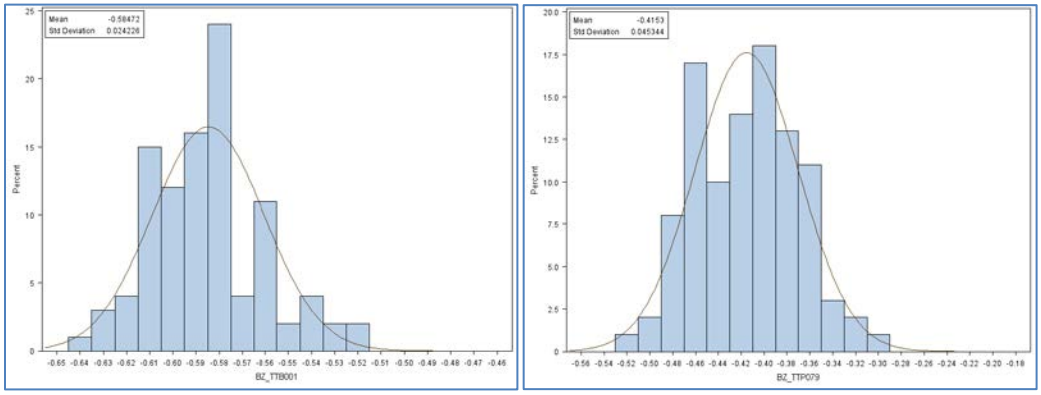
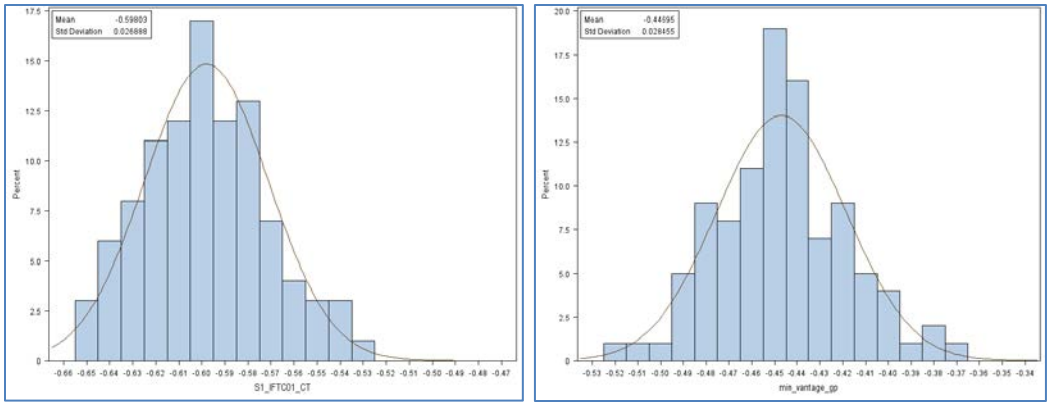


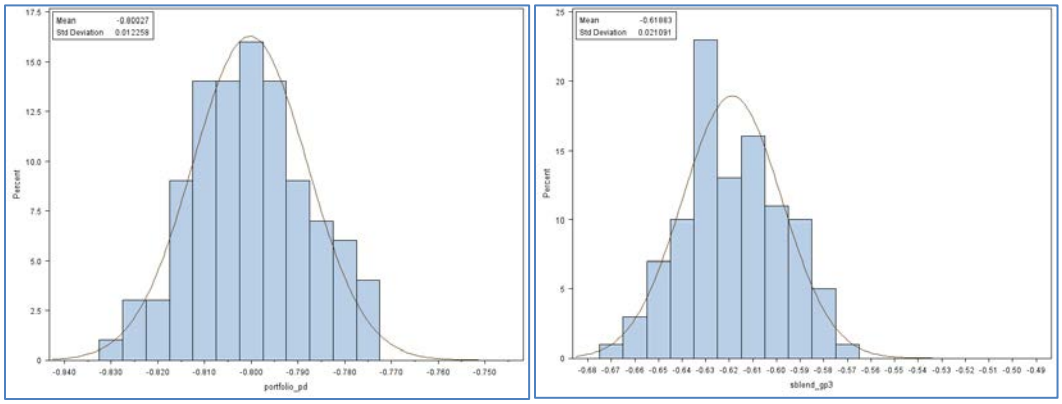
Figure 33-34. Distribution of intercept and BZ_LGC004 parameters



Figures 35-36. Distribution of BZ_TTB001 and BZ_TTP079 parameters



Figures 37-38. Distribution of S1_IFTC01_CT and min_vantage_gp parameters



Figures 39-40. Distribution of portfolio PD and sblend_gp3 parameters

IGN DETERMINED BINS

The Interactive Grouping Node in Enterprise Miner can be set to determine the bins by optimizing on, for instance, the Information Values of the variables. While we believe the correct way to determine the bins is the method followed by the developer, that of getting buy in from the business, using various procedures to determine variables and their statistics, returning to the business for confirmation, etc., MRM thought it useful to let the Interactive Grouping determine the groupings just to see what would happen. The Model Comparison Node is used to compare the results of the original model (created in the string of nodes on the top of F*igure 18)* to the results of the string of nodes on the bottom in which we allow Enterprise Miner determine the bins.

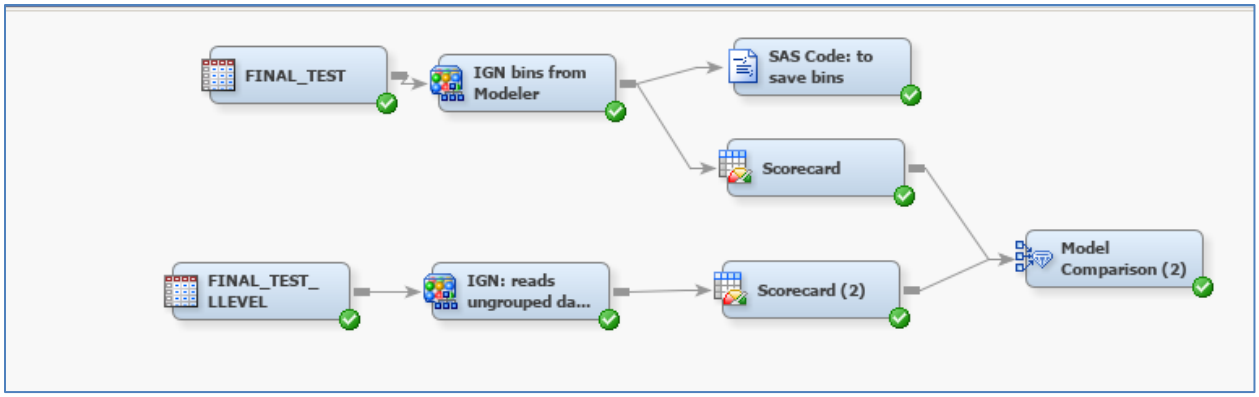


Figure 41. Enterprise Minor Nodes to Create SAS code for comparing binning approach

MRM expects the bins to make sense in terms of IV, distribution of defaults and observations across bins, and a scorecard with higher ROC, KS and Gini statistics than that of the original model.

Run	ROC	KS	Gini
Original	0.7690	0.5380	0.3970
IGN determined	0.7710	0.5420	0.4000

Table 42. Comparison of original and IGN determined statistics

While we were able to obtain modestly higher statistics as we see in *Table 42* above we note that the difference is slight. We notice that the ROC curves in *Figure 43* below are almost identical.

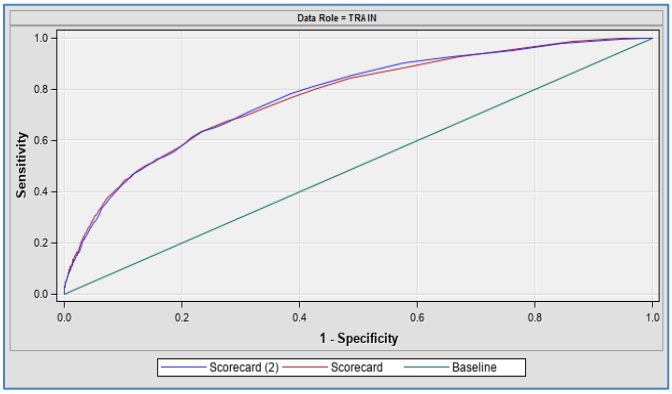


Figure 43. Graph of ROC curves for different binning approach

What MRM did not expect is the unsupervised binning produced by the program to be as close in many respects to the original model.

CONCLUSION

The initial set up of a model in Enterprise Miner / Scorecard is within reach of the SAS user. The system offers an ease of use and capabilities beyond what is obtained from running a diagram. The underlying tables, the interaction with SAS code Nodes expands the functionality

REFERENCES

Frost, Jim. "Confidence Intervals vs Prediction Intervals vs Tolerance Intervals." Statistics by Jim: Making statistics intuitive. Available at https://statisticsbyjim.com/hypothesis-testing/confidence-prediction-tolerance-intervals.

ACKNOWLEDGMENTS

Special thanks to Liz Edwards and especially Mike Stockstill of SAS Support.

CONTACT INFORMATION

George Rezek
TCF Bank
tel. 612 247-3226
grezek@tcfbank.com