Content Evaluator (CE) Tool – version 5

The Content Evaluator (CE) tool – version 5 is the latest version of the Merkle developed process for automating preliminary predictive modeling. It includes sampling, EDA and recoding, variable reduction, model training and model evaluation.

While CE can automate much of the “science” part of modeling, it cannot do the “art” of modeling. The list of variables that comes out of the CE process should be used as a starting point for you to use in building your final model. No set of code can replicate what a good analyst can intuit.

All code can be found on the Shared drive at /mnt/projects/shared/pst\_qmgisi/Modeling/CE/. The files you will need are:

* CE0\_Macro\_Call\_v5.sas
* Variable explore.sas
* DS Variable.xlsm

Copy these to your project directory to use as your working files that you will modify.

# Variable List Setup

The CE process needs to be told how to process each of your independent variables. This is done by specifying the variable type for each variable:

* Binary – these are variables with values of Y/N or 0/1. The code will recode all records with a value of Y, y or 1 to 1 and everything else to 0. These variables can be character or numeric.
* Nominal – these are variables where there is no inherent ordering to the values of a variable. Examples are occupation and Mosaic type. These variables can be character or numeric.
* Ordinal – these are variables with a limited number of values that have an inherent ordering. Examples are education level and number of children. These variables must be numeric.
* Continuous – these are variables with an unlimited number of values that have inherent ordering. Examples are raw income and total purchase dollars. These variables must be numeric.

A tool has been created to help you with creating the necessary lists.

1. Open the SAS program *variable explore* that you have copied to your project directory.

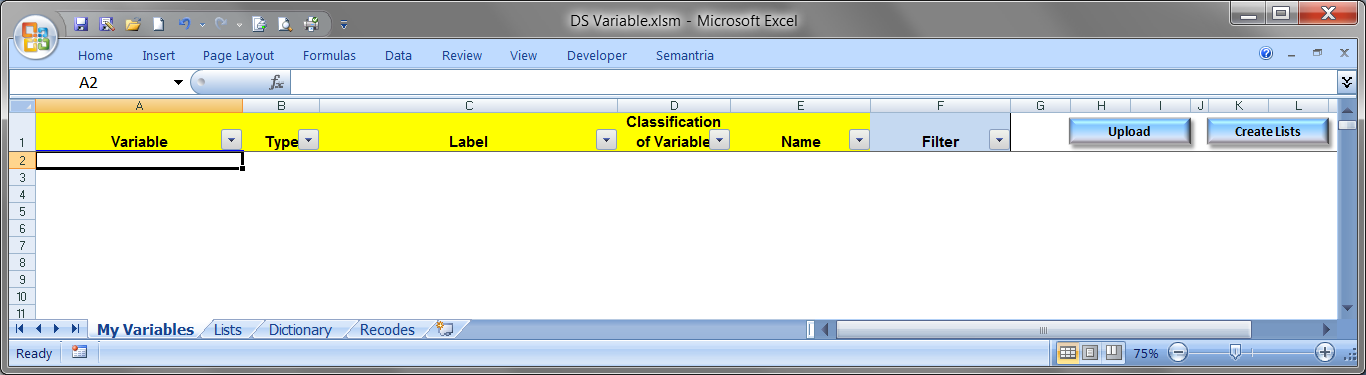
%let loc=/mnt/projects/locked/pst\_qmgisi/Modeling/CE Rebuild/;

libname lib "&loc.sasdata";

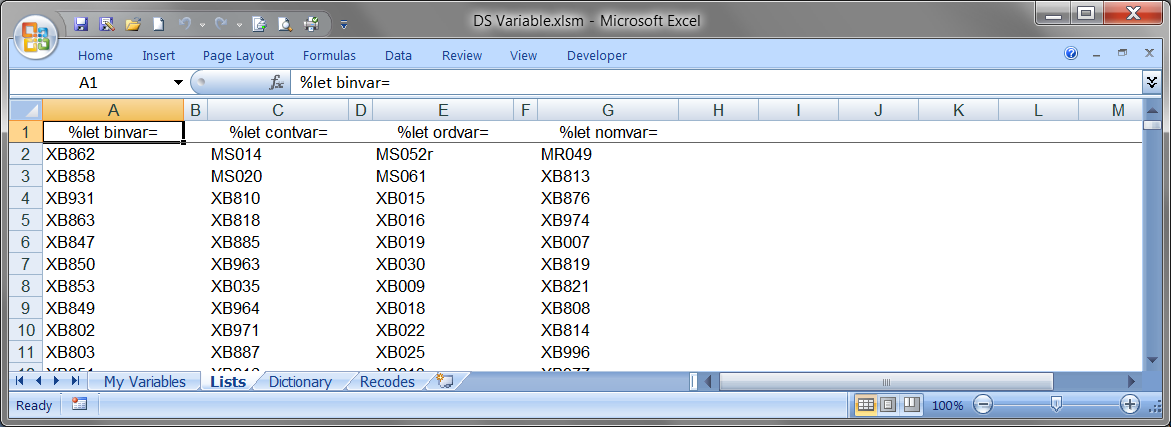
%let inds=lib.bin\_file;

* 1. Set the macro variable **loc** to point to your project folder
  2. Set libname **lib** to the location of your SAS dataset
  3. Set the macro variable **inds** to the name of your SAS dataset
  4. Run the code. This will write out an Excel workbook called *vars.xls* to the location specified by **loc**

1. Open the Excel workbook DS Variable.xlsm that you have copied to your project directory.
   1. Enable the macros



* 1. Click the button called **Upload** on the *My Variables* page. This will open a file selector box. Navigate to the folder **loc**, click the *vars.xls* file and click open. A pop-up will inform you that the file has been successfully uploaded. Click OK to close.
  2. You will now see a table with all of your available variables. If the variable is a DataSource variable, it will pre-populate the Classification and Name. If the variable is in the standard DS recodes (available in step 1 of CE) then the Name field will use the name for the recoded variable rather than the raw variable. You can change the Classification of any of these variables. We have just tried to give you a good starting point and minimize the amount of work you have to do.
  3. If the variable is not in the DataSource variable list, the Classification and Name will be blank. You need to set these. The Classification field has a drop-down list of options for you to select from. The possible values are binary, nominal, ordinal, continuous, idvar, depvar and exclude. Idvar is used for the variable that identifies unique records. Depvar is used for your dependent variable. Exclude is used to exclude the variable from the list of independent variables. If you change the Classification to anything other than exclude, the Name field will populate with the variable name.  
       
     Column F is set up to help you filter the variable names if you have a lot of client specific variables. Change cell F1 to a string to look for in column A to identify a block of similar variables. For example, if all of your binary variables end in \_flg, change F1 to \_flg. Then filter for rows that have a blank for the classification (are not yet defined) and is not blank in column F. Set the first row to binary and copy to the rest of the selected rows. Continue with other strings that can be used to identify other blocks of similar variables. See the "Tips" document for a more detailed example.
  4. Once you have completed the classification of all of your variables, click the button **Create Lists** to populate the lists of variables by type in the sheet *Lists* and create the necessary text files. Files will be created in folder **loc**.



The standard document names are:

* + - binary – binvar\_list.txt
    - nominal – nomvar\_list.txt
    - ordinal – ordvar\_list.txt
    - continuous – contvar\_list.txt

The code will erase any existing files with these names before creating new ones.

* 1. Save your workbook so if you need to make changes you will not need to start from scratch unless the variables available changes.

# CE SAS Code

Open the program CE0\_Macro\_Call.sas. This is the program that contains all of the macro variables needed for the CE process and will call the necessary macros. Where possible, default values for the macro variables have been set to provide guidance.

Below is the code and an explanation of what needs to be entered for each section.

%let loc=/mnt/projects/locked/pst\_qmgisi/Modeling/CE Rebuild/;

libname lib "&loc.sasdata";

libname out "&loc.ce\_output";

These three lines of code tell the process where things are located:

* loc – this is the path to your project directory. It will be used as the base for all library names and paths
* libname lib – this is the location for your input dataset
* libname out – this is where any new datasets created will be written out to

%include "&loc.contvar\_list.txt";

%include "&loc.ordvar\_list.txt";

%include "&loc.nomvar\_list.txt";

%include "&loc.binvar\_list.txt";

These lines will call the variable lists you have created and create the macro variables binvar, contvar, ordvar and nomvar. If you are missing a classification, comment out the appropriate line so it will be skipped.

The macro variable definitions are broken into two sections – mandatory and optional. The mandatory are variables that are unique to your dataset so must be defined. The optional all have default values that can be used.

## Mandatory Macro Variables

\*\*\* General Macro Variables \*\*\*;

%let path\_output=&loc.ce\_output/;

%let inds=lib.bin\_file;

%let id = KL\_id;

%let dep\_var=resp;

%let binary\_dv=Y;

%let weight= ;

These macro variables are used across multiple steps in the process:

* path\_output – this is the path any reports and logs will be written to
* inds – this the name of your input dataset including the library name
* id – this is the name of the variable that is the unique identifier for records in your file
* dep\_var – this is the name of the variable that is your dependent variable
* binary\_dv – this is a flag to tell the process if your dependent variable is binary or not. Acceptable values are Y for binary and N for anything else. If you have an ordinal dependent variable, you can treat it as continuous for this process (set to N) and it will work. If you have a nominal dependent variable you should be using the PST Nominal modeling process as this has code to handle this situation.
* weight – this is the name of the variable that you want to use as a weight. If you do not want to use weights, set this variable to blank (**%let weight = ;**)

\*\*\* Macro 1: Sampling macro variables \*\*\*;

%let split\_if = if ranuni(12345)<0.7;

%let exclusion\_if=if &Dep\_var =.;

%let DS\_Present= Y;

These are the macro variables that are needed for the first step of the CE process – sampling:

* split\_if – this is your split condition to identify the learning portion of your file. If you already have a flag on your file to identify this, you would set split\_if to something like **if lrnval='L',** where **lrnval** is the name of your flag variable. If you do not already have a flag on your file to identify this, you would set split\_if to something like **if ranuni(12345)<0.7**. The normal split is 70% learning, 30% validation.
* exclusion\_if – this is a condition that will cause records from your input file to be excluded. Common conditions are **if &dep\_var = .** and **if &dep\_var < 0**. If you do not have an exclusion clause, set to blank (**%let exclusion\_if = ;**)
* DS\_Present – this tells the process whether or not you want to apply the standard DataSource recodes. Set to Y to apply.

\*\*\* Macro 3: Variable Reduction macro variables \*\*\*;

%let fast\_opt = N;

This variable is a shortcut to minimizing runtime by eliminating most tests from the variable reduction step. By setting this to Y, all tests will be turned off except multivariate regression – logistic if the dependent variable is binary, linear if continuous. Since only one test will be run, the number of sources to pass will also be set to 1. The benefit of this is faster running time. The downside is that some potentially predictive variables will be excluded.

## Optional Macro Variables

\*\*\* General Macro Variables \*\*\*;

%let prefix=R1\_;

%let keep\_list=&id &dep\_var &weight mod\_val\_test;

These macro variables are used across multiple steps in the process:

* prefix – this is the prefix that will be used when creating the recoded independent variables. As an example with prefix set to R1\_, raw variable *sales* will be recoded to *R1\_sales*.
* keep\_list – this is the list of variables you want to keep in addition to your independent variables. You should always include &id, &dep\_var, &weight and mod\_val\_test. You can add any additional variables you like. Any variables in your input dataset that are not listed in one of the independent variable lists or keep\_list will be dropped from the datasets created by the process.

\*\*\* Macro 1: Sampling macro variables \*\*\*;

%let split\_if = if lrnval = 'L';

%let exclusion\_if=if &Dep\_var =.;

%let DS\_Present= Y;

These are the macro variables that are needed for the first step of the CE process – sampling:

* Path\_DS – this is the path to the standard DataSource recodes. Do not change.
* bootstrap – this is how you can request sampling to achieve a desired proportion between 0’s and 1’s. This only applies if your dependent variable is binary. Set to Y to invoke.
* oversampled\_rr – if you are doing sampling, this is the desired ratio of 1’s to 0’s. This value should be between .1 and .5.
* min\_num\_resp – if you are doing sampling, this is the minimum number of 1’s to be selected
* Seed – if you are doing sampling, this is the seed to use when doing random selects

\*\*\* Macro 2: Recoding macro variables \*\*\*;

%let Profiling = Y;

%let missrate = .75;

These are the macro variables that are needed across all variable types for the second step of the CE process – recoding:

* Profiling – this allows you to do profiling on all of the recoded variables. Profiling will automatically be done during step 5, final model selection, on the variables in the final model. If you want profiling on all variables, set this to Y.
* missrate – this is the maximum missing percent allowed for any variable. If the missing rate is greater than this number, the variable will be discarded.

\* Binary and Nominal variable types;

%let concrate = .9;

For binary and nominal variables, concrate is the maximum percent of the file that can be in a single value. If an independent variable has more than this, it will be discarded because it is too concentrated to provide stable and useful information.

\* Nominal variable types;

%let valcnt = 50;

%let minbinnc = 500;

%let minbinnp = .05;

%let talpha = 0.05;

%let bonfer = N;

%let nom\_method = INDEX;

These are the macro variables needed to process nominal variables:

* valcnt – this is the maximum number of unique values a nominal variable is allowed to have. Variables with more than this number of unique values will be discarded. If you do not want to restrict the number of values, set valcnt to zero.
* minbinnc and minbinnp – these are the minimum count and minimum percent of file in a bin for it to be usable. Nominal variables are grouped into bins based on similar performance. Minbinnc and minbinnp are used to determine if there are enough records in the bin for the evaluation to be valid. Either one can be set to zero to only use the other. For example, with a file of 50,000 records, setting minbinnc to 500 and minbinnp to 0 will force each bin to have at least 500 records in it. Setting minbinnc to 500 and minbinnp to .05 will force each bin to have at least 2,500 records in it (50,000 x .05 = 2,500 is greater than 500). Setting minbinnc to 1,000 and minbinnp to .01 will for each bin to have at least 1,000 records in it. (50,000 x .01 = 500 is less than 1,000).
* talpha – this is the t-test significance level for collapsing bins.
* bonfer – this tells the process if you want to use a Bonferoni adjustment for talpha. Set to Y to use this option.
* nom\_method – this is the method to use for the recoded variables. The options are:
  + Binary – this creates a new binary variable for each bin. This is the method used in the previous version of CE.
  + Index – this creates a single new variable with the index of the bin average dependent value to the overall average used for the values.
  + Mean – this creates a single new variable with the bin average dependent value as the values.

Example: a variable is collapsed into four bins with the following values for average dependent and index

|  |  |  |
| --- | --- | --- |
| Bin | Average | Index |
| Overall | .16667 |  |
| 1 | .10959 | 65.75 |
| 2 | .11815 | 70.89 |
| 3 | .17002 | 102.01 |
| 4 | .21542 | 129.25 |

If you use nom\_method = BINARY you will get three new binary variables – one for bin 1, one for bin 2, and one for bin 3. If you use nom\_method = INDEX you will get one new variable with four possible values – 65.75, 70.89, 102.01, 129.25. If you use nom\_method = MEAN you will get one new variable with four possible values - .10959, .11815, .17002, .21542.

\* Continuous and Ordinal variable types;

%let pvalue = .05;

%let min\_size = 500;

%let num\_category = 10;

%let equal\_dist = N;

These are the macro variables needed to process both continuous and ordinal variables:

* pvalue – this is the p-value threshold to include a variable. Once a variable is recoded, a proc logistic or proc reg will be run with only the independent variable under consideration in the model statement. The probability value for the appropriate test will be used to determine if the variable has any predictive power or not. If the p-value is greater than the value in pvalue, the variable is discarded.
* min\_size – if you choose Equal Response for the method of missing value imputation, min\_size gives the minimum number of missing records to use this method. If you have chosen Equal Response and the missing count is less than this number for a variable, the median will be used instead.
* num\_category – if you choose to do profiling, this is the maximum number of categories that will be used.
* equal\_dist – if you choose to do profiling, you can choose equal distance for dividing variables into groups by setting this to Y. If set to N, then an equal number of records will be assigned to each group.

\* Continuous variable types;

%let p\_lo = 1;

%let p\_hi = 99;

%let impmethodC = median;

%let stdmethodC = STD;

%let cap\_flrC = Y;

%let transformationC = Y;

These are the macro variables needed to process continuous variables:

* p\_lo and p\_hi – these is the lower and upper percentiles to be used when checking a variable for constant value. The code will determine the value of the independent variable at the lower and upper percentiles. If the values are the same, the variable is considered constant and will be discarded.
* impmethodC – this is the method to use for missing value imputation. The options are ER for Equal Response or any proc stdize method. If you choose a proc stdize method, the location associated with the method will be used for missing values. If you choose a method that needs a parameter, you will need to include it.

|  |  |
| --- | --- |
| Method | Location |
| MEAN | mean |
| MEDIAN | Median |
| SUM | 0 |
| EUCLEN | 0 |
| USTD | 0 |
| STD | mean |
| RANGE | Minimum |
| MIDRANGE | Midrange |
| MAXABS | 0 |
| IQR | Median |
| MAD | Median |
| ABW(c) | Biweight one-step M-estimate |
| AHUBER(c) | Huber one-step M-estimate |
| AWAGE(c) | Wave one-step M-estimate |
| AGK(p) | Mean |
| SPACING | Mid-minimum spacing |
| L(p) | L(p) |

* stdmethodC – this is the standardization method to use. You can specify any of the proc stdize methods or NO to not do standardization.
* cap\_flrC – this controls whether capping and flooring are done to handle outliers before standardization is done. Set to Y to do.
* transformation – this controls whether transformations are done. The transformations include square, square root and natural log. Set to Y to include evaluation of the transforms as well as the natural values of the variables.

\* Ordinal variable types;

%let impmethodO = mean;

%let stdmethodO = Range;

%let cap\_flrO = N;

%let transformationO = N;

These are the macro variables needed to process ordinal variables:

* impmethodO – this is the method to use for missing value imputation. The options are ER for Equal Response or any proc stdize method. If you choose a proc stdize method, the location associated with the method will be used for missing values. If you choose a method that needs a parameter, you will need to include it.
* stdmethodO – this is the standardization method to use. You can specify any of the proc stdize methods or NO to not do standardization.
* cap\_flrO – this controls whether capping and flooring are done to handle outliers before standardization is done. Set to Y to do.
* TransformationO – this controls whether transformations are done. The transformations include square, square root and natural log. Set to Y to include evaluation of the transforms as well as the natural values of the variables.

\*\*\* Macro 3: Variable Reduction macro variables \*\*\*;

%let samplesize = 50000;

%let redu\_weight = N;

%let sources = 3;

These are the common macro variables that are needed for the third step of the CE process – variable reduction:

* samplesize – this is the maximum number of records that will be used in the variable reduction processes. If you have fewer records than this number then all records will be used. If you have more records than this number than a sample of this size will be taken. The larger this number is, the longer this process will take so it is recommended that you use at most 50,000. This is generally sufficient to evaluate the usefulness of your independent variables.
* redu\_weight – this allows you to use weights in this step. Set to Y to use the variable specified in *&weight* as weights. Set to N to not use weights.
* sources – this is the minimum number of sources to survive the variable reduction phase. There are multiple methods available for variable evaluation. Each has a criterion for passing that method. *Sources* is the number of passing grades a variable must have or it will be discarded. Make sure that this number is less than or equal to the number of methods selected.

\* Univariate regression option;

%let univ\_reg = Y;

%let maxpuni = .0001;

These are the macro variables needed for the univariate regression method of variable evaluation. This method evaluates each independent variable by itself for a relationship with the dependent variable.

* univ\_reg – this tells the process whether you want to use this method. Set to Y to include; set to N to skip.
* maxpuni – this is the maximum p value for passing this evaluation method.

\* Correlation option;

%let correlation = Y;

%let corrcut = .01;

These are the macro variables needed for the correlation method of variable evaluation. This method evaluates each independent variable based on its correlation with the dependent variable.

* correlation – this tells the process whether you want to use this method. Set to Y to include; set to N to skip.
* corrcut – this is the minimum correlation for passing this evaluation method.

\* Principal components option;

%let principal = Y;

%let nprin = 10;

%let minprin = .5;

These are the macro variables needed for the principal components method of variable evaluation. This method evaluates each independent variable based on its best correlation with the factors created.

* principal – this tells the process whether you want to use this method. Set to Y to include; set to N to skip.
* nprin – this is the number of principal components that should be used.
* minprin – this is the minimum factor correlation for passing this evaluation method.

\* Cluster option;

%let cluster = Y;

%let maxc = 20;

%let maxratio = .5;

These are the macro variables needed for the cluster method of variable evaluation. This method evaluates each independent variable based on its R\_squared ratio when clusters are created.

* cluster – this tells the process whether you want to use this method. Set to Y to include; set to N to skip.
* maxc – this is the number of clusters that should be used.
* maxratio – this is the maximum R-squared ratio for passing this evaluation method.

\* Linear regression option;

%let regression = Y;

%let alphareg = .05;

These are the macro variables needed for the linear regression method of variable evaluation. This method builds a linear regression model using all independent variables together. If a variable enters the model, it passes the evaluation.

* regression – this tells the process whether you want to use this method. Set to Y to include; set to N to skip.
* alphareg – this is the maximum alpha value for a variable to enter the model using forward selection.

\* Logistic regression option - only applicable if binary dependent variable;

%let logistic = Y;

%let alphalog = .05;

These are the macro variables needed for the logistic regression method of variable evaluation. This method is only available if you have a binary dependent variable. This method builds a logistic regression model using all independent variables together. If a variable enters the model, it passes the evaluation.

* logistic – this tells the process whether you want to use this method. Set to Y to include; set to N to skip.
* alphalog – this is the maximum alpha value for a variable to enter the model using forward selection.

\* Information value option;

%let information = Y;

%let decile = 20;

%let infvcut = .01;

These are the macro variables needed for the information value method of variable evaluation. This method evaluates each independent variable based on its information value.

* information – this tells the process whether you want to use this method. Set to Y to include; set to N to skip.
* decile – this is the number of groups the variables should be split into to calculate the information value of the variable.
* infvcut – this is the minimum information value for passing this evaluation method.

\* Maximum correlation between independent variables option;

%let ind\_correlation = Y;

%let maxcorr = .7;

These are the macro variables needed for evaluating the maximum correlation between independent variables. This method evaluates each independent variable based on its correlation to all other independent variables. If a correlation is too high the variable with the highest correlation to the dependent variable will be kept and the other discarded. This will override the results of the rest of the methods. Even if a variable is selected by 5 sources, if the correlation to another variable is too high it will not be included in the final variable list.

* Ind\_correlation – this tells the process whether you want to use this check. Set to Y to include; set to N to skip.
* maxcorr – this is the maximum correlation allowed between the independent variables.

\* Maximum correlation to dependent variable option;

%let ind\_dv\_corr = Y0;

%let max\_dv\_corr = .7;

These are the macro variables needed for evaluating the maximum correlation to the dependent variable. If a correlation is too high the independent variable will be discarded. This will override the results of the rest of the methods. Even if a variable is selected by 5 sources, if the correlation to the dependent variable is too high it will not be included in the final variable list.

* Ind\_dv\_corr – this tells the process whether you want to use this check. Set to Y to include; set to N to skip.
* Max\_dv\_corr – this is the maximum correlation allowed to the dependent variable.

\*\*\* Macro 4: Model selection and tuning \*\*\*;

%let sel\_alpha = .05;

%let includelist = ;

%let startlist = ;

%let excludelist = ;

%let criteria = c;

%let threshold = 0;

%let SQL\_join = union;

%let minimp = .01;

%let graph\_plot = Y;

These are the common macro variables that are needed for the fourth step of the CE process – model selection and tuning:

* sel\_alpha – this is the alpha level to use for addition and removal of variables from the model
* includelist – this is the list of variables that must be in the model
* startlist – this is the list of variables that must be in the first model. With backwards selection, these can be removed so are not guaranteed to be in the final model
* excludelist – this is a list of variables that should not be included in the model
* criteria – this is the metric to use during variable tuning. The options for a binary dependent variables are: AIC, SC, LogL2, Rsquare, SomersD, Gamma, TauA, c, Concord, Discon, LackFit, Lift\_Index, infv, ks. The options for a continuous dependent variables are: AIC, SBC, JP, Rsquare, AdjRsq, RMSE, CoeffVar, PC, Lift\_Index, gini, infv, ks. The default for binary is c and for continuous is AdjRsq. Variable tuning looks at this measure to determine if a variable should be discarded because it appears to be causing over fitting.
* threshold – this is the minimum change in the evaluation metric to include a variable
* SQL\_join – this is the type of join to use between file portions in variable tuning. Variable tuning will select a set of variables based on each portion of the learning file independently. It will then join the lists using this variable. If you use union (recommended), you will get all variables selected in either portion. If you use intersect, you will only get the variables that are in both portions.
* Minimp – this is the minimum relative importance to keep a variable.
* graph\_plot – this controls whether the process creates a PDF report on the variables selected by this step. Set to Y to create.

\*\*\* Macro 5: Final Model selection \*\*\*;

%let fin\_alpha = .05;

%let method = stepwise;

%let fin\_num\_category = 10;

%let fin\_equal\_dist = N; \*\* Use equal distance for dividing variables into groups for profiling (Y/N);

These are the common macro variables that are needed for the fifth step of the CE process – final model selection:

* fin\_alpha – this is the alpha level to use for addition and removal of variables from the model
* method – this is the selection method to use when building the final model. Options are forward, backward and stepwise.
* fin\_num\_category – this is the maximum number of categories that will be used in profiling of the variables in the final model.
* equal\_dist – if you choose to do profiling, you can choose equal distance for dividing variables into groups by setting this to Y. If set to N, then an equal number of records will be assigned to each group.
* fin\_equal\_dist – this controls whether equal distance will be used for dividing variables into groups. Set to Y to use. Set to N to use number of records.

\*\*\* Load code \*\*\*;

%inc "/mnt/projects/shared/pst\_qmgisi/Modeling/CE/CE\_Macros\_v5.sas";

This line of code will load the macros needed for the CE process.

\*\*\* 1.Sampling \*\*\*;

%let LogFile = "&path\_output/01\_CE\_Sampling\_Log\_File.log";

%let LstFile = "&path\_output/01\_CE\_Sampling\_LST\_File.lst";

**proc** **printto** log=&LogFile new; **run**;

**proc** **printto** print=&LstFile new; **run**;

%***CE\_Sampling***(inds=&inds, outds=out.CE1\_Resampled)

**proc** **printto**;

**run**;

This block of code will run the first step of the CE process – sampling.

1. If *exclusion\_if* is not blank, any records that meet the condition will be discarded.
2. If *DS\_Present* is set to Y, the standard DataSource recodes will be applied
3. If *bootstrap* is set to N then the portion of the file identified by *split\_if* will be split in half as model and validation portions and the remainder of the file will be the test portion. If *bootstrap* is set to Y and you have a binary dependent variable, then the portion of the file identified by *split\_if* will be sampled to meet the proportion *oversampled\_rr* with the minimum number of 1’s equal to *min\_num\_resp*. It will then be split in half as model and validation portions and the remainder of the file will be the test portion.
4. A new dataset, **CE1\_Resampled**, will be written to the output library. It will have a variable called *mod\_val\_test* that identifies the portions of the file.

\*\*\* 2.EDA,profiling and Recode \*\*\*;

%let LogFile = "&path\_output/02\_CE\_Var\_EDA\_Recode\_Log\_File.log";

%let LstFile = "&path\_output/02\_CE\_Var\_EDA\_Recode\_List\_File.lst";

**proc** **printto** log=&LogFile new; **run**;

**proc** **printto** print=&LstFile new; **run**;

%***CE\_EDA\_RECODE***(INSDN=out.CE1\_Resampled);

**proc** **printto**;

**run**;

This block of code will run the second step of the CE process – recoding. This code goes through each variable type that is present and applies rules to recode the raw data into a more usable form. It does not use test portion of the file.

All variables:

* All variables are checked for missing rate. If the missing rate is greater than *missrate*, then the variable is discarded as unusable.

Binary variables:

* If the length of the variable name plus the length of prefix is greater than 32 then the variable is discarded because the recoded variable name will be too long.
* The code will recode all records with a value of Y, y or 1 to 1 and everything else to 0.
* The concentration rate is the percent of the file that is recoded to 1. If the concentration rate is greater than *concrate* or less than (1-*concrate*) then the variable is discarded.

Nominal variables:

* If the length of the variable name plus the length of prefix is greater than 32 then the variable is discarded because the recoded variable name will be too long. If you have requested the recoding method of BINARY then the limit is 29.
* The concentration rate is the highest percent of the file that is in any single value. If the concentration rate is greater than *concrate* then the variable is discarded.
* If *valcnt* is greater than 0 and the number of unique values for a variable is greater than *valcnt* the variable is discarded.
* The file is summarized by value and sorted by mean dependent variable. The data is then collapsed to bins with at least the minimum number of records in each. The minimum number of records is calculated as the maximum of *minbinnc* and (*minbinnp* x the total number of observations).
* Bins are further collapsed based on a t-test of the difference in mean dependent variable between bins. If *bonfer* is set to N, the significance level for the t-test is *talpha*. If set to Y, the significance level is *talpha* / (number of bins – 1).

Ordinal variables:

* If you have character variables in your ordinal variable list the code will throw an error and stop.
* If the length of the variable name plus the length of prefix is greater than 32 then the variable is discarded because the recoded variable name will be too long. If you have requested transforms then the limit is 29.
* The concentration rate is the highest percent of the file that is in any single value. If the concentration rate is greater than *concrate* then the variable is discarded.
* The minimum value for the dependent variable on the non-missing portion of the file will be compared to the maximum. If they are the same, the variable will be discarded since there is no variance in the dependent variable.
* If *transformationO* is set to Y, transforms of the raw variable will be created.
* Missing values will be assigned a value based in *impmethodO*.
* If *stdmethodO* is not set to NO, then the data will be standardized using the method specified in *stdmethodO*.
* The best form of the variable will be selected using a proc logistic or proc regression depending on whether the dependent variable is binary or not. If the p-value for the test statistic (chi-square test for binary, f test for continuous) is greater then *pvalue*, then the variable is discarded as not predictive.

Continuous variables:

* If you have character variables in your ordinal variable list the code will throw an error and stop.
* If the length of the variable name plus the length of prefix is greater than 32 then the variable is discarded because the recoded variable name will be too long. If you have requested transforms then the limit is 29.
* The value for the variable at the *p\_lo* percentile will be compared to the value at the *p\_hi* percentile. If they are the same, then the variable is discarded as having a constant value.
* The minimum value for the dependent variable on the non-missing portion of the file will be compared to the maximum. If they are the same, the variable will be discarded since there is no variance in the dependent variable.
* If *transformationC* is set to Y, transforms of the raw variable will be created.
* Missing values will be assigned a value based in *impmethodC*.
* If *stdmethodC* is not set to NO, then the data will be standardized using the method specified in *stdmethodC*.
* The best form of the variable will be selected using a proc logistic or proc regression depending on whether the dependent variable is binary or not. If the p-value for the test statistic (chi-square test for binary, f test for continuous) is greater then *pvalue*, then the variable is discarded as not predictive.

A new dataset, **CE2\_Recoded**, will be written to the output library. It will contain the variables listed in *keep\_list* as well as all of the new recoded variables.

Once all of the recoding is done, the following reports will be written out: EDA, Correlation, and if profiling is set to Y, Profile. There is an Excel tool to read and format the profiling report. It is called CE\_Profiling.xlsm. Simply open the tool, enable macros, click the "Upload Data" button and select the file just created by the SAS process.

You will also have text files for the recodes for each variable type.

\*\*\* 3.Variable reduction and ranking \*\*\*;

%let LogFile = "&path\_output/03\_Var\_Redu\_Log\_File.log";

%let LstFile = "&path\_output/03\_Var\_Redu\_List\_File.lst";

**proc** **printto** log=&LogFile new; **run**;

**proc** **printto** print=&LstFile new; **run**;

%***CE\_Var\_Redu***(insdn=out.CE2\_Recoded);

**proc** **printto**;

**run**;

This block of code will run the third step of the CE process – variable reduction. The purpose of this code is to reduce the number of candidate variables based on various statistical tests before building a model. This is done to reduce the amount of resources and length of time needed to build a model. It does not use the test portion of the file.

There are seven evaluation methods that you turn on or off with a control macro variable:

* Univariate regression – evaluates each variable in isolation by building a logistic (binary dependent) or linear regression model with a single independent variable. It is turned on by setting *univ\_reg* to Y. The p-value of the appropriate test statistic (binary is chi-square, otherwise f test) is captured. Variables with a p-value less than or equal to *maxpuni* pass this test.
* Correlation – evaluates each variable by calculating the correlation with the dependent variable. It is turned on by setting *correlation* to Y. Variables with an unsigned correlation greater than or equal to *corrcut* pass this test.
* Principal components – evaluates variables using principal component analysis. It is turned on by setting *principal* to Y. In the first pass, the variables are broken into groups to make the analysis more manageable. *nprin* components are created for each group. The variables then have the correlation with each component calculated and the maximum is kept. All variables with unsigned correlation greater than or equal to *minprin* are then combined and passed through the logic again. All variables in the second pass with unsigned correlation greater than or equal to *minprin* pass this test.
* Cluster – evaluates variables using cluster analysis. It is turned on by setting *cluster* to Y. In the first pass, the variables are broken into groups to make the analysis more manageable. *maxc* clusters are created for each group. All variables with r-square ratio less than or equal to *maxratio* are then combined and passed through the logic again. . All variables in the second pass with r-square ratio less than or equal to *maxratio* pass this test.
* Linear regression – evaluates variables in combination by building a linear regression model using forward selection and significance level to enter of *alphareg*. It is turned on by setting *regression* to Y. In the first pass, the variables are broken into groups to make the analysis more manageable. A model is built for each group. All variables that enter a model are then combined and a final model is built. All variables that enter the second model pass this test.
* Logistic regression – evaluates variables in combination by building a logistic regression model using forward selection and significance level to enter of *alphalog*. It is only available if your dependent variable is binary. It is turned on by setting *logistic* to Y. In the first pass, the variables are broken into groups to make the analysis more manageable. A model is built for each group. All variables that enter a model are then combined and a final model is built. All variables that enter the second model pass this test.
* Information value – evaluates variables using its information value. It is turned on by setting *information* to Y. Each variable is summarized to bins using *decile* as the maximum number of bins to use. The variable information value is then calculated. Variables with a information value greater than or equal to *infvcut* pass this test.

A variable must pass *sources* number of tests to be kept. If *ind\_correlation* is set to Y, the kept variables will be tested for correlation between the independent variables. If any pair has an unsigned correlation greater than maxcorr, the variable with the lower correlation to the dependent variable will be removed.

The output of this step is a report of the statistics of all variables and a text file listing the reduced list of variables.

\*\*\* 4.Model selection and tuning \*\*\*;

%let LogFile = "&path\_output/04\_Model\_Selection\_Log\_File.log";

%let LstFile = "&path\_output/04\_Model\_Selection\_List\_File.lst";

**proc** **printto** log=&LogFile new; **run**;

**proc** **printto** print=&LstFile new; **run**;

%inc "&path\_output/CE3\_Varlist\_redu.txt";

%***CE\_Model\_Val***(out.CE2\_Recoded, &varlist\_redu);

**proc** **printto**;

**run**;

This block of code will run the fourth step of the CE process – model selection and tuning. It runs the process on the model portion of the file, then the validation portion of the file, then compares the variables selected by each.

1. Create the list of independent variables to use by using *varlist\_redu* (from step 3), *include\_list*, *start\_list*, and *exclude\_list*.
2. Create a regression model (logistic if a binary dependent, otherwise linear) using forward selection.
3. Create a regression model (logistic if a binary dependent, otherwise linear) using backward selection.
4. Combine all variables selected by either model into the candidate variable list.
5. Calculate statistics for the full model using all candidate variables.
6. Calculate statistics for a series of models using all but one of the candidate variables in each model. As an example, if you have 10 candidate models the first model would use variables 2 through 10, the second variables 1 and 3 through 10, etc.
7. Subtract the statistics for the subset models from the statistics for the full model. This will show what the impact of each individual variable is.
8. Accumulate all of the statistics into a dataset.

Once this process has been run on both portions of the file, the next step is variable tuning. This will select variables from each portion based on *criteria*, *threshold* and *minimp*. The two lists will be joined based on *SQL\_join*.

The output of this step is a report of the statistics for the models built, a text file listing the selected variables and, if *graph\_plot* is set to Y, a PDF report showing basic information about the selected variables.

\*\*\* 5.Final Model build and validation on test sample \*\*\*;

%let LogFile = "&path\_output/05\_Final\_Model\_Fit\_Validation\_Log\_File.log";

%let LstFile = "&path\_output/05\_Final\_Model\_Fit\_Validation\_LST\_File.lst";

**proc** **printto** log=&LogFile new; **run**;

**proc** **printto** print=&LstFile new; **run**;

%inc "&path\_output/CE4\_Varlist\_Final.txt";

%***CE\_Model\_Lift***(insdn=out.CE2\_Recoded, varlist=&varlist\_final)

**proc** **printto**;

**run**;

This block of code will run the fifth step of the CE process – final model selection.

1. A model will be built using everything but the test portion of the file – logistic if a binary dependent, otherwise linear. The selection method is controlled by *method*.
2. Statistics will be calculated in the same way as Step 4.
3. Gains table are created.
4. Final variables are profiled.
5. Report, final variable list and scoring equation are written out.