Paper PO-88

Trash to Treasures: Salvaging Variables of Extremely Low Coverage for Modeling

Alec Zhixiao Lin, PayPal Credit, Timonium, MD

ABSTRACT

Variables with extremely rare occurrences either exhibit very low Information Values (IV) in scorecard development or fail to be selected by a regression model, and hence are usually discarded at the stage of data cleaning. However, some of these variables could contain valuable information and are worth retaining. We can aggregate different events of rare occurrences into a composite variable to be used in a subsequent regression or analysis. This paper introduces a SAS® macro that can help to discover and salvage these variables in hope of turning them into potentially useful predictors.

INTRODUCTION

Let's start with a historical anecdote. In late nineteenth century, tens of thousands of Chinese came to the United States to work on the transcontinental railroad and later to catch the gold rush in California. In the new world they invented Chop Suey, a stir-fry dish using miscellaneous leftovers of chicken, meat and vegetables. Each ingredient used is of too small a portion to sustain a main dish, but a combination of these scraps and pieces can turn into a delicious treat if cooked properly.

Chop Suey was invented in a time of prevalent poverty and frugality, but the practice of maximizing the use of all materials at hand sheds some valuable insights to data mining in current days. In the realm of big data, it is not uncommon that a great number of raw variables show a low coverage. It usually suggests that observations show very low occurrences for such activities. These variables are often considered as "inconvenient nuisances" and discarded in the early stage of data cleaning. But is it possible to extract some useful information from these variables? For example, debit card holders using express checkout cancel online purchases of less than $10 quite infrequently and might not attract much attention, but they could be good prospects for credit offering because cancelations of low-value transactions suggest need for cash. If multiple segments of such customers are found and bundled to form a composite variable, they can become noticeable to modelers.

Table 1 uses an example of scorecard development to illustrate how individual variables of extremely low coverage or occurrences could easily escape the attention of modelers.

Variable 1

Value Status	count	% count	# Bad	% Bad Rate	# Good	Dist Good	Dist Bad	WOE	for IV
with occurrences	12	0.20%	6	50.00%	6	0.11%	1.48%	-262.52	0.0360
no occurrences	6000	99.80%	400	6.67%	5600	99.89%	98.52%	1.38	0.0002
Total	6012	100.00%	406	6.75%	5606	100.00%	100.00%		
								Information Value:	**0.0362**

Variable 2

Value Status	count	% count	# Bad	% Bad Rate	# Good	Dist Good	Dist Bad	WOE	for IV
with occurrences	22	0.37%	8	36.36%	14	0.25%	1.97%	-206.56	0.0355
no occurrences	5990	99.63%	398	6.64%	5592	99.75%	98.03%	1.74	0.0003
Total	6012	100.00%	406	6.75%	5606	100.00%	100.00%		
								Information Value:	**0.0358**

Variable 3

Value Status	count	% count	# Bad	% Bad Rate	# Good	Dist Good	Dist Bad	WOE	for IV
with occurrences	30	0.50%	10	33.33%	20	0.36%	2.46%	-193.21	0.0407
no occurrences	5982	99.50%	396	6.62%	5586	99.64%	97.54%	2.14	0.0005
Total	6012	100.00%	406	6.75%	5606	100.00%	100.00%		
								Information Value:	**0.0411**

Some overlapping between three behaviors is assumed.

Table 1. IVs for Separate Variables

For assessing the predictive power based on Information Value (IV), Siddiqi provides the following rule of thumb:

< 0.02: unpredictive
0.02 to 0.1: weak
0.1 to 0.3: medium
0.3 to 0.5: strong
> 0.5: suspicious

Although customers with any of the above activities are four to six times more likely to become "bad" than others, these three variables show very low Information Value and hence hard to be noticed by modelers or to be treated as dummy variables in a regression. Is there an easy way to salvage and turn these variables into useful predictors in an effort to maximize the utilization of data information?

THE SALVAGING

Although the risk-prone variables mentioned in the previous section do not shine individually, Table 2 shows how the IV increases when three are combined into one.

3 Variables combined

Value Status	count	% count	# Bad	% Bad Rate	# Good	Dist Good	Dist Bad	WOE	for IV
with occurrences	48	0.80%	20	41.67%	28	0.50%	4.93%	-228.88	0.1013
no occurrences	5964	99.20%	386	6.47%	5578	99.50%	95.07%	4.55	0.0020
Total	6012	100.00%	406	6.75%	5606	100.00%	100.00%		
								Information Value:	**0.1033**

Table 2. IV for A Composite Variable

The coverage by the composite variable now increases to 0.8%, and IV increases to 0.1033. The composite variable is much easier to be noticed.

The SAS program in the appendix provides a simple process that will help modelers to identify variables of extremely low activities for a possible use. Users only need to make needed changes to the beginning part of the code:

```
%let inpdata=libname.datasetname;                          /* dataset to examine */
%let inclzero=0;     /* consider 0 the same as missing. Set inclzero=. otherwise */
%let target=y;                                              /* target variable Y */
%let varall=v1 v2 v3 …;            /* list of independent variables(x) to screen */
```

THE OUTPUT

The SAS macro will generate a summary table (named as 'rankcontrast' in the program and labeled as "Bad Rate Ratio" in the summary below) that ranks performance ratio between with-coverage group and no-coverage group in a descending order.

Variable	Bad Rate (no occurrence)	Bad Rate (with occurrence)	# Records (no occurrence)	# Records (with occurrence)	Bad Rate Ratio	Coverage
x22	5.39%	40.86%	143,029	793	7.5776	0.55138%
x3	5.46%	41.30%	143,299	523	7.5681	0.36364%
x15	5.46%	40.56%	143,319	503	7.4215	0.34974%
x10	5.57%	41.18%	143,754	68	7.3917	0.04728%
x11	5.50%	40.46%	143,476	346	7.3523	0.24058%
x15	5.43%	35.46%	143,055	767	6.5342	0.53330%
x17	5.44%	34.73%	143,108	714	6.3825	0.49645%
x33	5.43%	33.50%	143,034	788	6.1657	0.54790%
x67	5.53%	34.04%	143,540	282	6.1542	0.19608%
x9	5.57%	34.04%	143,728	94	6.1130	0.06536%
x47	5.58%	34.04%	143,775	47	6.1028	0.03268%
X12	5.32%	30.77%	142,288	1,534	5.7881	1.06660%

Table 3. SAS Output

(*Only partial summary is pasted above)*

A typical composite variable can be created in the following way:

```
if sum(X22, X3, X15, ……, Xn) not in (0, .) then dummy_var=1; else dummy_var=0;
```

The dummy variable for combined behaviors can be used in a subsequent scorecard development or regression.

Three observations are worth sharing:

- The more raw variables bundled, the higher the incremental Information Value for the composite variable.
- The less overlapping between behaviors, the higher the incremental Information Value for the composite variable.
- If a variable has a good coverage and shows a good contrast between with-coverage group and no-coverage group, one can considering using it a dummy variable by itself.

While gathering top raw variables, modelers can also do the following:

- Apply business knowledge to create multiple composite dummy variables by bundling raw attributes of similar behaviors together.
- Examine the bottom part of the table as well. For variables at the bottom, with-coverage group shows a much lower performance outcome than no-coverage group.
- Create an ordinal or class variable to differentiate different levels of activity. Since some observations might appear in multiple categories, we can use the following example:

```
if sum(X1, X2, X3, ……, X10) not in (0, .) then dummy_var=2;
else if sum(X11, X12, ..., X18) not in (0, .) then dummy_var=1;
else dummy_var=0;
```

Even though the paper uses a binary dependent variable as an example, the method is equally applicable to continuous dependent variables such as sales, revenue, loss amount, etc. In the case of polynomial outcomes, one can process the same list of variables for different outcomes to gain insights.

CONCLUSION

Instead of being “inconvenient nuisances”, variables of extremely low coverage can be salvaged and turn into useful predictors. This helps modelers to extract more business intelligence and hence to maximize the utilization of data. This paper hopes to provide some assistance in this effort.

ACKNOWLEDGEMENTS

I would like to thank Credit Modeling team of PayPal Credit led by Shawn Benner for their inspiration and support for my work.

REFERENCES

Siddiqi, Naeem (2006) “Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring”, SAS Institute, pp 79-83.

CONTACT INFORMATION

Alec Zhixiao Lin
Principal Statistical Analyst
PayPal Credit
9690 Deereco Road
Timonium, MD 21093
Phone: 703-593-4290
Email: alecindc@gmail.com
Web: www.linkedin.com/pub/alec-zhixiao-lin/25/708/261/

APPENDIX

The following is the entire SAS program. As many users regularly encounter the issue of insufficient memory when processing large data sets, the macro automatically partitions a data set into 10 parts and processes each part separately. Therefore it requires the data set to have at least 10 variables. If your data set has less than ten

variables, create some temp variables, such as xtemp1=ranuni(123), xtemp2=ranuni(34), etc to make up to 10 before running the program.

```
%let inpdata=libname.datasetname;                          /* dataset to examine */
%let inclzero=0;     /* consider 0 the same as missing. Set inclzero=. otherwise */
%let target=y;                                              /* target variable Y */
%let varall=v1 v2 v3 …;                                 /* list of X's to screen */

data inpdata; set &inpdata(keep=&target &varall); run;
proc contents data=inpdata(keep=&varall) out=checkvar noprint; run;
data varcnt; set checkvar(keep=name); varcnt+1; run;
proc univariate data=varcnt noprint;
      var varcnt;
     output out=pctscore pctlpts=0 10 20 30 40 50 60 70 80 90 100
          pctlpre=pct_; run;

data _null_;
set pctscore;
call symputx('start1', 1);
call symputx('end1', int(pct_10)-1);
call symputx('start2', int(pct_10));
call symputx('end2', int(pct_20)-1);
call symputx('start3', int(pct_20));
call symputx('end3', int(pct_30)-1);
call symputx('start4', int(pct_30));
call symputx('end4', int(pct_40)-1);
call symputx('start5', int(pct_40));
call symputx('end5', int(pct_50)-1);
call symputx('start6', int(pct_50));
call symputx('end6', int(pct_60)-1);
call symputx('start7', int(pct_60));
call symputx('end7', int(pct_70)-1);
call symputx('start8', int(pct_70));
call symputx('end8', int(pct_80)-1);
call symputx('start9', int(pct_80));
call symputx('end9', int(pct_90)-1);
call symputx('start10', int(pct_90));
call symputx('end10', pct_100);
run;

proc sql noprint; select name into :varmore separated by ' ' from varcnt; quit;
proc sql; create table vcnt as select count(*) as vcnt from varcnt; quit;
data _null_; set vcnt; call symputx('vmcnt', vcnt); run;
proc sql noprint; select name into :v1-:v&vmcnt from varcnt; quit;

proc sql noprint;
select max(varcnt), compress('&x'||put(varcnt, 10.))
into :varcount, :tempvar separated by ' '
from varcnt
order by varcnt;
quit;

proc sql noprint; select name into :x1-:x&end10 from varcnt; quit;

%macro stkorig;
%do i=1 %to &vmcnt;
data temp2;
length name $32.;
set inpdata(keep=&&v&i &target);
name="&&v&i";
format name $32.;
run;

proc sql;
create table tempsum(rename=(withtrans=&&v&i)) as select
```

```
case when &&v&i in (0, .) then 0 else 1 end as withtrans, avg(&target) as yrate,
count(*) as groupsize
from temp2(keep=&&v&i &target)
group by withtrans; quit;

proc transpose data=tempsum out=temp_trans; run;

data transy(rename=(col1=y0 col2=y1))
     transcount(rename=(col1=count0 col2=count1));
set temp_trans;

if _name_="&&v&i" then delete;
if _name_="yrate" then output transy;
if _name_="groupsize" then output transcount;
drop _name_;
run;

data v&i;
retain varname y0 y1 count0 count1;
if _N_ = 1 then set transy;
set transcount;
 varname="&&v&i";
 ycontrast=y1/y0;
 coverage=count1/sum(count0, count1);
run ;
%end;
%mend;
%stkorig;

data rankcontrast;
length varname $32.;
set v1-v&vmcnt;
if count0 > 0 and count1 > 0;

label y1="Bad Rate (with-coverage)";
label y0="Bad Rate (no-coverage)";
label count0="# Records (with-coverage)";
label count1="# Records (no-coverage)";
label ycontrast="Compare Bad Rate";
run;

proc sort data=rankcontrast; by descending ycontrast; run;
proc print data=rankcontrast; run;
```