BINNING PREDICTORS FOR LOGISTIC REGRESSION

BRUCE LUND
Statistical Trainer, Novi, MI

Blund_data@mi.rr.com … or
Blund.data@gmail.com

Introduction to Binning Terminology

Let **X** be a **classification predictor** … character or numeric … there are three types (… NOD):

- **N**ominal {blue, green, brown}
- **O**rdinal: {poor, fair, good}
- **D**iscrete Numeric {0, 1, 2, 3}

- Number of levels of **X** is **L.** Generally < 10 but might be as high as 20 … see TABLE at bottom
- In the TABLE: the Levels of **X** are A, B, C, D so that **L** = 4
- For each observation of **X** there is a value of variable **Y** called the Target … with levels 0 and 1
  We will say the **Y**=1 is an "event" and **Y**=0 is a "non-event"
- The "event-rate" at a level of **X** is (number of events) / (total cases) … for that level
- The "odds" at a level of **X** is (number of events) / (number of non-events) … for that level
- … See TABLE for event rates: 2/4 1/2 1/4 3/5 … odds 2/2 1/1 1/3 3/2

X	Non-Events	Events	Total	Event Rate	Odds
A	2	2	4	2/4	2/2
B	1	1	2	1/2	1/1
C	3	1	4	1/4	1/3
D	2	3	5	3/5	3/2

Event Rate =2/4 … Odds =2/2

next

Description and Goal of Binning

next

- Here is some Binning: Suppose X has levels A, B, C, D.
  - After first step, the levels of binned X might become {A,B}, {C}, {D}
  - Next: bin together {A,B} and {D} from Step 1 to get {A,B,D}, {C}
- Goal of binning a predictor X for a binary target Y is to:
  - Simplify X by combining some of the levels,
  - while revealing an underlying relationship between X and event-rate of the target,
  - and while maintaining most of the strength of X to predict Y.

To Do's:

1. Define measures of predictive strength
2. Present algorithms for binning so that predictive strength is optimized.

- As Binning proceeds we call the current predictor: "X_BIN".

After binning is completed, this is how we use X_BIN in the Logistic Model … goes in CLASS

PROC LOGISTIC; CLASS X_BIN <other X>; MODEL Y = X_BIN <other X>;

An alternative is coding X_BIN as a weight-of-evidence predictor.
This is not discussed today.

Introduction to Binning – More Terminology

- There are **two alternatives** when binning. Depends on whether X has meaningful ordering.

  For example: If X has 4 *ordered* levels A, B, C, D

  ORDERED BINNING: levels within a bin are adjacent (no gaps) with respect to ordering of X
  - The 2-bin solutions with ordered bins are {A} {B C D} and {A B} {C, D} and {A B C} {D}.

  UNRESTRICTED BINNING: levels within a bin are unrestricted with respect to ordering of X
  - A 2-bin solution which is not ordered is {A C} {B D} … also {A C D} {B}

An **ordered** bin solution is **monotonic** if the ordered bins are monotonic versus the event-rate, [or equivalently, versus the odds]

Consider a predictor X with four *ordered* levels: 1, 2, 3, 4 and Y (target) counts as shown:

X	Y=0	Y=1	3-bins for X	Event Rate	Odds
1	2	1	{1, 2}	2/5	0.67
2	1	1			
3	3	1	{3}	1/4	0.33
4	4	1	{4}	1/5	0.25

A 3-bin *ordered* solution for X is {1 2}, {3}, {4}

This binning is also a Monotonic Binning Solution because event rate is decreasing: 2/5, 1/4, 1/5

Odds are also monotonic: 0.67, 0.33, 0.25.

Measures of predictive strength for X

Here are two measures of predictive strength (and there are others)

1. Information Value (IV)
2. Entropy (or equivalently, Log-Likelihood)

See next slides

next

Two measures of strength are Information Value (IV) and Entropy

Information Value

Information Value is easily explained by giving an example.
Simply work through columns from left to right.

X	Frequencies		Col % Y=0 "b_k"	Col % Y=1 "g_k"	X_woe: Log(g_k/b_k)	g_k - b_k	IV Terms: (g_k - b_k) * Log(g_k/b_k)
	Y = 0	Y = 1					
X1	2	1	25.0%	12.5%	-0.69315	-0.125	0.08664
X2	1	1	12.5%	12.5%	0.00000	0	0.00000
X3	5	6	62.5%	75.0%	0.18232	0.125	0.02279
SUM	8	8	100%	100%		IV =	0.10943

IV Range	Interpretation
IV < 0.02	"Not Predictive"
IV in [0.02 to 0.1)	"Weak"
IV in [0.1 to 0.3)	"Medium"
IV ≥ 0.3	"Strong"

In well-known book by Naeem Siddiqi (2017, p 179), this Table is presented to give a guide for usage of IV in evaluating X_BIN.

Information Value is especially used by modelers who work on credit-risk applications.

next

Two measures of strength are Information Value (IV) and Entropy

Entropy

Assume X has been binned to create variable X_BIN. Put it in this Logistic Model.

PROC LOGISTIC; CLASS X_BIN; MODEL Y = X_BIN;

If there are k bins ($k \geq 2$), the formula for Log-Likelihood for this model is given by:

$$\text{Log-Likelihood} = \sum_{j=1}^{k} G_j * \log(G_j / (G_j + B_j)) + B_j * \log(B_j / (G_j + B_j)) \ldots = LL$$

where G_j = count of Y=1 for bin j and B_j = count of Y=0 for bin j and $1 \leq j \leq k$

Entropy with base e (not base 2) is given by this formula:

$$\text{Entropy}_e = -(\text{Log-Likelihood}) / n$$

where log-likelihood is from the Logistic Model (above) and n is the sample size.

Whenever two levels of binned X are combined (making one less bin), then entropy increases (or is same).

The goal of an algorithm with the objective function of entropy is to minimize the increase of entropy as the binning proceeds.

More entropy is bad because it means more uncertainty regarding relationship of X_BIN to Y

Equivalent to entropy is -Log-Likelihood. But we will multiply by 2 … use -2*Log(L) from now on.

next

A Binning Example

next

Y is the target (1=satisfied, 0=not satisfied) vs. two demographics DEMO1 and DEMO2

DEMO1 and DEMO2 are CLASSIFICATION, ordered but not numeric.

Counts of Customers (*)					
	DEMO2				
DEMO1	1	2	3	4	Total
B	76	189	321	102	688
C	136	287	418	152	993
D	263	451	538	219	1471
E	298	564	550	243	1655
F	290	350	265	202	1107
G	114	95	76	42	327
Total	1177	1936	2168	960	6241

Counts of Satisfied Customers (*)					
	DEMO2				
DEMO1	1	2	3	4	Total
B	41	113	193	50	397
C	109	224	292	104	729
D	208	326	384	160	1078
E	233	421	422	207	1283
F	248	275	205	181	909
G	97	74	62	33	266
Total	936	1433	1558	735	4662

Main effects model:

```
PROC LOGISTIC DATA = DEMO_SAT desc;
CLASS DEMO1 DEMO2; MODEL Y = DEMO1 DEMO2;
run;
```

Effect	DF	Chi-Square	Pr > ChiSq
DEMO1	3	12.3466	0.0063
DEMO2	5	134.4910	<.0001

(*) Hypothetical Data

A Binning Example … continued

Table shows the % of Y=1 in each cell of the DEMO1 * DEMO2 grid … color coded

- Highest percentage of satisfied (red) are lower left and lower right.
- Highest percentage of dissatisfied (blue) are upper right corner.

Color coding shows no simple pattern for finding cells with high / low levels of satisfaction.

DEMO1	DEMO2			
	1	2	3	4
B	53.9%	59.8%	60.1%	49.0%
C	80.1%	78.0%	69.9%	68.4%
D	79.1%	72.3%	71.4%	73.1%
E	78.2%	74.6%	76.7%	85.2%
F	85.5%	78.6%	77.4%	89.6%
G	85.1%	77.9%	81.6%	78.6%

Main effects model doesn't track complex patterns between the Demos and Y.

There are 24 cells in the Table.
Perhaps do this:

Define X=DEMO1 || DEMO2 (24 levels)

Then use a binning algorithm to bin X against target Y.

NOTE: X is not ordered. Is 1B < 4G ???

Need a binning algorithm !!!

next

%NOD_BIN is Macro that can bin X = DEMO1 || DEMO2

%NOD_BIN allows any two levels of X to be combined so as to maximize IV for that combine.

Starting at 24 levels of X, a pair is selected to combine so as to maximize IV. Now: 23 bins.

- This is a stepwise process. Maximize IV again to combine a pair to form 22 bins.
- Steps are repeated until there are only 2 bins.
- For each k in the range 24 ≥ k ≥ 2 the algorithm reports a k-bin solution.

Here is 2 bin solution

Some Columns not shown																								
k	IV	-2*Log L	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10	L11	L12	L13	L14	L15	L16	L17	L18	L19	L20	L21	L22	L23	L24
24	0.172	6859.8	1B	1C	1D	1E	1F	1G	2B	2C	2D	2E	2F	2G	3B	3C	3D	3E	3F	3G	4B	4C	4D	4E	4F	4G
23	0.172	6859.8	1B	1C	1D	1E	1F	1G	2B	2C	2D	2E	**2F+4G**	2G	3B	3C	3D	3E	3F	3G	4B	4C	4D	4E	4F	
22	0.172	6859.8	1B	1C	1D	1E	1F	**1G+4E**	2B	2C	2D	2E	2F+4G	2G	3B	3C	3D	3E	3F	3G	4B	4C	4D	4F		
21	0.172	6859.8	1B	1C	1D	1E	1F	1G+4E	2B	**2C+2G**	2D	2E	2F+4G	3B	3C	3D	3E	3F	3G	4B	4C	4D	4F			
			Rows 20 to 3 removed																							
2	0.088	6952.6	1B+4B+2B+3B	1C+3G+1D+2F+4G+1E+2C+2G+2E+3E+3F+2D+4D+3D+3C+4C+1F+1G+4E+4F																						

next

%NOD_BIN can lead to suboptimal solutions

- The %NOD_BIN algorithm, using IV, can lead to suboptimal solutions.

  IV is maximized when going from k to k-1.
  This precludes any sub-maximal combine.
  But a sub-maximal combine might lead later to a better IV for some smaller k.

- Alternatively, -2*Log (L) … may be minimized and again solutions can be suboptimal
- Stepwise maximization of IV method and stepwise minimization of -2*Log (L) can lead to different solutions.

Examples of the above are in a paper by Lund at SESUG 2023.

next

%NOD_BIN applied to X with target Y and freq W

```
%NOD_BIN(
DATASET = DEMO_SAT,
X = X,
TARGET = Y,
ZERO_ONE = YES,
W = W,
METHOD = IV,
MODE = A,
ORDER = D,
MISS = ,
MIN_PCT = ,
MIN_NUM = 15,
MIN_BIN = ,
MAX_BIN = ,
VERBOSE = YES,
VERBOSE2 = ,
LL_STAT = YES,
WOE = ,
ADD = ,
RUN_TITLE =
);
```

or LL

or J

Get MACRO documentation from me for description of parameters

LL_STAT = YES: Here is how this works …

Suppose there are 3 bins: X_bin1 X_bin2 X_bin3.

These bins are dummy variables in the Logistic Model:

MODEL Y = X_bin1 X_bin2 /* X_bin3 is reference */

A statistical test checks whether coefficients of X_bin1 and X_bin2 are "statistically equal".

If "statistically equal", then OK to combine X_bin1 and X_bin2.

LL_STAT gives the p-value for this test.

MIN_NUM gives the minimum count we will allow for a BIN.

If a bin count < MIN_NUM, then this bin is immediately combined via IV (… or LL)

Stopping at k=4 is good … also consider k=5 or k=6

Stopping at k=4 is good.

- First k where LL_STAT is below 0.05 … don't combine **L2** and **L4**. Coefficients are "unequal"
- IV falls greatly after k=4 (**0.1599** to 0.1406)
- Another test, not shown, supports stopping at k=4 (or k=5 or k=6) … see user documentation

k	IV	-2*Log L	L1	L2	L3	L4	L5- L24 not shown	Pr > ChiSq
	Rows 24 to 9 are not displayed							
8	0.1701	6862.3	1B+4B	1C+3G+1D+2F+4G+1E+2C+2G	1F+1G+4E	2B+3B		0.329
7	0.1691	6863.5	1B+4B	1C+3G+1D+2F+4G+1E+2C+2G	1F+1G+4E	2B+3B		0.270
6	0.1664	6866.7	1B+4B	1C+3G+1D+2F+4G+1E+2C+2G+2E+3E+3F	1F+1G+4E	2B+3B		0.077
5	0.1633	6869.2	1B+4B	1C+3G+1D+2F+4G+1E+2C+2G+2E+3E+3F	1F+1G+4E+4F	2B+3B		0.112
4	**0.1599**	6873.4	1B+4B+2B+3B	1C+3G+1D+2F+4G+1E+2C+2G+2E+3E+3F	1F+1G+4E+4F	2D+4D+3D+3C+4C		**0.040**
3	0.1406	6896.2	1B+4B+2B+3B	1C+3G+1D+2F+4G+1E+2C+2G+2E+3E+3F+ 2D+4D+3D+3C+4C	1F+1G+4E+4F			0.000
2	0.0879	6952.6	1B+4B+2B+3B	1C+3G+1D+2F+4G+1E+2C+2G+2E+3E+3F+ 2D+4D+3D+3C+4C+1F+1G+4E+4F				0.000

- -2*Log(L) for main effects model was 6900.9. But k=4 has 6873.4 … which is better !
- 4-bin solution uses 3 d.f. versus 8 d.f for the main effects model.

Color coded 4 bin solution – Sat Rate for demo1 x demo2

	DEMO2				
DEMO1	1	2	3	4	Total
B	53.9%	59.8%	60.1%	49.0%	57.7%
C	80.1%	78.0%	69.9%	68.4%	73.4%
D	79.1%	72.3%	71.4%	73.1%	73.3%
E	78.2%	74.6%	76.7%	85.2%	77.5%
F	85.5%	78.6%	77.4%	89.6%	82.1%
G	85.1%	77.9%	81.6%	78.6%	81.3%
Total	79.5%	74.0%	71.9%	76.6%	74.7%

Of course, in most cases %NOD_BIN would be applied to a single predictor … not a concatenation of predictors, as shown by this example.

next

Binning Ordinal Predictors

- X has meaningful order. Want X_BIN to be ordered to retain meaning
- Need to bin low frequency levels of X … low freqs: See X = 9 to 19
- Want to achieve parsimony (fewer d.f. after binning)
- Need to include Missing … "no bureau" has information about Y
- Could do an ad hoc X binning has follows:
  X_BIN1 = {0}, X_BIN2 = {1}, X_BIN3 = {2}, X_BIN4 = {3 through 19}
  X_BIN5 = {missing} … roughly equal sized bins
- BUT this binning does not maintain order of X vs. monotonicity of odds (see table below)

X_BIN	ODDS
1	0.914
2	**0.907**
3	1.071
4	1.294
5	1.764

In this example it is sensible to require that:
"more trade lines ➔ more defaults"

Want Binning to preserve predictive strength of X.
Either in terms of IV or -2*Log(L)

X=Num Trades Open inLast12M	Y=RiskPerformance		
	Bad	Good	Total
? (no bureau)	291	165	456
0	565	618	1183
1	569	627	1196
2	511	477	988
3	343	274	617
4	210	162	372
5	97	100	197
6	57	44	101
7	40	17	57
8	18	9	27
9	8	5	13
10	8	2	10
11	4	0	4
12	1	0	1
13	2	0	2
14	2	0	2
16	1	0	1
17	1	0	1
19	1	0	1
Total	2729	2500	5229

Home Equity Loans:
https://community.fico.com/s/explainable-machine-learning-challenge?tabset-3158a=2

%ORDINAL_BIN

%ORDINAL_BIN can do all these things:

- Consolidate low freq levels of X (see 9 through 19)
- Achieve parsimony (fewer d.f.)
- Keep ordering for Binned X (no gaps in a bin of X's)
- Include Missing … Missing has information about Y
- Find solutions with **optimal** "predictive strength" (IV, -2*LL)
- Find monotonic solutions … more Trade ➔ higher Bad rate

NOTE:

For given k > 2 there may not exist a monotonic solution:

Consider the table for X & Y. There is no monotonic 3-bin solution for this table.

X	Y=0	Y=1
1	1	4
2	5	4
3	5	3
4	1	4

next

X=Num Trades Open inLast12M	Y=RiskPerformance		
	Bad	Good	Total
? (no bureau)	291	165	456
0	565	618	1183
1	569	627	1196
2	511	477	988
3	343	274	617
4	210	162	372
5	97	100	197
6	57	44	101
7	40	17	57
8	18	9	27
9	8	5	13
10	8	2	10
11	4	0	4
12	1	0	1
13	2	0	2
14	2	0	2
16	1	0	1
17	1	0	1
19	1	0	1
Total	2729	2500	5229

%ORDINAL_BIN applied to the Home Equity Dataset

next

```
DATA TRAINx; SET HELOC_2.TRAIN;
if NumTradesOpeninLast12M = -9 then NumTradesOpeninLast12M = . ;
run;

%Ordinal_Bin(
DATASET=TRAINx,
X=NumTradesOpeninLast12M,
TARGET=RiskPerformance,
W=1,
RANKING=IV, /* ranks solutions */
ORDER=D,
MISS=MISS,
SUMMARYONLY=YES,
N_BEST=,
N_MONO=,
MIN_PCT=,
MIN_NUM=15,
MIN_BIN=,
MAX_BIN=,
NOPRINT_WOE=,
PRINT1_WOE=,
PRINT2_WOE=,
RUN_TITLE=HELOC Train,
DELETE_PRIOR=);
```

The algorithm is "Complete Enumeration".

ALL Solutions are inspected

So, the **optimal** solution (IV or -2*LL) will be found for each k

Consider case of a predictor with three levels: 1, 2, 3.

There are 2 ordered bin solutions with 2 bins: {1 2} {3} and {1} {2 3} and one solution with 3 bins: {1} {2} {3}.

%ORDINAL_BIN finds them ALL, finds which are monotonic with respect to odds of target, and computes IV and -2*LL for each bin solution.

MISS=MISS, includes Missing as part of the final binning (but not included within a bin of X levels)

MIN_NUM=15, requires that any solution reported by %ORDINAL_BIN have at least 15 observations in each bin.

%ORDINAL_BIN is restricted to X where L ≤ 20. This is discussed further on a later slide.

Results of %ORDINAL_BIN … shows many solutions

next

A promising monotonic solution occurs at k=6 & Sol_num=4

It has a strong IV of 0.0599.

This IV is higher than the IV's of the monotonic solutions which follow at k=5 through k=2.

Bins_in_sol	Sol_num	IV	minus 2LL	turns	best_rank	best_mono
10	1	0.0625	7161.3	5	*	
9	1	0.0625	7161.3	4	*	
8	1	0.0624	7161.4	4	*	
7	1	0.0621	7161.7	2	*	
6	1	0.0611	7163.1	2	*	
6	4	0.0599	7164.6	0		*
5	1	0.0598	7164.7	0	*	*
4	1	0.0580	7167.1	0	*	*
3	1	0.0516	7175.2	0	*	*
2	1	0.0393	7188.7	0	*	*

Values of IV and -2*LL include contribution from Missing.

Missing contribution to IV or -2*LL is a fixed number that does not vary with k.

BINS	Sol_num	IV	minus2LL	L1	L2	L3	L4	L5	L6
6	4	0.060	7164.64	0+1	2	3+4+5	6	7+8+9	10+11+12+13+14+16+17+19

SAS code … CLASS coding or WOE coding

SAS for bin coding

```
if NumTradesOpeninLast12M in ( 0,1 ) then NumTradesOpeninLast12M_B = 1 ;
if NumTradesOpeninLast12M in ( 2 ) then NumTradesOpeninLast12M_B = 2 ;
if NumTradesOpeninLast12M in ( 3,4,5 ) then NumTradesOpeninLast12M_B = 3 ;
if NumTradesOpeninLast12M in ( 6 ) then NumTradesOpeninLast12M_B = 4 ;
if NumTradesOpeninLast12M in ( 7,8,9 ) then NumTradesOpeninLast12M_B = 5 ;
if NumTradesOpeninLast12M in ( 10,11,12,13,14,16,17,19 ) then NumTradesOpeninLast12M_B = 6 ;
if NumTradesOpeninLast12M= . then NumTradesOpeninLast12M_B = 0 ;
```

SAS for WOE coding

```
if NumTradesOpeninLast12M in ( 0,1 ) then NumTradesOpeninLast12M_B_woe = 0.1810288345 ;
if NumTradesOpeninLast12M in ( 2 ) then NumTradesOpeninLast12M_B_woe = 0.0187914105 ;
if NumTradesOpeninLast12M in ( 3,4,5 ) then NumTradesOpeninLast12M_B_woe = -0.105193692 ;
if NumTradesOpeninLast12M in ( 6 ) then NumTradesOpeninLast12M_B_woe = -0.171217124 ;
if NumTradesOpeninLast12M in ( 7,8,9 ) then NumTradesOpeninLast12M_B_woe = -0.668023028 ;
if NumTradesOpeninLast12M in ( 10,11,12,13,14,16,17,19 ) then NumTradesOpeninLast12M_B_woe = -2.214940583 ;
if NumTradesOpeninLast12M= . then NumTradesOpeninLast12M_B_woe = -0.479733283 ;
```

%ORDINAL_BIN and "Zero-Bins"

next

X	Y		
	0 (B)	1 (G)	Total
1	5	2	7
2	6	1	7
3	4	5	9
4	**0**	**4**	**4**
Total	15	12	27

%ORDINAL_BIN does not report solutions where there is a zero-bin. A solution with a zero-bin is generally not desirable. (e.g. Logistic Model would not converge.)

A zero-bin where there is large number of non-events (or events) probably suggests there is a problem.

%ORDINAL_BIN simply skips reporting these solutions.

k	BIN contents				Action
4	{1}	{2}	{3}	{4}	Skip because B_4=0
3	{1 2}	{3}	{4}		Skip because B_4=0
3	{1}	{2 3}	{4}		Skip because B_4=0
3	{1}	{2}	{3 4}		Report this solution
2	{1 2 3}	{4}			Skip because B_4=0
2	{1 2}	{3 4}			Report this solution
2	{1}	{2 3 4}			Report this solution

Zero-bins, IV, and %NOD_BIN

%NOD_BIN cannot start stepwise binning for METHOD=IV if there is a zero-bin.
Cannot compute IV because of the zero in the Logarithm of IV formula.

[Recall this step where X_woe = $\text{Log}(g_k/b_k)$]

If %NOD_BIN cannot start, then it can't go to a second step.

Entropy can be calculated for a zero bin because 0 * Log(0) (in Entropy formula) is defined = 0

Regarding IV and a zero cell:

… One approach … add 0.5 to the zero-bin … this makes IV incorrect … issue for small samples.

A better approach is provided by %NOD_BIN.

- A little bit complicated … gives correct IV values once zero-cells are collapsed.
- Details are given in the Appendix 1 of these slides.

Does SAS/STAT have a Binning Procedure ?

No PROC in SAS/STAT “overtly” supports binning.

… but PROC HPSPLIT might be tried.

See Enterprise Miner … the Credit Scoring Add-On … it performs Binning.

HPSPLIT creates decision trees …

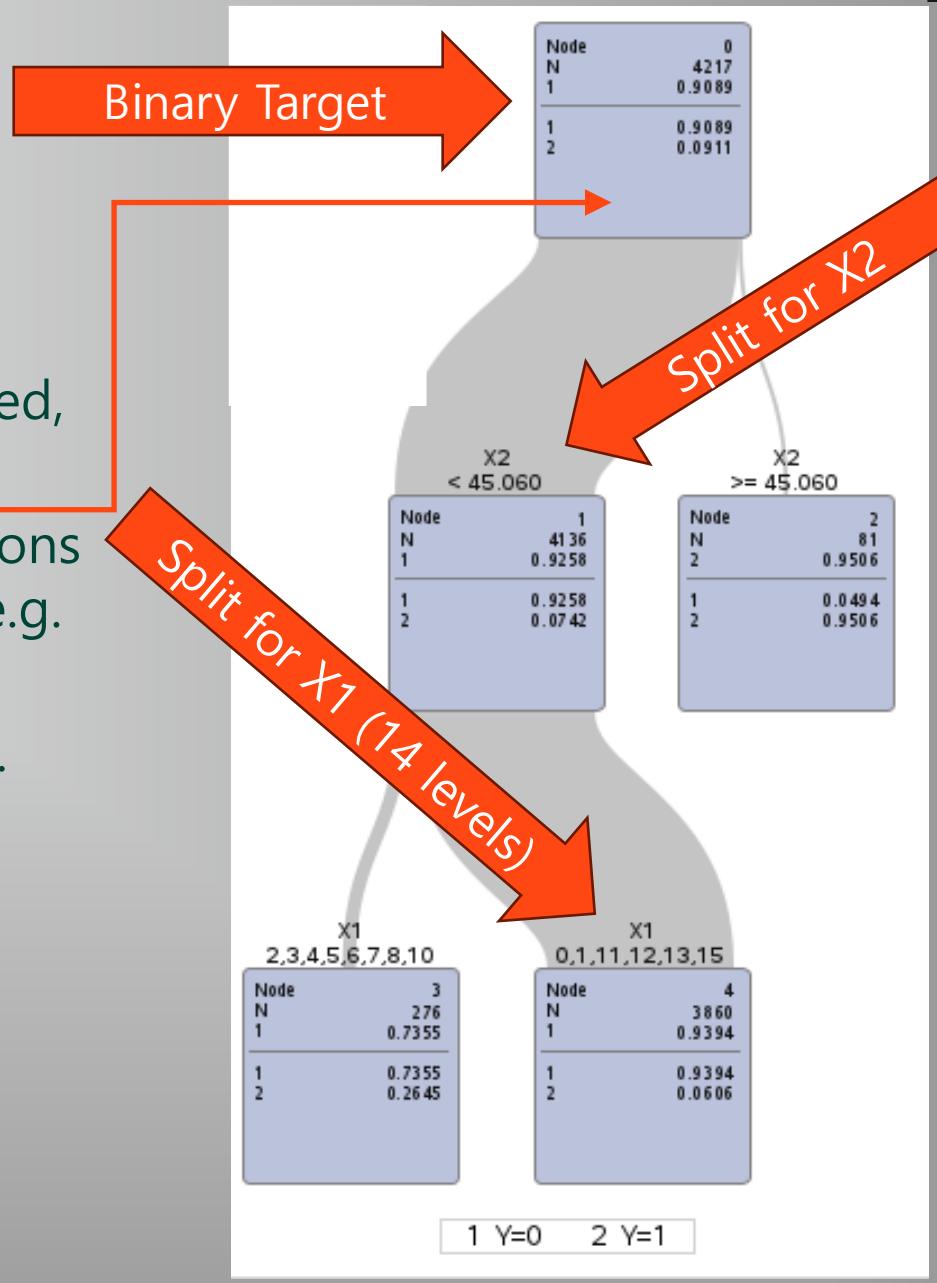
Let’s look at HPSPLIT over the next few slides to see if it can doing binning.

PROC HPBIN is not a binning procedure …
… No algorithm is applied to Target to achieve optimum binning … such as using entropy or IV

Decision Trees for Classification

- CLASSIFICATION trees: Target nominal … binary or multi-nominal (e.g. none, mild, severe … but regarded as unordered)
- X's can be nominal or ordered. In this example: X2 is ordered, X1 is nominal (14 levels)
- X's split the Dataset at NODES into non-overlapping partitions so that Target is "maximumly" dissimilar across partitions (e.g. dissimilarity by entropy).
- A Node can be split two way (most common) or multi-way. (Two-way in this example.)
- Other options for measuring dissimilarity: Gini, Chi-Square
- PROC HPSPLIT *tries* to consider each X and each allowed grouping of levels of X when doing a Split. (See SAS doc.)

  X2 (4695 levels): splits 4694, X1: splits $2^{14} - 2$



PRUNING for classification tree … a high-level discussion

PROC HPSPLIT grows a maximal tree and then relies on PRUNING to optimize Final Tree

PROC HPSPLIT provides two choices:

1. Grow full TREE on Dataset and use Cross Validation to PRUNE to minimize ERROR
2. Divide Dataset into TRAIN and VALIDATE. (See HPSPLIT PARTITION statement).
   - Full TREE is grown on TRAIN (uses only TRAIN to fit the TREE) … need large samples

ERROR is "misclassification error":

If leaf has 6 "1's" and 4 "0's", then it is classified as "1". Misclassification Error rate = 40%
Note: "Tie" (e.g. 5 "1's" and 5 "0's") is classified as "0"

Using "Cost-Complexity" a *nested* sequence of trees is found from: Full Tree to Root Node.
It is mathematically true that ERROR only needs to be computed for this nested sequence.

- If TRAIN/VALIDATE, then PRUNED true is smallest tree with minimum VALIDATE ERROR
- For Cross-Validation, the PRUNING is complicated and won't be discussed here

See Appendix 2 for a discussion of C-V Pruning with cost-complexity and a worked example.

PROC HPSPLIT doing Binning

But if only one predictor is used and target is binary, then this is BINNING.
… HPSPLIT does not bin using IV but can use ENTROPY

The Binning starts at the Root … unlike %NOD_BIN, %ORDINAL_BIN which start with "all leaves"

We have used %ORDINAL_BIN to bin Home Equity Dataset with NumTradesOpeninLast12M
Now let's compare to HPSPLIT.

```
DATA TRAINx; SET HELOC_2.TRAIN;
if NumTradesOpeninLast12M = -9 then NumTradesOpeninLast12M = .;
PROC HPSPLIT data=TRAINx
ASSIGNMISSING=BRANCH
CVMETHOD=random(10) seed=123;
CLASS RiskPerformance;
MODEL RiskPerformance = NumTradesOpeninLast12M;
 code file ="/home/blund0/HPSPLIT/Heloc_code.sas";
GROW entropy;
PRUNE costcomplexity;
run;
```

Assign MISSING to one NODE

SPLIT a NODE which gives most decrease of Entropy

With cross-validation (10) and costcomplexity algorithm, PRUNE the TREE to "optimal size"

SAS code to SCORE TRAINx for further processing

HPSPLIT Tree vs. Binning by %ORDINAL_BIN

	OUT of BOX Solution: BINS from **HPSPLIT**						
BINs →	MISSING	0-1	2	3-4	5	6	7-19
Good (G)	165	1134	511	553	97	57	86
Bad (B)	291	1245	477	436	100	44	33
Ratio B/G	1.764	1.098	0.933	0.788	**1.031**	0.772	0.384

	6 BIN solutions from **%ORDINAL_BIN**						
Best OVERALL	MISSING	0-1	2	3-4	5-6	7-9	10-19
Ratio	1.764	1.098	0.933	0.788	**0.935**	0.470	0.100
Best MONO	MISSING	0-1	2	3-5	6	7-9	10-19
Ratio	1.764	1.098	0.933	0.825	0.772	0.470	0.100

HPSPLIT: Out of BOX defaults created 6 Bins plus Missing

- Not MONOTONIC Binning … See BIN with "5"
- IV = 0.0557
- -2*Log(L) = 7167.26

(See Appendix 5 for SAS code)

%ORDINAL_BIN:

Best solution for 6 bins (by IV):

- IV = 0.0611
- -2*Log(L) = 7163.09

MONOTONIC for 6 bins

- IV = 0.0599
- -2*Log(L) = 7164.64

%ORDINAL_BIN: The OVERALL and MONOTONIC solutions are both better than Tree solution. "Complete enumeration" will always find best BINNING for ordered X (via any measure)

Comments

next

(1) PROC HPSPLIT is excellent for fitting decision trees … with many predictors and with many features not discussed today.

… with additional back-end programming, HPSPLIT might be used for binning.

(2) But in the case of binning an ordinal predictor …

%ORDINAL_BIN can't be beat … if number of levels of X ≤ 20

(See next slide regarding "20")

HPSPLIT might be competitive vs. %NOD_BIN for binning unordered X … needs study

But %NOD_BIN was much better than HPSPLIT TREE for the earlier "satisfaction" example:

(See Appendix 6 for discussion)

(3) If target is numeric 0-1 and not in CLASS, then HPSPLIT fits a REGRESSION TREE

ERROR is measured by RSS = root sum of squares (can be computed since target is numeric)

But still, REGRESSION TREE can't beat %ORDINAL_BIN for binning ordered X

%ORDINAL_BIN … why programmed for X with L ≤ 20?

next

For ordered X with L levels: Total number of ordered bin solutions across k for 2 ≤ k ≤ L is

$$2^{(L-1)} - 1$$

If L = 3 then $2^{(3-1)} - 1 = 3$ solutions. Here they are: {1 2} {3} and {1} {2 3} and {1} {2} {3}

If L = 20 then **$2^{(L-1)} - 1 = 524{,}287$**

- For %ORDINAL_BIN, the run-time doubles with each added level … $2^{(L-1)} - 1$ vs. $2^{L} - 1$
- %ORDINAL_BIN was run on a data set with 20 levels for X … run-time was good.

If L = 25 then number of solutions is 16,777,215 … %ORDINAL_BIN is not so practical.

If the number of levels L of ordinal X is over 20, then preliminary binning is needed:

Could use PROC HPBIN or PROC RANK.

Any other ideas? … see next slide !

%NOD_BIN and %ORDINAL_BIN working together

next

%NOD_BIN with specification MODE = J also finds an ordinal solution for each k.

- %NOD_BIN runs quickly as X goes from 100 levels to 2 levels (with MODE = J)
  - 4,950 trial solutions
  - 99 trials at first step, then 98, 97 … , 2, 1 = 100*99/2 = 4,950

But %NOD_BIN does not guarantee the optimal solution (unlike %ORDINAL_BIN)

%NOD_BIN in connection with %ORDINAL_BIN …

Provides an approach to binning ordinal X when L ≤ 100 (or even higher)

- Preliminary binning by %NOD_BIN provides an ordered bin solution with L = 20.
- This intermediate solution with L = 20 can then be processed by %ORDINAL_BIN.

This approach is explored later in the paper.

%MONOBIN

next

%MONOBIN performs monotonic binning for a Continuous Numeric X with no restriction on L

- WenSui Liu developed %MONOBIN
- For the SAS code see: https://statcompute.wordpress.com/2017/09/24/granular-monotonic-binning-in-sas/
- See https://statcompute.wordpress.com/about/ for an "about me" about WenSui Liu.
- The core of this macro is the usage of PROC TRANSREG.

%MONOBIN applied to Home Equity. Rows removed with missing X. Numeric target Y is needed.

```
DATA TRAINx; SET HELOC_2.TRAIN;
if NumTradesOpeninLast12M = -9 then delete;
Y = (RiskPerformance = "Bad");

%MONOBIN(data = TRAINx, y = Y, x = NumTradesOpeninLast12M);
```

Lower	Upper	#Bads	#Freq	BadRate	WoE	IV
0	1	1134	2379	0.47667	-0.13655	0.009293
2	2	511	988	0.51721	0.02569	0.000137
3	5	650	1186	0.54806	0.14967	0.005547
6	6	57	101	0.56436	0.21570	0.000978
7	9	66	97	0.68041	0.71250	0.009829
10	19	20	22	0.90909	2.25942	0.016600
		2438	4773			0.042384

Apart from "missing", the solution by %MONOBIN is the same as from %ORDINAL_BIN

Monotonic binning when L > 20

- The German Credit dataset (*) is often used as an example for fitting a Probability of Default (PD) model
  - It has 1000 rows.
  - Target is binary with levels 0 and 1 where 1 is a payment default and 0 is paid as agreed.
  - There are 3 continuous numeric predictors, one of them being AGE (age of borrower).
  - Let's restrict discussion to AGE. … AGE has 53 levels (L=53) and no missing and 5 zero-bins..
- It is sometimes a normal practice of a business to bin any continuous numeric X. This practice is founded on the idea that a binned predictor (now with multiple degrees of freedom) will better track the event-rate.
- Also, sometimes, the business may believe that "more of X" should imply "higher event-rate"
  - Then a monotonic binning is required.
- %ORDINAL_BIN is limited by program design to L ≤ 20 and L = 53 is absolutely out of the question.
- In contrast %MONOBIN has handle L = 53 without a problem.

(*) See https://archive.ics.uci.edu/dataset/144/statlog+german+credit+data.

next

%MONOBIN applied to AGE

next

Result from %MONOBIN

Lower	Upper	#Bads	#Freq	BadRate	WoE	IV
62	75	7	38	0.18421	-0.64078	0.01343
35	61	101	414	0.24396	-0.28378	0.03135
30	34	55	177	0.31073	0.05061	0.00046
26	29	57	181	0.31492	0.07007	0.00090
20	25	79	188	0.42021	0.52540	0.05654
19	19	1	2	0.50000	0.84730	0.00161
		300	1000			0.10429

After pro-process to combine 19-19 with 20-25

Lower	Upper	#Bads	#Freq	BadRate	WoE	IV
62	75	7	38	0.18421	-0.64078	0.01343
35	61	101	414	0.24396	-0.28378	0.03135
30	34	55	177	0.31073	0.05061	0.00046
26	29	57	181	0.31492	0.07007	0.00090
19	25	80	190	0.42105	0.52884	0.05792
		300	1000			0.10406

Result of binning includes a bin consisting of AGE=19 … with only 2 cases.

This bin solution is not a satisfactory.

But post-processing to combine 19-19 with 20-25 gives good results. See the lower table.

My hunch is %MONOBIN provides a very usable result (after perhaps minor post or pre-program processing) … at least, in most instances.

%NOD_BIN Followed by %ORDINAL_BIN for Monotonic Binning

Now I'll give a messy and somewhat ad hoc approach that has worked in the examples I've tried.

- %NOD_BIN with MODE=J easily processed the predictor AGE from GERMAN CREDIT
- The 20-bin solution from %NOD_BIN was specified.
- Also, MIN_NUM is set at 10 to combine low frequency bins (e.g. forces combine of AGE=19)
- Zero-bins were effectively handled by %NOD_BIN (see Appendix 1).

- The 20-bin solution was processed by %ORDINAL_BIN to find good monotonic binning solutions.

The Appendix 5 has all the supporting SAS code.

next

MONOTONIC AGE, AGAIN

- First monotonic solution occurs at k=6.
- But IV is much lower (worse) at k=6 than for the monotonic solution at k=5.
- Monotonic solution for k=5 is the same as post-processed %MONOBIN solution.
- Column "Turns" counts the times the odds changes direction (down-up or up-down).
- The best non-monotonic solution at k=4 has only 1 turn and higher IV (0.12394).

This k=4 solution might be even better ??

bins_in_solution	solution_num	IV	minus2LL	turns	best_rank	best_mono
20	1	0.23602	1175.43	16	*	
Rows 19 to 12 omitted						
11	1	0.20570	1180.84	9	*	
10	1	0.20032	1182.59	7	*	
9	1	0.19375	1183.97	5	*	
8	1	0.18680	1184.86	5	*	
7	1	0.16953	1187.42	4	*	
6	1	0.15168	1190.63	2	*	
6	5204	0.08337	1204.10	0		*
5	1	0.13436	1193.70	2	*	
5	205	**0.10406**	1199.87	0		*
4	1	0.12394	1196.28	1	*	
4	12	0.10402	1199.88	0		*
3	1	0.10015	1200.60	0	*	*
2	1	0.07317	1206.09	0	*	*

BIN Coding	Counts	Age Range
if AGE_B in (1,2) then AGE_B_B = 1 ;	190	19-25
if AGE_B in (3,4) then AGE_B_B = 2 ;	181	26-29
if AGE_B in (5,6) then AGE_B_B = 3 ;	177	30-34
if AGE_B in (7,8,9,10,11,12,13,14,15,16,17,18) then AGE_B_B = 4 ;	414	35-61
if AGE_B in (19,20) then AGE_B_B = 5 ;	38	62+

The table above is from %ORDINAL_BIN.
It processed the 20-bin solution from %NOD_BIN.

How Monotonic Binning Works with PROC TRANSREG

I think, in principle, %MONOBIN (which is mostly TRANSREG) accomplishes this:

a) Identifies all monotonic binning of X v. binary Y … I'll call them generically: X_BIN
b) For each, R-Square is computed for the linear model which is *effectively:*

PROC GLM; CLASS X_BIN; MODEL Y= X_BIN;

c) The optimal solution is the one with largest R-square (*)

(*) R-square, when applied to a binary target, is not the same measure as either IV or entropy. I assume it would be strongly correlated. But I have not investigated the correlation between R-square and IV or entropy.

Google Search of Entries for: "R Monotonic Binning"

The best that I found was Package 'monobin',

Monotonic Binning for Credit Rating Models Version 0.2.4 … Maintainer: Andrija Djurovic

Description:

iso.bin implements three-stage monotonic binning procedure. The first stage is isotonic regression (TRANSREG) used to achieve the monotonicity, while the remaining two stages achieve fine tuning (based on parameters of iso.bin

Same result as SAS macro %MONOBIN after post processing

See the Appendix 3 for R code implementation of 3 functions (iso.bin and two others) from 'monobin' which perform monotonic binning.

Same binning as %NOD_BIN - %ORDINAL_BIN combination

k	bin	count	iv_term
1	< 26	190	0.05792
2	26 - 29	181	0.00090
3	30 - 34	177	0.00046
4	35 - 61	414	0.03135
5	62+	38	0.01343
		IV =	0.10406

APPENDICES

Appendix 1: Handling zero-bins by %NOD_BIN

Appendix 2: PROC HPSPLIT and PRUNING by Cost-Complexity and Cross-Validation

Appendix 3: An R Package for Monotonic Binning

Appendix 4: %NOD_BIN - %ORDINAL_BIN combined for monotonic binning of continuous X

Appendix 5: SAS Code to process output from HPSPLIT and compare to %ORDINAL_BIN

Appendix 6: SAS Code to Fit and Process the Demo_Sat Example by HPSPLIT And Compare to %NOD_BIN

CONTACT ME FOR SAS MACRO CODE

BRUCE LUND

blund_data@mi.rr.com … or
blund.data@gmail.com

- Each macro has ~1200 lines of code (about 25% comments)
- Much code is used for data and parameter error checking
- It would be challenging to isolate actual binning logic

APPENDIX 1

Handling zero-bins by %NOD_BIN

How is a Zero-Bin handled by %NOD_BIN ?

Cannot run %NOD_BIN with METHOD=IV with a zero-bin due to computing a logarithm

One approach to avoid problem:

… Add a small positive number to the zero. The problem was $B_4 = 0$ … let's add 0.5 to B_4

- Use %NOD_BIN parameter ADD = 0.5 … Alternatively, ADD = 0.0001
- For small samples, adding 0.5 could have an undue influence on the value of IV

For METHOD=LL: A zero-bin is not a problem because we set 0*Log(0) = 0 in the LL equation

x	y		
	0 (B)	1 (G)	Total
1	5	2	7
2	6	1	7
3	4	5	9
4	0.5	4	4.5
Total	15.5	12	27.5

NOTE:

In %NOD_BIN processing: Any bin with either G or B < 1 is forced to combine before "normal" binning begins.

This prevents low frequency bins from "hanging around".

next

%NOD_BIN: An alternative for zero-bins (without adding 0.5)

I use "X_STAT" for the model c (or c-statistic) for the one-variable logistic model:

PROC LOGISTIC; CLASS X; MODEL Y = X;

X_STAT is another measure of power of X to predict Y

X_STAT can be computed by the formula shown below (no logarithms) (*)

$$X\_STAT = 0.5 * \{ \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} | B_i*G_j - B_j*G_i | / M + 1 \}$$

where $G = \sum_{k=1}^{L} G_k$ and $B = \sum_{k=1}^{L} B_k$ and $M = G * B$

Whenever two Bins are combined, the new X_STAT decreases (or stays the same).

x	y		
	0	1	Total
1	5	2	7
2	6	1	7
3	4	5	9
4	0	4	4
Total	15	12	27

If %NOD_BIN parameter ADD = "space", then %NOD_BIN combines a zero-bin with another bin so as to maximize X_STAT

k	REASON collapse row k to k-1	ZERO CELL	IV	Like-Ratio Chi_Sq	-2*Log L	X_STAT	L1	L2	L3	L4
4		YES	N/M	N/M	N/M	0.8056	1	2	3	4
3	X_STAT	NO	1.1121	6.9302	30.1657	0.7611	1	2	3+4	
2		NO	1.0199	6.4994	30.5965	0.7417	1+2	3+4		

Here is %NOD_BIN on the data above with ADD = , (space)

next

(*) See Lund (2019), Logistic Regression, Basics and Beyond, MWSUG 2019

APPENDIX 2

PROC HPSPLIT and PRUNING by Cost-Complexity and Cross-Validation

Appendix 2 gives an example which shows the steps in Pruning

Pruning by Cost-Complexity and Cross-Validation is a complicated process. A discussion, (but without an example) is given in this SAS Documentation:

https://documentation.sas.com/api/collections/pgmsascdc/9.4_3.4/docsets/stathpug/content/stathpug.pdf?locale=en#nameddest=stathpug_hpsplit_overview01

See pages 718-720

The following 7 slides go through an example. However, the reader should first read the SAS Documentation cited above.

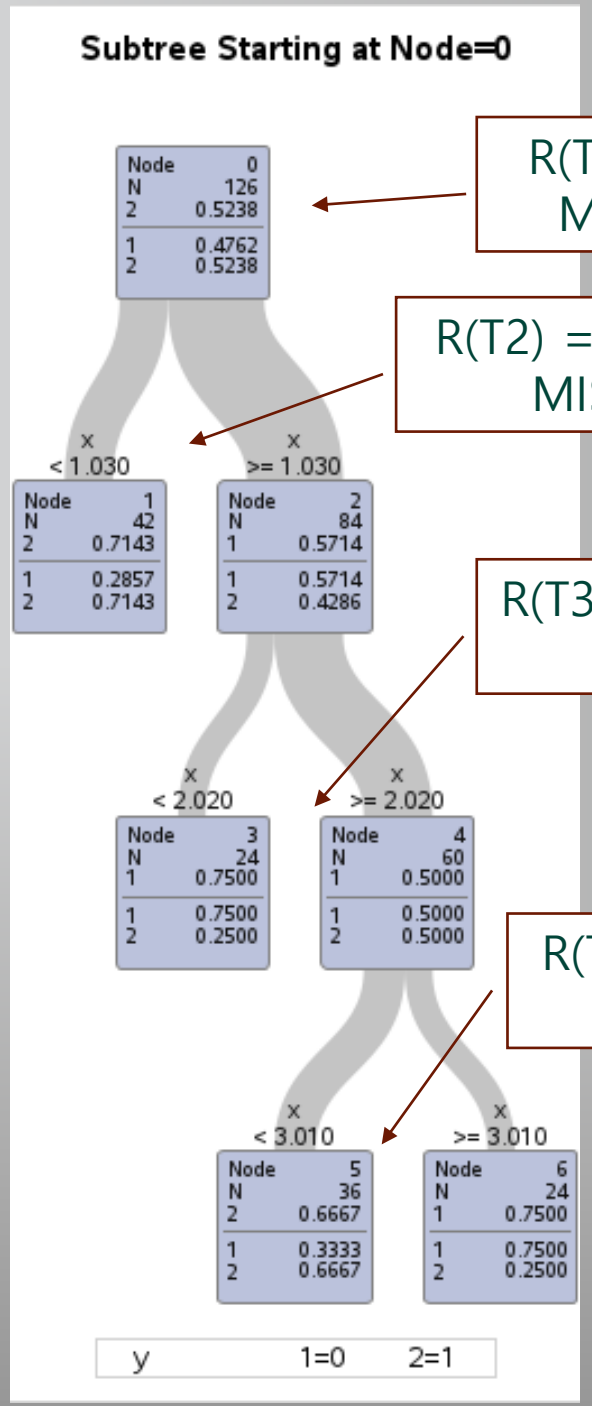
Full Tree before Pruning

```
DATA small;
input x y $ @@;
datalines;
4 0 3 1 1 0
4 0 3 1 1 0
4 0 3 1 1 1
4 1 2 0 1 1
3 0 2 0 1 1
3 0 2 0 1 1
3 1 2 1 1 1
;
run;
DATA small;
SET small small small small small small;
ods graphics on;
PROC HPSPLIT data = small seed=5
Cvmethod=random(3) CVMODELFIT;
CLASS y;
MODEL y = x;
GROW entropy;
PRUNE costcomplexity(leaves=4);
```

The Misclassification rates can be computed from the graphic on the right.

To avoid this hand calculation, this HPSPLIT code can be run 4 times where each time "leaves" is changed in the PRUNE. See below.

PRUNE costcomplexity (leaves=x) where x = 1 to 4.



R(T1)= One Leaf
MISC = .4762

R(T2) = 2 Leaves (1, 2)
MISC = .3810

R(T3) =3 Leaves (1, 3, 4)
MISC = .3810

R(T4) = 4 Leaves (1, 3, 5, 6)
MISC = .2857

Computing cost-complexity for the subtrees

Let T be a subtree, (= a tree starting at the root).

- Cost-complexity (CC) of T is define as: CC = R(T) + α*Leaves
  - For nominal target, R(T) is misclassification rate of T (computed across the leaves).
  - Leaves = number of leaves of T,
  - α is a penalty.

For SAS Documentation p. 718-719

Breiman et al. (1984) show that for each value of α, there is a subtree of T that minimizes cost complexity. When α_0, this is the full tree, T_0. As α increases, the corresponding subtree becomes progressively smaller, and the subtrees are in fact nested. Then, at some value of α, the root node has the minimal cost complexity for any α greater than or equal to that value. Because there are a finite number of possible subtrees, each subtree corresponds to an interval of values of α; that is,

$[0, \alpha_1)$ interval where T_0 (the full tree) has minimal cost complexity
$[\alpha_1, \alpha_2)$ interval where T_1 has minimal cost complexity
…
$[\alpha_m, \infty)$ interval where T_m (the root node) has minimal cost complexity

Computing α's and β's for the subtrees

The first step in Pruning is to solve for α's, as shown below.

Iteration #1

Solve for alpha from MISC rates and Difference in Leaves … using this formula and the MISC's from two slides back.

$$CC(Tx) - CC(T4) = R(Tx) - R(T4) + \alpha*(\#Leaves\ T4 - \#Leaves\ Tx) = 0$$

R(T3) – R(T4) + α*(Leaves T4 – Leaves T3) = 0.3810 - 0.2857 + α*(4-3) = 0 … solved: α = 0.0953
R(T2) – R(T4) + α*(Leaves T4 – Leaves T2) = 0.3810 - 0.2857 + α*(4-2) = 0 … solved: α = 0.0477 … this is α1
R(T1) – R(T4) + α*(Leaves T4 – Leaves T1) = 0.4762 - 0.2857 + α*(4-1) = 0 … solved: α = 0.0635

Iteration #2 … starting at T2:

R(T4) – R(T2) + α*(Leaves T2 – Leaves T1) = 0.4762 - 0.3810 + α*(2-1) = 0 … solved: α = 0.0952 … this is α2

Iteration #3 … starting at T1

R(T1) – R(T1) + α*(Leaves T1 – Leaves T1) = 0.4762 - 0.4762 + α*(1-1) = 0 … solved: α = ∞

Alpha intervals:
[0, α1) = [0, 0.0477)
[α1, α2) = [0.0477, 0.0952)
[α2, ∞) = [0.0952, ∞)

The next step is to compute Betas. Here is the SAS formula …

β0 = sqrt(0*α1) = 0
β1 = sqrt(α1*α2) = 0.0674,
β2 = sqrt(α2*∞) = 952.4 (🡐 computation unknown … this value is given by SAS)

Using cross-validation and β's to Prune

In this example of pruning by cost-complexity I wanted to use 3 fold cross-validation.
However, HPSPLIT does not put the cv folds into an output file to enable my example.
So, I created random samples k = 1, 2, 3 named cv1 cv2 cv3, each having 42 observations.

From here on, we cannot use the SAS reports … we must create our reports using cv1 cv2 cv3 and custom-made calculations.

See the SAS code:

cv12 cv13 c23 are the training datasets with holdouts cv3 cv2 cv1 respectively.

```
DATA sort; Set small;
R = ranuni(1);
PROC SORT Data = sort; by R;
DATA cv1 cv2 cv3; Set sort;
if _N_ <= 42 then output cv1;
else if _N_ <= 84 then output cv2;
else output cv3;
run;
DATA cv12; Set cv1 cv2;
DATA cv13; Set cv1 cv3;
DATA cv23; Set cv2 cv3;
```

Obtaining MISC for cross-validation folds and holdouts

HPSPLIT was run 4 times on each of the six datasets from the prior slide

The 4 HPSPLIT runs corresponded to Leaves = 1, …, 4.
These four runs provide the MISC for each number of leaves 1, …, 4.
Note: cv3 gives the holdout MISC for cv12, etc., etc.

```
%MACRO Run_HPSPLIT(dataset);
ods graphics off;
%DO Leaves = 1 %TO 4;
%IF &Leaves = 4 %THEN %DO;
ods graphics on;
%END;
PROC HPSPLIT DATA = &dataset seed=5;
Class y;
Model y = x;
GROW entropy;
PRUNE costcomplexity(leaves=&Leaves);
run;
%END;
%MEND;
```

```
%Run_HPSPLIT(CV12);
%Run_HPSPLIT(CV13);
%Run_HPSPLIT(CV23);
%Run_HPSPLIT(CV1);
%Run_HPSPLIT(CV2);
%Run_HPSPLIT(CV3);
```

Then look at the reports to obtain MISC.

Train	Misclassification		
Leaves	cv12	cv13	cv23
4	0.2738	0.2857	0.2857
3	0.3690	0.3492	0.3810
2	0.3690	0.3492	0.3810
1	0.4762	0.4603	0.4881

Holdout	Misclassification		
Leaves	cv1	cv2	cv3
4	0.2557	0.2619	0.3095
3	0.3333	0.3571	0.4048
2	0.3333	0.3571	0.4048
1	0.4524	0.5000	0.4762

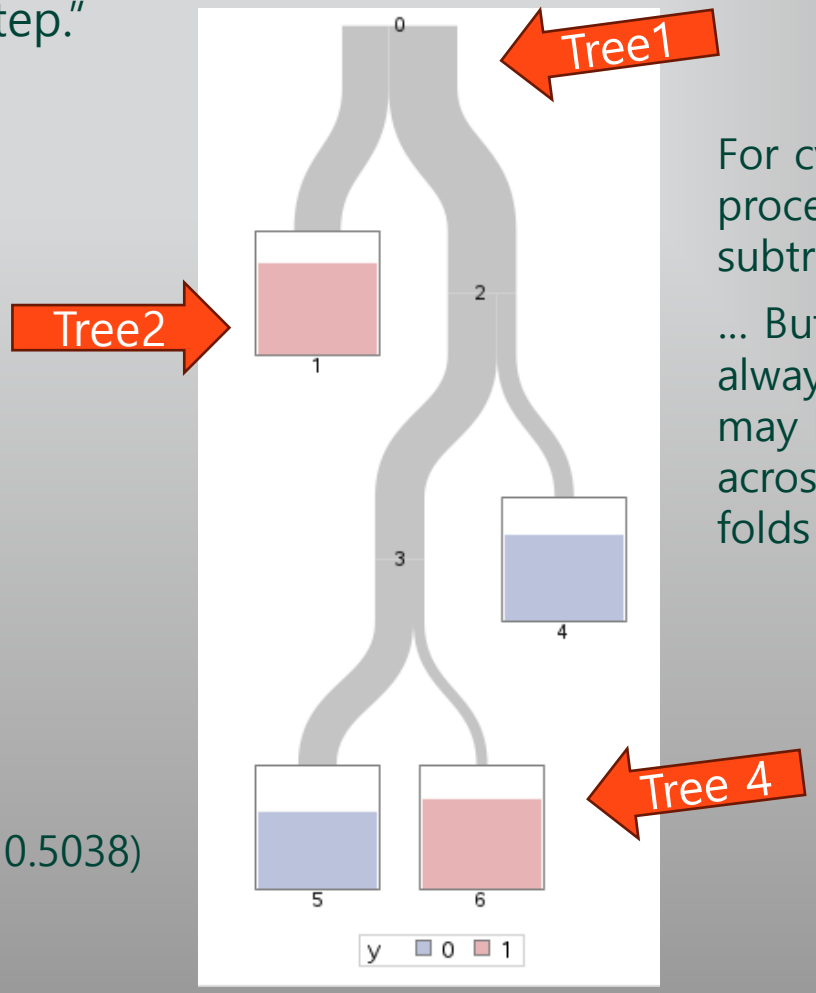
Prune each of the 3 trees (from cv12 cv13 cv23) using β's

From SAS HPSPLIT Documentation … determine the leaves (=subtrees) for the beta's as explained below:
"Using the β1, … , βm values that are calculated in step 2, create a sequence of subtrees for each βi as described in the pruning steps given earlier, but now using βi as a fixed value for α and minimizing the cost complexity, CC(T), to select a subtree at each pruning step."

Begin with cv12.
Compute CC = MISC + βi * Leaves
For i = 0 to 2 and Leaves = 1 to 4
For each βi, find the Leaves which give minimum CC.

Full Tree	MISC		
Leaves	cv12	cv13	cv23
4	0.2738	0.2857	0.2857
3	0.3690	0.3492	0.3810
2	0.3690	0.3492	0.3810
1	0.4762	0.4603	0.4881

For β0 = 0 the minimum occurs for Leaves = 4 (this is Tree 4)
For β1 = 0.0674 the minimum occurs for Leaves = 2 (value is 0.5038)
For β2 = 952.4 the minimum occurs for Leaves = 1



For cv13 and cv23 this process finds the same subtrees

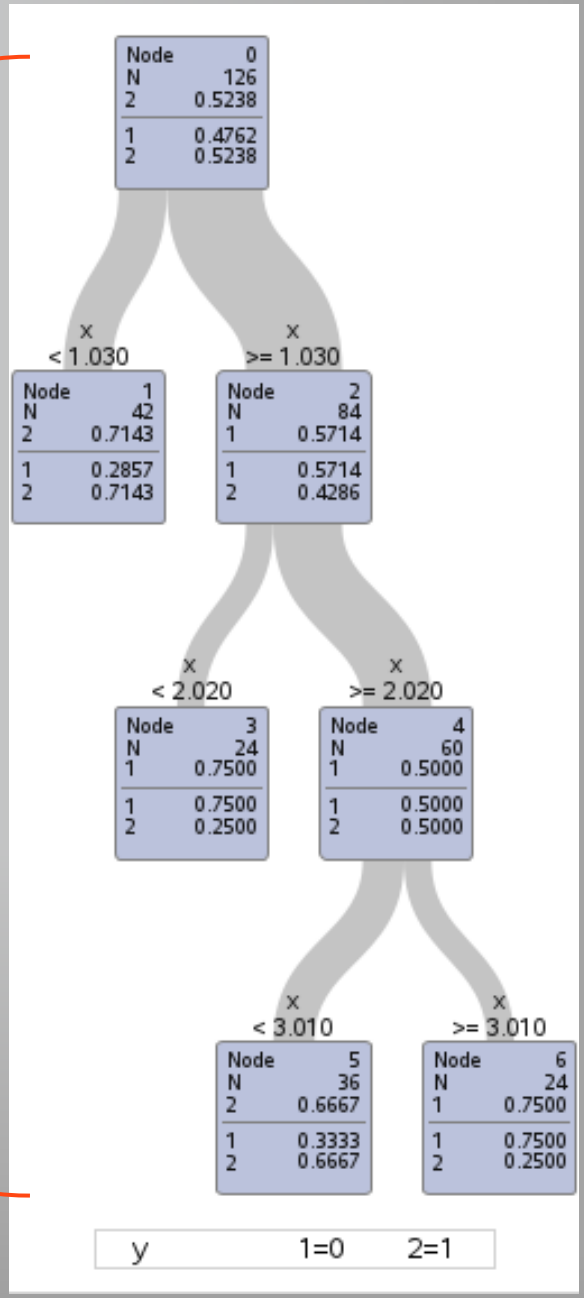
… But this is not always true. The trees may be different across the training folds

Find min avg. misc. rate on holdouts

Compute average holdout MISC. error for Leaves 1, 2, 4
Minimum average determines the PRUNED TREE.
(i.e. which subtree from the original full sample Tree).

A. alphas	B. Misclassification Holdout			C. Minimum gives PRUNED TREE
	cv1	cv2	cv3	Avg Misc.
[0, α1)	0.2557	0.2619	0.3095	0.2757
[α1, α2)	0.3333	0.3571	0.4048	0.3651
[α2, ∞)	0.4524	0.5000	0.4762	0.4762

This is Pruned Tree



Go the full-sample Tree (see Right) and PRUNE to 4 Leaves

The next slide how the PRUNED TREE from HPSPLIT compares with the example (where ad hoc cross-validation samples were created.).

Cost-complexity Pruning (3 fold) by running HPSPLIT is shown below

```
DATA small;
input x y $ @@;
datalines;
4 0 3 1 1 0
4 0 3 1 1 0
4 0 3 1 1 1
4 1 2 0 1 1
3 0 2 0 1 1
3 0 2 0 1 1
3 1 2 1 1 1
;
run;
DATA small;
SET small small small small small small;
ods graphics on;
PROC HPSPLIT DATA = small seed=5
CVCC PLOTS=CVCC
Cvmethod=random(3) CVMODELFIT;
CLASS y;
MODEL y = x;
GROW entropy;
PRUNE costcomplexity;
run;
```

3-Fold Cross Validation Assessment of Pruning Parameter													
N Leaves	Pruning Parameter		Average Square Error				Number of Leaves			Misclassification Rate			
			Min	Avg	Std Error	Max	Min	Median	Max	Min	Avg	Std Error	Max
* Selected pruning parameter													
4	0	*	0.2073	0.2287	0.0165	0.2475	4	4.0	4	0.2432	0.2833	0.0290	0.3111
2	0.0673		0.2313	0.2604	0.0243	0.2907	1	2.0	4	0.3111	0.4942	0.1393	0.6486
1	952.4		0.2494	0.2634	0.0135	0.2817	1	1.0	1	0.4444	0.5689	0.0892	0.6486

Compare Avg Misc. Rate (above) to Column C (below). Both the HPSPLIT and my created 3 fold cross-validation came up with the same pruning and similar MISC's.

A. alphas	B. Misclassification Holdout			C. Minimum gives PRUNED TREE
	cv1	cv2	cv3	Avg Misc.
[0, α1)	0.2557	0.2619	0.3095	0.2757
[α1, α2)	0.3333	0.3571	0.4048	0.3651
[α2, ∞)	0.4524	0.5000	0.4762	0.4762

Pruned Tree (4 Leaves)

APPENDIX 3

An R Package for Monotonic Binning

R code to run package “monobin”

```
# download a csv called German_Credit_Data.csv
# file csv must have a column called "age" and a column called "class" …
# where age is age of borrower in years and class has levels 0 and 1 (1=default on loan)
setwd(“<your path to German_Credit_Data.csv”>)
gcd <- read.csv("German_Credit_Data.csv", stringsAsFactors=TRUE)
install.packages("monobin")
library(monobin)
```

There are several functions in the package but we will use only three. Each will create monotonic binning of “age” where the target is “class”. There are no missing values in the dataset.

The first function is “iso.bin”. This function provides monotonic binning for “age” vs “class” in a manner similar to the SAS macro %MONOBIN but “iso.bin” has additional parameters to “clean up” the binning. In particular, there is a parameter “min.pct.obs” which gives the Minimum percentage of observations per bin (default 0.05). With the default the upper bin [35, ∞) was created. This may be satisfactory, but setting min.pct.obs = 0.03 will give two bins: [35, 62) and [62, ∞). This last bin has 38 observations (and is the same binning as produced by %NOD_BIN - %ORDINAL_BIN).

The code and reports are on the next slide.

iso.bin

```
age.bin_iso =iso.bin(gcd$age, gcd$class, min.pct.obs = 0.03)
# Need to have min.pct.obs = 0.03 in order to split [35, inf) into two bins
# remove comment below in order to see full report from cum.bin
# age.bin
report.age =
   data.frame(
      bin=age.bin_iso[[1]]$bin,
      count=age.bin_iso[[1]]$no,
      iv_term=age.bin_iso[[1]]$iv.b)
report.age
sum(report.age$iv_term)
```

```
> report.age
          bin count       iv_term
1 01 (-Inf,26)   190 0.0579210237
2   02 [26,30)   181 0.0009008687
3   03 [30,35)   177 0.0004579000
4   04 [35,62)   414 0.0313514651
5  05 [62,Inf)    38 0.0134258498
> sum(report.age$iv_term)
[1] 0.1040571
```

cum.bin

Cum.bin utilizes MAPA (Monotone Adjacent Pooling Algorithm) to perform monotonic binning. There are many papers and related algorithms and I can't do justice to this topic. Here is a simplified example of MAPA.

Suppose the (x,y) data is: (1,0), (1,1), (1,1), (2,0), (3,0), (3,1), (3.1)

Suppose the goal is to bin so that the y-rate is non-decreasing for increasing values of x.

In the first step the y values are averaged for x=1 to obtain 0.67. The first block becomes (1, 0.67). The current second block is (2, 0). The y-rates are decreasing. Therefore, merge the first and second blocks: (1_2, 0.5).

Now the block at 3 is (3, .67) with higher y-rate than for 1_2.

The final binning of x is {1,2}, {3} with monotonic y-rate.

```
age.bin_cum =cum.bin(gcd$age, gcd$class, g = 50)
# default for g is 15 ... the default is too small to obtain bins
which match iso.bin
# remove comment in order to see full report from cum.bin
#age.bin_cum
report.age =
  data.frame(
    bin=age.bin_cum[[1]]$bin,
    count=age.bin_cum[[1]]$no,
    iv_term=age.bin_cum[[1]]$iv.b)
report.age
sum(report.age$iv_term)
```

← This code produces the same reports as iso.bin

mdt.bin

mdt.bin implements monotonic binning driven by decision tree. The splitting metric for the binary target is the Gini index.

The parameter g is defined as: Number of splitting groups for each node. Default is 50. This definition is unclear to me. It appears to specify how may branches can come from a node. I tried g=2 and obtained a binning solution with 1 bin. Perhaps mdt.bin performs just 1 split with "g" branches … giving "g" leaves?

```
age.bin_mdt = mdt.bin(
  gcd$age,
  gcd$class,
  g = 50,
  min.pct.obs = 0.001, min.avg.rate = 0.001)
# remove comment in order to see full report from cum.bin
# age.bin_mdt
report.age =
  data.frame(
    bin=age.bin_mdt[[1]]$bin,
    count=age.bin_mdt[[1]]$no,
    iv_term=age.bin_mdt[[1]]$iv.b)
report.age
sum(report.age$iv_term)
```

With the default of g=50 the mdt.bin produced a sub-optimal binning solution with only 4 bins. I relaxed the min.pct.obs and min.avg.rate to small values of 0.001 (much smaller than defaults) and still obtained the same sub-optimal solution shown below:

```
> report.age
          bin count       iv_term
1 01 (-Inf,26)   190 0.0579210237
2   02 [26,30)   181 0.0009008687
3   03 [30,61)   584 0.0199136440
4  04 [61,Inf)    45 0.0067577518
> sum(report.age$iv_term)
[1] 0.08549329
```

APPENDIX 4

%NOD_BIN - %ORDINAL_BIN combined for monotonic binning of continuous X

Overview of the Approach

%NOD_BIN with specification ***MODE***=J finds an ordered solution for each k. When ***MODE***=J, an X with a large numbers of levels can be run through %NOD_BIN within a short time period. %NOD_BIN has been run successfully on a predictor X with 100 levels. However, %NOD_BIN does not guarantee that the optimal solution is found for any of the k's except k = L-1.

%NOD_BIN combined with %ORDINAL_BIN provides an approach to binning ordinal X when L ≤ 100 (or more). Preliminary binning by %NOD_BIN provides an ordered bin solution with L = 20. This 20-bin solution can then be processed by %ORDINAL_BIN.

This two-step approach was applied to AGE from German Credit. A 20-bin solution from %NOD_BIN was found where, additionally, ***MIN_NUM*** was set at 10 to combine low frequency bins.

Zero-bins were handled by X_STAT maximization (see Appendix 1).

Then the 20-bin solution was processed by %ORDINAL_BIN to find good monotonic binning solutions. Here is the code for the two-step process.

```
%NOD_BIN(DATASET=GERMAN.BANK, X=AGE, TARGET=Y, ZERO_ONE=YES,
  W=1, METHOD=IV, MODE=J, ORDER=D, MISS= , MIN_PCT= , MIN_NUM=10,
  MIN_BIN=20, MAX_BIN=20, VERBOSE= , VERBOSE2= , LL_STAT= ,
  WOE= WOE, ADD= , RUN_TITLE= 20 bin solution for ORDINAL_BIN);
```

%NOD_BIN provides the 20-bin solution as SAS code (see next slide)

The output SAS code from %NOD_BIN with 20 bins

```
DATA TEMP; SET GERMAN.BANK;
  if AGE in ( 19,20,21 ) then AGE_B = 001 ;
  if AGE in ( 22,23,24,25 ) then AGE_B = 002 ;
  if AGE in ( 26,27 ) then AGE_B = 003 ;
  if AGE in ( 28,29 ) then AGE_B = 004 ;
  if AGE in ( 30,31,32 ) then AGE_B = 005 ;
  if AGE in ( 33,34 ) then AGE_B = 006 ;
  if AGE in ( 35,36 ) then AGE_B = 007 ;
  if AGE in ( 37 ) then AGE_B = 008 ;
  if AGE in ( 38 ) then AGE_B = 009 ;
  if AGE in ( 39,40,41 ) then AGE_B = 010 ;
  if AGE in ( 42,43,44 ) then AGE_B = 011 ;
  if AGE in ( 45,46,47,48 ) then AGE_B = 012 ;
  if AGE in ( 49 ) then AGE_B = 013 ;
  if AGE in ( 50 ) then AGE_B = 014 ;
  if AGE in ( 51,52 ) then AGE_B = 015 ;
  if AGE in ( 53,54 ) then AGE_B = 016 ;
  if AGE in ( 55,56,57 ) then AGE_B = 017 ;
  if AGE in ( 58,59,60,61 ) then AGE_B = 018 ;
  if AGE in ( 62,63,64 ) then AGE_B = 019 ;
  if AGE in ( 65,66,67,68,70,74,75 ) then AGE_B = 020 ;
```

Report from %ORDINAL_BIN on 20 bins from %NOD_BIN

Now %ORDINAL_BIN is applied to the predictor AGE_B.

%ORDINAL_BIN(DATASET=TEMP, X=AGE_B, TARGET=Y, W=1, RANKING=IV, ORDER=D, MISS=, SUMMARYONLY=YES, N_BEST=, N_MONO=, MIN_PCT=, MIN_NUM=, MIN_BIN=, MAX_BIN=, NOPRINT_WOE=, PRINT1_WOE=, PRINT2_WOE=, RUN_TITLE=20 bin solution for ORDINAL_BIN, DELETE_PRIOR=);

Here is the report from %ORDINAL_BIN:

Obs	bins_in_ solution	solution _num	IV	minus2LL	turns	missing	best_ rank	best_ mono
1	20	1	0.23602	1175.43	16	N	*	
2-13	Obs 2 to 13 are omitted.							
14	7	1	0.16953	1187.42	4	N	*	
15	6	1	0.15168	1190.63	2	N	*	
16	6	5204	0.08337	1204.10	0	N		*
17	5	1	0.13436	1193.70	2	N	*	
18	5	205	0.10406	1199.87	0	N		*
19	4	1	0.12394	1196.28	1	N	*	
20	4	12	0.10402	1199.88	0	N		*
21	3	1	0.10015	1200.60	0	N	*	*
22	2	1	0.07317	1206.09	0	N	*	*

Best Solutions from %ORDINAL_BIN

A promising monotonic solution is found at k=5 with solution_num = 205. The IV is 0.10406. There is a monotonic solution for k=6 but the IV is much lower at IV = 0.08337.

The bins that comprise the k=5 solution are obtained by an additional run of %ORDINAL_BIN. Set ***MIN_BIN***=5 and ***MAX_BIN***=5 to exclude other k. To obtain SAS code for the 5-bin monotonic solution, set ***PRINT1_WOE***=5 and ***PRINT2_WOE***=5.

%ORDINAL_BIN(DATASET=TEMP, X=AGE_B, TARGET=Y, W=1, RANKING=IV, ORDER=D, MISS=, SUMMARYONLY=, N_BEST=, N_MONO=, MIN_PCT=, MIN_NUM=, MIN_BIN=5, MAX_BIN=5, NOPRINT_WOE=, PRINT1_WOE=5, PRINT2_WOE=5, RUN_TITLE=20 bin solution for ORDINAL_BIN, DELETE_PRIOR=);

solution_num	BIN Coding	Counts	Age Range
205	if AGE_B in (1,2) then AGE_B_B = 1 ;	190	19-25
205	if AGE_B in (3,4) then AGE_B_B = 2 ;	181	26-29
205	if AGE_B in (5,6) then AGE_B_B = 3 ;	177	30-34
205	if AGE_B in (7,8,9,10,11,12,13,14,15,16,17,18) then AGE_B_B = 4 ;	414	35-61
205	if AGE_B in (19,20) then AGE_B_B = 5 ;	38	62+

This table is identical to the table from %MONOBIN. However, admittedly, the method of using the "%NOD_BIN 20-bin solution followed by %ORDINAL_BIN" is a bit messy and is a somewhat ad hoc approach.

This method has the advantage of displaying other monotonic solutions and alternative non-monotonic solutions. The ***MIN_NUM*** parameter in %NOD_BIN can prevent solutions with small bin counts.

APPENDIX 5

SAS Code to Process Output from HPSPLIT

And Compare to %ORDINAL_BIN

Membership in Tree Bins

```
DATA TRAINx; SET HELOC_2.TRAIN;
if NumTradesOpeninLast12M = -9 then NumTradesOpeninLast12M = .;
PROC HPSPLIT data=TRAINx Assignmissing=BRANCH maxbranch=2
CVCC Plots=CVCC Cvmethod=random(10) seed=123;
Class RiskPerformance;
*PARTITION fraction(validate=.2, seed=1);
Model RiskPerformance = NumTradesOpeninLast12M;
Code file ="/home/blund0/HPSPLIT/Heloc_code.sas";
Grow entropy;
Prune costcomplexity;
*prune off;
run;
data TRAINx_scored; set TRAINx;
%INCLUDE "/home/blund0/HPSPLIT/Heloc_code.sas";
if NumTradesOpeninLast12M = . then NumTradesOpeninLast12M = -9;
run;
PROC FREQ Data = TRAINx_scored;
Tables (NumTradesOpeninLast12M RiskPerformance ) * _node_
/ missing norow nocol nopercent;
Title "Counts of Predictor by HPSPLIT Bins";
Title2 "-9 = missing, Rows given membership of Predictor in _Node_";
Title3 "_Node_ numbers are not ordered by Predictor ordering";
run;
TITLE; run;
```

Counts of Predictor by HPSPLIT Bins
-9 = missing, Rows given membership of Predictor in _Node_
Node numbers are not ordered by Predictor ordering

Frequency | Table of NumTradesOpeninLast12M by _Node_

NumTradesOpeninLast12M	_Node_(Node number)							
	2	4	5	7	9	11	12	Total
-9	456	0	0	0	0	0	0	456
0	0	0	1183	0	0	0	0	1183
1	0	0	1196	0	0	0	0	1196
2	0	0	0	988	0	0	0	988
3	0	0	0	0	617	0	0	617
4	0	0	0	0	372	0	0	372
5	0	0	0	0	0	197	0	197
6	0	0	0	0	0	0	101	101
7	0	57	0	0	0	0	0	57
8	0	27	0	0	0	0	0	27
9	0	13	0	0	0	0	0	13
10	0	10	0	0	0	0	0	10
11	0	4	0	0	0	0	0	4
12	0	1	0	0	0	0	0	1
13	0	2	0	0	0	0	0	2
14	0	2	0	0	0	0	0	2
16	0	1	0	0	0	0	0	1
17	0	1	0	0	0	0	0	1
19	0	1	0	0	0	0	0	1
Total	456	119	2379	988	989	197	101	5229

Frequency | Table of RiskPerformance by _Node_

RiskPerformance	_Node_(Node number)							
	2	4	5	7	9	11	12	Total
Bad	291	86	1134	511	553	97	57	2729
Good	165	33	1245	477	436	100	44	2500
Total	456	119	2379	988	989	197	101	5229

IV and Log-Likelihood from Tree Binning

```
/* Contact Bruce Lund for the two MACRO's that appear below */
DATA _TEMP_; Message = "Compute IV and Log-Likelihood for HPSPLIT bins of HELOC";
PROC PRINT Data = _Temp_;  run;
%CUM_LOGIT_SCREEN_2(TRAINx_scored, RiskPerformance, _NODE_ NumTradesOpeninLast12M, , );
%MULTI_LOGIT_SCREEN_1(TRAINx_scored, RiskPerformance, _NODE_ NumTradesOpeninLast12M, );
```

CUM_LOGIT_SCREEN_2, Version v05e, run on 25-MAR-2024 11:31:44
DATA SET = TRAINx_scored, TARGET = RiskPerformance, IV_ADJ = , MISS =
Summary Report

Obs	DATASET	VAR_NAME	Levels	CHAR-ACTER	MONO-TONIC	C_STAT	MODEL c (x-Stat)	LIFT per d.f.	IV (Info Value)
1	TRAINx_scored	NumTradesOpeninLast12M	19	NO		0.5139	0.5583	0.0048	n/a
2	TRAINx_scored	_Node_	7	NO		0.5036	0.5578	0.0179	**0.0557**

Multi_Logit_Screen_1, Version v06b, run on 25-MAR-2024 11:31:44
DATASET = TRAINx_scored, TARGET = RiskPerformance, SORT =
Likelihood Ratio Chi-Square (Significance) and Model c

Obs	Var_Name	Levels	Log_L_Intercept	Log_Likelihood	LRCS	df	Pr>ChiSq(Num)	Pr > ChiSq	MODEL_C	MODEL_C_ASE
1	NumTradesOpeninLast12M	19	-3619.45	-3578.92	81.0649	18	5.58E-10	<.0001	0.5583	0.007779
2	_NODE_	7	-3619.45	**-3583.63**	71.6443	6	1.88E-13	<.0001	0.5579	0.007465

Six Bin Solution (overall and monotone) from %ORDINAL_BIN

```
%ORDINAL_BIN(
DATASET=TRAINx, X=NumTradesOpeninLast12M, TARGET=RiskPerformance,
W=1, RANKING=IV, ORDER=D, MISS=MISS, SUMMARYONLY= , N_BEST=1,
N_MONO=1, MIN_PCT=, MIN_NUM=20, MIN_BIN=6, MAX_BIN=6, NOPRINT_WOE=YES,
PRINT1_WOE=, PRINT2_WOE=, RUN_TITLE=HELOC Train, DELETE_PRIOR=);
PROC PRINT DATA = __OBIN_NUMTRADESOPENINL_BEST_6;
VAR solution_num G_SUM1 - G_SUM6 B_SUM1 - B_SUM6;
Title "Counts of Good and Bad by (non-missing) BIN from ORDINAL_BIN";
run;
TITLE;
run;
```

See Next Slide for usage of PROC PRINT

HELOC Train
ORDINAL_BIN Version v15b_working, RUN ON 25-MAR-2024 11:49:18
SUMMARY REPORT - ALL LEVELS between 6 and 6
Dataset= TRAINx, Predictor= NumTradesOpeninLast12M, Target= RiskPerformance, Freq= 1, Miss= MISS, RANKING= IV
MISSING is NOT counted as a level in Bins_in_Solution

Obs	bins_in_solution	solution_num	IV	minus2LL	turns	missing	best_rank	best_mono	L1	L2	L3	L4	L5	L6
1	6	1	0.061083	7163.09	2	Y	*		0+1	2	3+4	5+6	7+8+9	10+11+12+13+14+16+17+19
2	6	4	0.059885	7164.64	0	Y		*	0+1	2	3+4+5	6	7+8+9	10+11+12+13+14+16+17+19

Repeat of Slide from Presentation

	OUT of BOX Solution: BINS from **HPSPLIT**						
BINs →	MISSING	0-1	2	3-4	5	6	7-19
Good (G)	165	1134	511	553	97	57	86
Bad (B)	291	1245	477	436	100	44	33
Ratio B/G	1.764	1.098	0.933	0.788	**1.031**	0.772	0.384

	6 BIN solutions from **%ORDINAL_BIN**						
Best OVERALL	MISSING	0-1	2	3-4	5-6	7-9	10-19
Ratio	1.764	1.098	0.933	0.788	**0.935**	0.470	0.100
Best MONO	MISSING	0-1	2	3-5	6	7-9	10-19
Ratio	1.764	1.098	0.933	0.825	0.772	0.470	0.100

HPSPLIT: Out of BOX defaults created 6 Bins plus Missing

- Not MONOTONIC Binning … See BIN with "5"
- IV = 0.0557
- -2*Log(L) = 7167.26

(See **Appendix** for SAS code)

%ORDINAL_BIN:

Best solution for 6 bins (by IV):

- IV = 0.0611
- -2*Log(L) = 7163.09

MONOTONIC for 6 bins

- IV = 0.0599
- -2*Log(L) = 7164.64

%ORDINAL_BIN: The OVERALL and MONOTONIC solutions are both better than Tree solution.
"Complete enumeration" will always find best BINNING for ordered X (via any measure)

APPENDIX 6

SAS Code to Fit and Process the Demo_Sat Example by HPSPLIT

And Compare to %NOD_BIN

SAS Code for HPSPLIT for DEMO_SAT example

```
DATA DEMO_SAT;
SET DEMO_SAT;
length X $4;
X = DEMO1 || DEMO2;
Do i = 1 to W;
output;
end;
run;
PROC HPSPLIT data=DEMO_SAT
CVMETHOD=random(10) seed=123;
CLASS Y X;
MODEL Y = X;
code file ="/home/blund0/HPSPLIT/Demo_sat_code.sas";
GROW entropy;
PRUNE costcomplexity;
run;
data DEMO_SAT_scored; set DEMO_SAT;
%INCLUDE "/home/blund0/HPSPLIT/Demo_sat_code.sas";
run;
PROC FREQ Data = DEMO_SAT_scored;
Tables (X Y ) * _node_
/ missing norow nocol nopercent;
Title1 "_Node_ numbers are not ordered by Predictor ordering";
run;
TITLE; run;
%CUM_LOGIT_SCREEN_2(DEMO_SAT_scored, Y, _NODE_, X, , );
%MULTI_LOGIT_SCREEN_1(DEMO_SAT_scored, Y, _NODE_ X, );
```

%NOD_BIN found a better BINNING solution

HPSPLIT solution found 4 leaves		Sat.	Not Sat.	Sat. Rate
		Y = 1	Y = 0	%(Y=1)
NODE #	TOTAL	4662	1579	74.7%
2	ALL OTHER	4265	1288	76.8%
4	2B+3B	306	204	60.0%
5	4B	50	52	49.0%
6	1B	41	35	54.0%

HPSPLIT: Out of BOX defaults created 4 Bins (leaves)

- IV = 0.0916
- -2*Log(L) = 6947.9

%NOD_BIN: 4 Bins were chosen as the solution

- IV = 0.1599
- -2*Log(L) = 6873.4

K=4 binning solution for X by %NOD_BIN		Sat.	Not Sat.	Sat. Rate
		Y = 1	Y = 0	%(Y=1)
BIN #	TOTAL	4662	1579	74.7%
1	1B+4B+2B+3B	397	291	57.7%
2	2D+4D+3D+3C+4C	1266	512	71.2%
3	1C+3G+1D+2F+4G+1E+2C+2G+2E+3E+3F	2266	660	77.4%
4	1F+1G+4E+4F	733	116	86.3%

HPSPLIT found two BINs (5 and 6) which have very low Sat. Rates.

But these BINs are small and do not contribute significantly to fit (IV and -2*Log(L))