LOGISTIC REGRESSION (SCORECARD)

Linear models can also be used to gain insights into risky transaction scenarios. We aim to predict whether a transaction is fraudulent or not using a probabilistic binary classification model. In this case, logistic regression emerges as a suitable model for our rule extraction task due to its straightforward operation. For this model, we cannot directly obtain rules in the same manner as with decision trees, as logistic regression is not based on explicit rules or logical expressions to create a tree-like structure. However, we can still utilize the coefficients generated by the model to identify the most influential variables on the prediction. This helps us to determine which conditions to include or not in the decision rules.

Before that, we recommend grouping certain values for each variable using the Interactive Grouping node of SAS VDMML (Fig. 6). This allows grouping together values within each variable that have similar levels of risk. By "level of risk," we mean the average fraud probability obtained from the training data. For interval variables, Interactive Grouping first performs binning of values using one of the two following methods:

- The **quantile method**, which involves creating N ordered groups with roughly similar frequency between each group. This divides a distribution of data into multiple groups approximately populated with a similar number of observations.
- The **bucket method**, which consists in generating N groups by dividing the data into equidistant intervals determined by the range between the maximum and minimum values.

For the experiment, we decided to use the quantile binning method with up to 20 bins in order to obtain groups formed from continuous intervals. It is also possible to perform a variable selection process that helps filter out any variables with very low information content. We chose to use the Information Value criterion, setting a threshold of 0.1. This threshold is relatively low and thus permissive, as we aim to extract groups of values from a sufficient number of variables for the formation of our rules. However, it is possible to increase this threshold, especially in cases where a higher number of variables are available. We will discuss the method for calculating Information Value in more detail later on.

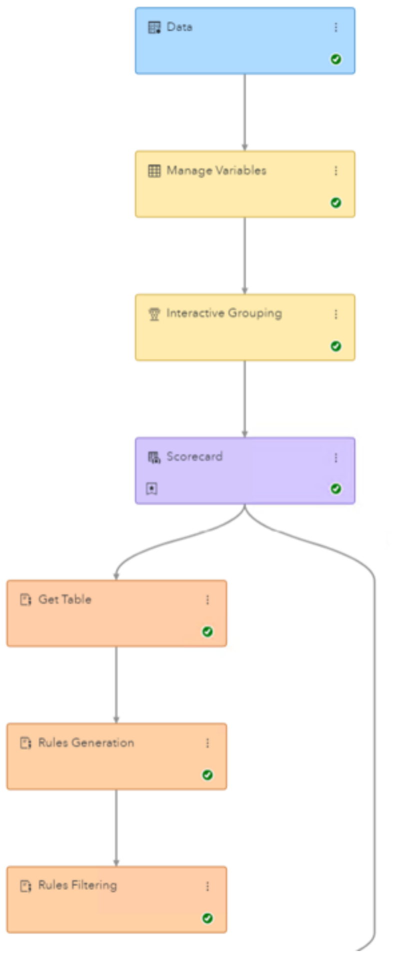


Fig.6., Pipeline for Scorecard / Logistic Regression.

Once we have defined the grouping of continuous values using the quantile method, a decision tree is applied to the data to create groups with associated risk levels. The groups are built based on entropy reduction and the maximum number of groups per variable (leaves per node in the decision tree) is set to 5. Moreover, we apply a constrained grouping method for the interval and ordinal values. It ensures to have a monotonically decreasing Weight of Evidence (Equation 1.) when the event rate for the group is monotonically increasing (e.g., Table 6.).

$$WoE_{attribute} = ln\frac{p_{attribute}^{non-event}}{p_{attribute}^{event}}$$

Equation 1., Weight of Evidence.

Variable	Group	Label	Event Count	Non-Event Count	Weight of Evidence	Group Event Rate	Event Rate
ucm_pos	1	1, 2, 81	3 106	3 012	-2,0525	0,5077	0,8762
ucm_pos	2	5, 80, 90	197	6 755	1,5131	0,0283	0,0556
ucm_pos	3	0, 7, 91, _MISSING_, _UNKNOWN_	242	17 004	2,2305	0,014	0,0683

Table 6., Groups for variable `ucm_pos` with monotonically decreasing WoE when increasing Group Event Rate.

Information Value (Equation 2.) can be to assess the predictive power of characteristics. This criterion is then used to perform the variable selection if needed. Usually, a predictor with an IV that is above 0.1 is considered a weakly predictive and can be dropped. IV is a weighted sum of the different WoE for each attribute of a variable.

$$Information\ Value = \sum_{i=1}^{m} (P(attribute_i \mid non - event) - P(attribute_i \mid event)) * WoE_i$$

Equation 2., Information Value.

The model retains 14 variables with an IV greater than 0.1. Then, the 14 selected variables are used in the form of their WoE as inputs for a logistic regression model. The logistic regression model is based on the principle of conditional probability (Bayes' theorem). It involves finding $P(Y = y_k \mid X) = \frac{P(Y= y_k)*P(X \mid Y= y_k)}{P(X)}$. The logistic function returns a value between 0 and 1 giving the probability of an event. We use logistic regression solely to calculate the coefficients for each variable in the model, and not for making direct predictions.

These coefficients represent the change in the log-odds of the target for one unit change in the corresponding feature (all other features remain constant). We can interpret the direction of the relationship between the variable and the probability of positive with the sign of the coefficient. A positive sign indicates a positive relationship while a negative sign indicates a negative relation.

The sign of the coefficient (positive or negative) indicates the direction of the relationship between the feature and the probability of the positive class. A positive coefficient means that an increase in the feature's value is associated with an increase in the odds of the positive class. By setting a significance level of 0.1 for the logistic regression model, we are able to further reduce the number of features to be considered in the model. This reduces the count from 14 variables obtained during the grouping with decision tree to 9 variables in the logistic regression.

By taking the square root of the coefficients squared, we obtain the absolute coefficients, which provide us with an indication of the strength of the influence of different features on the target variable. In other words, we can identify variables with values that strongly influence the risk of fraud. We can now filter the variables to be retained for constructing the decision rules by removing variables with relatively weak influence in the model. To achieve this, we decide to set a threshold of 0.2 for the acceptability of the variable based on its absolute coefficient value. The variables highlighted in blue (Table 7.) are kept while features in red will not be included in the rules.

Variable	Absolute
avg_val_day_mcc	0.27932
avg_vol_day	0.37209
card_present	0.33127
hct_mer_mcc	0.20884
hct_term_cntry_code	0.54615
prev_tran_dt	0.23216
rqo_tran_time	0.16967
tca_mod_amt	0.14177
ucm_pos	0.86233

Table 7., Variables absolute coefficients in logistic regression.

In order to build the decision rules, we assign each retained variable from the selection based on their absolute coefficients to the riskiest group (having the highest fraud rate) obtained during the grouping phase with the decision tree (Table 8.).

Variable	Values
avg_val_day_mcc	avg_val_day_mcc < 25.84
avg_vol_day	6 <= avg_vol_day
card_present	0
hct_mer_mcc	0, 1313, 1319, 1324, 1328, 1333, 1419, 1430, 1463, 1498, 1505, 2013, 2503, 2732, 3357, 4511, 4789, 4811, 4812, 4816, 5311, 5732, 5942, 5994, 5999, 7011, 7273, 7311, 7513, 7988, 9999
hct_term_cntry_code	112, 124, 156, 170, 192, 196, 203, 214, 246, 292, 344, 352, 356, 36, 372, 376, 392, 410, 442, 462, 504, 528, 554, 578, 604, 702, 752, 76, 792, 818
prev_tran_dt	prev_tran_dt < 833
ucm_pos	1, 2, 81

Table 8., Values from the riskiest groups for each selected variable.

In order to retrieve the output table from the node, including the coefficients, and manipulate it to create decision rules, we can use the following code (Code 5). This code allows us to save an output table from a previous node before the current node. We aim to locally save the "`scorecard`" table.

```
%let node_guid=;

data _null_;
   set &dm_predecessors end=eof;
   if eof then call symputx('node_guid', guid);
run;

data _null_;
   folder = dcreate(strip(symget('node_guid')), "&dmcas_workpath");
run;

libname nodelib "&dmcas_workpath.&dm_dsep.&node_guid";

%dmcas_fetchDataset(&node_guid, &dmcas_workpath.&dm_dsep.&node_guid, scorecard);

proc print data=nodelib.scorecard;
run;

data '/home/sasdemo/casuser/scorecard/table';
    set nodelib.scorecard;
run;
```

Code 5., Retrieve table from a predecessor node.

Then, we proceed to use an algorithm (Code 6.) to generate all possible combinations (Equation 3.) of these 7 variables to form rules containing from 1 to a maximum of 7 conditions.

$$\sum {}_{n}C_{r} = \sum_{i=1}^{N} \left(\frac{N!}{(N - r_i)!\, r_i!} \right)$$

where N is the number of remaining variables after the selection process, and r_i is the number of variables to combine at iteration i.

Equation 3., Total number of combinations.

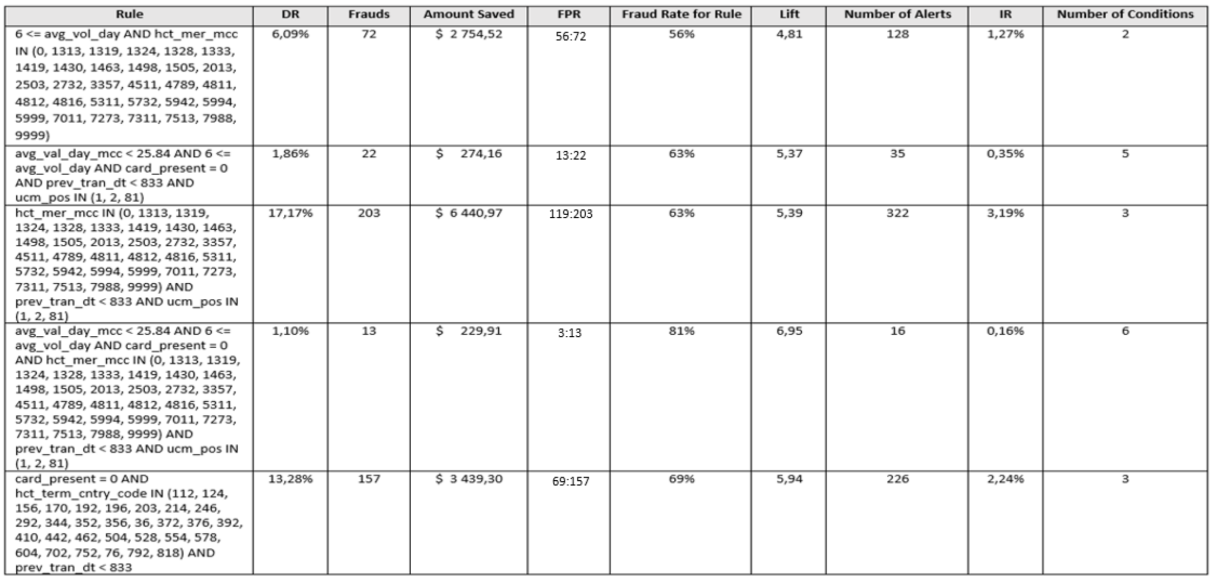
```
proc sql noprint;
    select count(*) into :N from work.conditions;
quit;

data work.combination(keep=item_:);
    array selection_[&N] $500;
    array item_[&N] $500;
    i = 1;
    do while (i <= &N);
        set work.conditions;
        selection_[i] = Condition;
        i = i + 1;
    end;

    i = 1;
    do while (i < 2**&N);
        i2 = i;
        j = 1;
        do while (j <= &N);
            if (mod(i2, 2) = 1) then item_[j] = selection_[j];
            else item_[j] = '';
            i2 = floor(i2 / 2);
            j = j + 1;
        end;
        output;
        i = i + 1;
    end;
run;
```

Code 6., Combinatory algorithm.

The above algorithm is just a part of the rule generation macro-function for this method. Creating the syntax is similar to the macro presented in the decision tree approach. Each selected condition is isolated one by one, and array object is used to temporarily store values and return them or not at each iteration in order to create unique combinations. The number of conditions to use to create the combinations is stored in the macro-variable `&N` and depends on the number of variables retained.

The number of possible combinations is equal to $2^N - 1$, and this algorithm will generate a large number of rules. For instance, with our 7 variables, we can produce 127 rules. Some of them may not be effective in detecting fraud, so we remove rules that have only one condition. These single-condition rules could significantly increase the number of false positives.

The main limitation of this approach is that the rules are focused only on few attributes which may create a large number of rules that overlap.



Rule	DR	Frauds	Amount Saved	FPR	Fraud Rate for Rule	Lift	Number of Alerts	IR	Number of Conditions
6 <= avg_vol_day AND hct_mer_mcc IN (0, 1313, 1319, 1324, 1328, 1333, 1419, 1430, 1463, 1498, 1505, 2013, 2503, 2732, 3357, 4511, 4789, 4811, 4812, 4816, 5311, 5732, 5942, 5994, 5999, 7011, 7273, 7311, 7513, 7988, 9999)	6,09%	72	$ 2 754,52	56:72	56%	4,81	128	1,27%	2
avg_val_day_mcc < 25.84 AND 6 <= avg_vol_day AND card_present = 0 AND prev_tran_dt < 833 AND ucm_pos IN (1, 2, 81)	1,86%	22	$ 274,16	13:22	63%	5,37	35	0,35%	5
hct_mer_mcc IN (0, 1313, 1319, 1324, 1328, 1333, 1419, 1430, 1463, 1498, 1505, 2013, 2503, 2732, 3357, 4511, 4789, 4811, 4812, 4816, 5311, 5732, 5942, 5994, 5999, 7011, 7273, 7311, 7513, 7988, 9999) AND prev_tran_dt < 833 AND ucm_pos IN (1, 2, 81)	17,17%	203	$ 6 440,97	119:203	63%	5,39	322	3,19%	3
avg_val_day_mcc < 25.84 AND 6 <= avg_vol_day AND card_present = 0 AND hct_mer_mcc IN (0, 1313, 1319, 1324, 1328, 1333, 1419, 1430, 1463, 1498, 1505, 2013, 2503, 2732, 3357, 4511, 4789, 4811, 4812, 4816, 5311, 5732, 5942, 5994, 5999, 7011, 7273, 7311, 7513, 7988, 9999) AND prev_tran_dt < 833 AND ucm_pos IN (1, 2, 81)	1,10%	13	$ 229,91	3:13	81%	6,95	16	0,16%	6
card_present = 0 AND hct_term_cntry_code IN (112, 124, 156, 170, 192, 196, 203, 214, 246, 292, 344, 352, 356, 36, 372, 376, 392, 410, 442, 462, 504, 528, 554, 578, 604, 702, 752, 76, 792, 818) AND prev_tran_dt < 833	13,28%	157	$ 3 439,30	69:157	69%	5,94	226	2,24%	3

Table 9., Sample of rules generated with Scorecard - Logistic Regression.