Paper 2019-3687

Transforming Ordinal Predictors to Numeric for Binary Logistic Models

Bruce Lund, Independent Consultant

ABSTRACT

In binary logistic regressions for credit risk and marketing applications as well as for many social science studies there are ordinal predictors that are considered for use in the model. The modeler faces the question of how to utilize these ordinal predictors. The choices include conversion to dummy variables, recoding as weight of evidence, or assuming an interval scale for the ordinal levels and entering the effect as linear (or possibly as some transformation). The use of dummies or weight of evidence add parameters to the model (for weight of evidence the added parameters are implicit in the recoding). This paper provides a procedure to aid in the decision making regarding the handling of ordinal predictors. First, a measure is computed of the monotonic tendency of the ordinal versus the target (the dependent variable). If the ordinal is nearly monotonic, then a simple numeric recoding of the ordinal is made. This imposes an interval scale. Is this recoding appropriate? To answer this question a model comparison test is made between the saturated model (all dummies) and the model with the numeric (recoded ordinal) predictor. Acceptance of the null hypothesis of "no difference" allows the numeric (recoded ordinal) to be considered for inclusion in the model. SAS® code is given which carries out the ordinal to numeric recoding as well as the model comparison test. This code can process many ordinal predictors using an efficient, data driven approach that does not require "hard coding".

INTRODUCTION

Binary logistic regression models are widely used in credit risk modeling and marketing applications as well as for many social science studies. Here, the target (dependent variable) has two levels. In these models it is common to consider nominal, ordinal, or discrete[1] (NOD) predictors for use in the model. Nominal predictors are entered into a logistic regression model as dummy variables (often via a CLASS statement) or as a weight of evidence (WOE) transformation.

It is common to "bin" NOD predictors. Binning is the process of reducing the number of levels of a NOD predictor to achieve parsimony while preserving, as much as possible, the predictive power of the predictor. Binning algorithms are presented in Lund (2017). Binning is not utilized in this paper.

An ordinal or discrete predictor can also be entered into a logistic regression model as dummy variables or as a WOE transformation. However, there is a third possibility in the case of an ordinal or discrete predictor X. This is to enter X as a linear effect (or possibly as a transformation of X, such as Log[X]). For a discrete predictor there is no ambiguity regarding how to do this. Simply use: MODEL Y = X.

But for an ordinal predictor X there is a need to assign numbers to the levels of X. An approach is given in this paper that allows an ordinal predictor, if certain conditions are met, to be recoded as numeric and to be entered into the model as a "linear effect". The expression "linear effect" begs the question as to how the ordinal levels are recoded to be numeric. This is explained later in the paper.

The benefit of recoding is parsimony (fewer parameters in the model) and, hopefully, simplicity in the relationship between the predictor and log odds of the target.[2]

If the goal of the modeling is prediction, then success is measured by validation on a holdout data set. In this case the use of dummies or weight of evidence could contribute to over-fitting, in contrast to recoding of an ordinal as a linear effect.

---

[1] A discrete predictor is a numeric predictor with only "few values". The designation of "few" is subjective. It is used here to distinguish discrete from continuous (interval) predictors with "many values". A discrete predictor is often a "count" (e.g. number of children in household).

[2] WOE coding also may provide "simplicity in the relationship between the predictor and log odds of the target", but WOE comes at a cost of implicitly using degrees of freedom by involving the target in coding of the WOE's.

This paper provides a process and supporting SAS programs to determine if and when to code an ordinal predictor as a linear effect for the purpose of fitting a logistic regression model.[3]

INITIAL SCREENING OF NOD PREDICTORS

INFORMATION VALUE AND WEIGHT OF EVIDENCE

Before any time is invested in considering how to use a NOD predictor in a model, the predictor should first pass a preliminary screening test of its predictive power. A widely used measure of predictive power when screening NOD predictors is Information Value (IV). The calculation of the WOE of predictor X and its IV is shown in Table 1 below. Here, the WOE recoding transformed X to a numeric predictor X_woe.

X	Frequencies		Col % Y=0 "b_k"	Col % Y=1 "g_k"	Log(g_k/b_k) = **X_woe**	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**

Table 1: Illustration of the calculation of WOE and IV

The well-known book by Siddiqi (2017, p. 179) gives an interpretation of a predictor in terms of its IV.

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"

Table 2: Interpretation of IV values in terms of predictive power. Siddiqi (2017, p. 179)

If levels of a predictor are collapsed through binning, the IV is non-increasing.[4] If a NOD predictor has many levels and its IV is "weak" in the sense of Siddiqi, then it is possible that binning would reduce the IV to "not predictive".

A MACRO TO SCREEN NOD PREDICTORS

The SAS code below creates a data set named TestData for use in a logistic regression example. The target is Y with levels 0 and 1, and there are four predictors X1-X4. These predictors are coded as character variables but our "User" believes they have a meaningful ordering versus Y. Further, the character sort sequence correctly gives this ordering. There are 500 observations and no missing values.

```
DATA TestData;
Length X1-X4 $4;
Do I = 1 To 500;
/* N1 appears as Linear in Xbeta */
   If MOD(I,5) = 0 then N1 = 0;
    Else if MOD(I,5) = 1 then N1 = 2;
    Else if MOD(I,5) = 2 then N1 = 2.5;
    Else if MOD(I,5) = 3 then N1 = 3;
    Else if MOD(I,5) = 4 then N1 = 5;
   X1 = PUT(N1,Z3.1);
/* N2 appears as Log(N2) in Xbeta */
```

[3] See the related discussion in Harrell (2001, p. 34)

[4] When two levels of X are collapsed, the IV is non-increasing. There is equality only when the odds of the target, at the two levels of X in the collapse, are equal. For a proof, see Lund and Brotherton (2013, p. 17)

```
   N2 = (ranuni(1)> 0.5) + 1;
   If N2 > 1 then N2 = 2 + floor(10*(1-ranuni(1))**2);
   X2 = PUT(N2,Z2.);
/* N3 appears as (N3-3)**2 in Xbeta */
   N3 = Max(Floor(rannor(1) + 3.5),1);
   X3 = PUT(N3,Z1.);
/* Levels of X4 appear as dummies in Xbeta (A3 and B with 0 coefficient) */
   Random1 = ranuni(1);
   If Random1 < 0.50 then X4 = "A1";
    Else If 0.50 <= Random1 < 0.55 then X4 = "A2";
    Else If 0.55 <= Random1 < 0.60 then X4 = "A3";
    Else If 0.60 <= Random1 < 0.70 then X4 = "B ";
    Else If 0.70 <= Random1 < 0.80 then X4 = "C ";
    Else If 0.80 <= Random1 < 0.90 then X4 = "D ";
    Else X4 = 'F ';
Xbeta = 1 + 3*rannor(1) + 0.5*N1 - 0.01*LOG(N2) - 0.2*(N3-3)**2 -
1.0*(X4="A1") - 0.25*(X4="A2") + 0.05*(X4="C") + 0.10*(X4="D") +
1.0*(X4="F");
  Y = (Exp(Xbeta)/(1 + Exp(Xbeta)) > 0.75);
  Output;
  End;
run;
```

%CUM_LOGIT_SCREEN_2 is a SAS macro which screens NOD predictors.[5] A predictor with low predictive power, as measured by the macro, would be eliminated from further consideration.

For the TestData example, this macro is applied to the predictors X1-X4 with Target Y. The results are given in Table 3. The columns of Table 3 are discussed below.

The C_STAT is the c-statistic between the predictor and the target.[6] The C_STAT is only meaningful if the predictor has an ordering. The "model c" is the name for the logistic "c" which is given in a PROC LOGISTIC report. The model c is the c-statistic for the modelled logistic probabilities versus the target.[7] The column MONOTONIC has value YES if Prob(Y=1) is monotonic versus the predictor. This assumes the ordering of the predictor is meaningful. Of course, even if monotonic, the predictor can be weak.

LIFT (per d.f.) = [(MODEL c - C_STAT) / C_STAT] / d.f.[8] LIFT is zero if predictor is monotonic.[9] Values near zero indicate the predictor is nearly monotonic. If LIFT is zero or near zero, then recoding the predictor as numeric (in a manner to be discussed) may be effective. Although LIFT is indicative, the true TEST of whether to recode is given in the sections that follow.

From Table 3, X1 is monotonic and, based on LIFT, X4 is close to monotonic. Predictors X2 and X3 have low monotonic tendency.[10] Due to monotonic tendency of X1 and X4, they are prime candidates for recoding (in a manner to be discussed) to be numeric. Predictors X2 and X3 probably are unsuitable for recoding. The IV of X2 is 0.101 (barely within the "medium" of Table 2) and there are 11 levels. It is possible that binning of X2 would reduce the IV of the binned X2 to a value which would be unacceptably low. But in this paper binning will not be performed.

---

[5] %CUM_LOGIT_SCREEN_2 is written for the cumulative logit model. TARGET is character or numeric with 2 or more ordered levels. The macro produces the columns of TABLE 3 for each binary split of ordered levels of TARGET. If levels of TARGET are A, B, C, then there is a row in TABLE 3 for A vs. B,C and for A,B vs. C for each predictor.

[6] Here are the steps to compute the c-statistic for ordered predictor X and binary Y:

- IP ["Informative Pairs"] are the pairs of observations (r, s) where Targets $Y_r \neq Y_s$
- If $Y_r > Y_s$ and $X_r > X_s$, then then IP is CONCORDANT
- If $Y_r > Y_s$ and $X_r < X_s$, then then IP is DISCORDANT
- Else TIE …. And c-Statistic = {CONCORDANT + 0.5*TIE} / IP

[7] See Lund and Brotherton (2013) for a study of association between IV and model c (called X_STAT in that paper).

[8] d.f. = number of levels of X minus 1.

[9] Predictor X is monotonic with respect to Y if C_STAT = MODEL c. (Otherwise, C_STAT < MODEL c).

[10] "Near zero" depends on number levels of X. See APPENDIX for discussion and examination of the lift of X2-X4.

Considering the "weak" IV for X3 of only 0.061, this predictor might be eliminated now as part of the screening. However, for sake of illustration it will be kept for further analysis.

Obs	VAR_ NAME	Levels	CHAR-ACTER	MONO-TONIC	C_STAT	MODEL c	LIFT per d.f.	IV (Info Value)
1	X1	5	YES	YES	0.610	0.610	0	0.177
2	X2	11	YES		0.543	0.576	0.0061	0.101
3	X3	6	YES		0.517	0.567	0.0193	0.061
4	X4	7	YES		0.601	0.604	0.0008	0.172

Table 3: %CUM_LOGIT_SCREEN_2 (TestData, Y, , X1 X2 X3 X4, YES,);

HOW TO ENTER X1, X2, X3, X4 IN THE LOGISTIC MODEL?

The reader may fit the model below and find that CLASS X1 and CLASS X4 are strongly significant. Meanwhile, CLASS X3 is only borderline and CLASS X2, with 10 degrees of freedom, is not significant. The model c is modestly strong at **0.690**.

```
PROC LOGISTIC DATA = TestData;
CLASS X1 X2 X3 X4;
MODEL Y = X1 X2 X3 X4;
run;
```

There are 25 parameters (excluding intercept). Can adequate model fit still be obtained if one or more of the CLASS effects were replaced with "LINEAR effects" and thereby reduce the number of parameters?

THE RECODING FOR A LINEAR EFFECT

However, the expression "LINEAR effect" begs the question as to how the ordinal levels are to be recoded to be numeric. The simple solution that is given in this paper is to replace the lowest level of the predictor with "1" and then to add "1" for each successive level.

So, ordinal levels A, B, T, U become 1, 2, 3, 4.

Now the question arises as to how to judge the adequacy of the replacement of a CLASS effect with this recoded LINEAR effect.

THE TEST TO DETERMINE WHETHER TO RECODE

Suppose X was originally ordinal but has been recoded as numeric X_n. Suppose X_n has k levels and the target is Y. Four steps are needed. The first two steps utilize the "score chi-squares" of X_n and of CLASS X from the fit of a logistic model.[11] The four steps are:

1. Compute X^2_{Linear}, the score chi-square for X, from MODEL Y = X_n;
2. Compute X^2_{Class}, the score chi-square for CLASS X from the model: CLASS X; MODEL Y = X;
3. Compute the model comparison chi-square as $\text{X}^2_{\text{Compare}} = \text{X}^2_{\text{Class}} - \text{X}^2_{\text{Linear}}$. This is approximately a chi-square with k-2 degrees of freedom.
4. Compute the right tail probability for $\text{X}^2_{\text{Compare}}$. A significant value accepts the alternative hypothesis that there is a difference between the CLASS Effect and the LINEAR Effect.

For example, for the case of X1 in TestData, the recoded X1 is called X1_n and has levels 1, 2, 3, 4, 5. The score chi-squares and test statistic are shown below (and in table 5 near the end of the paper):

$\text{X}^2_{\text{Compare}} = 21.578 - 18.615 = 2.963$ with 3 d.f.

[11] Why not use the likelihood ratio chi-square (which is approximated by the score chi-square)? The reason is that score chi-squares are available in an ODS OUTPUT data set from PROC LOGISTIC. This data set enables the programming of an efficient method to compare a large number of recoded ordinal (or also actual numeric) predictors against their CLASS equivalents. More detail is given in SAS coding sections of the paper that follow.

The right tail probability equals 0.397 and is insignificant. On this basis, X1_n can be used in the model to carry forward the information from predictor X1. This is a saving of 3 parameters versus CLASS X1.

SAS PROGRAM TO RECODE ORDINAL X AS NUMERIC X_n

Data set TestData is used in this discussion. The goal is to create a new data set with numeric predictors X1_n, X2_n, X3_n, X4_n which are recodings of the ordinal variables X1 X2 X3 X4. Additionally, this data set must be structured as input to PROC LOGISTIC. The desired recordings are shown below:

X1: “0.0”, “2.0”, “2.5”, “3.0”, “5.0” recoded as 1, 2, 3, 4, 5
X2: “01”, “02”, “03”, “04”, “05”, “06”, “07”, “08”, “09”, “10”, “11” recoded as 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
X3: “1”, “2”, “3”, “4”, “5”, “6” recoded as 1, 2, 3, 4, 5, 6
X4: “A1”, “A2”, “A3“, “B “, “C “, “D “, “F “ recoded as 1, 2, 3, 4, 5, 6, 7

There are three main steps to achieve these recodings. The data set TestData is used to illustrate.

Step 1:

In PROC SUMMARY the COMPLETETYPES creates an output observation for every combination of levels of X1 X2 X3 X4 even if such a combination does not exist in the data. If such a combination does not exist in the data, the value of _freq_ is 0. The output data set Summout is sorted by X1 X2 X3 X4.

```
/* Step 1 */
PROC SUMMARY DATA = TestData COMPLETETYPES;
CLASS X1 X2 X3 X4 ;
OUTPUT OUT = Summout;
TYPES X1 * X2 * X3 * X4;
run;
```

Step 2:

In this DATA Step the recoding is accomplished. The recoded predictors have suffix “_n”. Recoding is performed even when _freq_ = 0.

```
/* Step 2 */
DATA Convert; SET Summout;
  BY X1 X2 X3 X4;
RETAIN X1_n X2_n X3_n X4_n 0;
If first.X4 then X4_n + 1;
If first.X3 then X3_n + 1;
If first.X2 then X2_n + 1;
If first.X1 then X1_n + 1;
OUTPUT;
If last.X3 then X4_n = 0;
If last.X2 then X3_n = 0;
If last.X1 then X2_n = 0;
run;
```

Step 3:

The PROC SUMMARY in step 3 creates unique combinations of levels of X1 X2 X3 X4 and Y. The output Summout2 is sorted by X1 X2 X3 X4.

In the MERGE Step, only the observations from Convert that match to an observation from Summout2 are output. This is a 2 to 1 merge for cases where, for a combination of X1 X2 X3 X4, there is both an observation where Y=0 and an observation where Y=1.

The version of the variable _freq_ from Summout2 overwrites _freq_ from Convert.

The desired recoding X1_n, …, X4_n now exists in Convert2.

```
/* Step 3 */
PROC SUMMARY DATA = TestData ;
CLASS X1 X2 X3 X4 Y;
OUTPUT OUT = Summout2;
TYPES X1 * X2 * X3 * X4 * Y;
run;
DATA Convert2; MERGE Convert Summout2(IN=summout2);
BY X1 X2 X3 X4;
If summout2;
run;
```

SAS CODE FOR MODEL COMPARISON: RECODED ORDINAL V. CLASS

Now the score chi-squares for each recoded predictor versus the target, as well as for the CLASS version of each recoded predictor, can be generated by PROC LOGISTIC. These score chi-squares are output to a data set named EffectNotInModel via the ODS OUTPUT statement.[12] The ODS EXCLUDE ALL eliminates printout from PROC LOGISTIC.

In PROC LOGISTIC the DETAILS option in the SELECTION statement is required in order to create EffectNotInModel. The score chi-square for every predictor being considered for entry at each step of FORWARD is saved in EffectNotInModel. But since STOP=1, only the predictors and their score chi-squares for the first step are saved.

```
ODS EXCLUDE ALL;
ODS OUTPUT EffectNotInModel = EffectNotInModel;
PROC LOGISTIC DATA = Convert2;
CLASS X1 X2 X3 X4;
MODEL Y = X1_n X1  X2_n X2  X3_n X3  X4_n X4
/ SELECTION = FORWARD SLE = 1 STOP = 1 DETAILS;
FREQ _freq_;
run;
ODS EXCLUDE NONE;
PROC PRINT DATA = EffectNotInModel;
run;
```

Obs	Step	Effect	DF	ScoreChiSq	ProbChiSq
1	0	X1_n	1	18.6151	<.0001
2	0	X1	4	21.5781	0.0002
3	0	X2_n	1	1.5375	0.2150
4	0	X2	10	12.1026	0.2782
5	0	X3_n	1	0.2256	0.6348
6	0	X3	5	7.5899	0.1803
7	0	X4_n	1	18.3148	<.0001
8	0	X4	6	20.2135	0.0025

Table 4: Print out of EffectNotInModel

[12] For computation efficiency of PROC LOGISTIC, the "predictor to enter" at each step of FORWARD is the one with the largest score chi-square at this step. The "ideal" criterion would be to take the largest likelihood ratio chi-square but this is computationally unavailable. The score chi-square is usually a good approximation to likelihood ratio chi-square and is used instead.

In Table 4 the score chi-square for each recoded predictor X1_n, …, X4_n, as well as for the CLASS variable usage of X1 - X4, is printed. (Notice that the effects appear in Table 4 in the same order as in the statement MODEL Y = X1_n X1 X2_n X2 X3_n X3 X4_n X4. This fact is used by the DATA Step below.)

Neither X2_n or CLASS X2 is significant and could now be dropped. Also, X3_n is not significant and the significance of CLASS X3 is no better than borderline. But for illustration all four predictors X1_n, …, X4_n are analyzed further in the next step.

CODING THE MODEL COMPARISON TEST

The model with predictor X1_n is a nested model within the saturated model having predictor CLASS X1. A model comparison test, using score chi-squares, can be used to compare whether CLASS X1 is statistically different than X1_n. The test statistic is shown:

$$\text{Test-Statistic} = X^2_{\text{Compare}} = X^2_{\text{Class}} - X^2_{\text{Linear}}$$

For large samples this difference is a chi-square with k-1 - 1 = k-2 degrees of freedom when X1_n has k levels.

The same model comparison tests are applied to X2_n, X3_n, and X4_n. In the DATA Step below the EffectNotInModel data set is processed.

```
DATA Report; SET EffectNotInModel;
LABEL MODEL_Compare = "MODEL_Compare / Significance";
RETAIN
Var_Name Effect2 DF1 DF2 ScoreChiSq1 ScoreChiSq2 ProbChiSq1 ProbChiSq2;
If MOD(_N_,2)=1 then Do; /* Odd numbered Observations = Linear effects */
   Var_Name=Effect; DF1=DF; ScoreChiSq1=ScoreChiSq; ProbChiSq1=ProbChiSq;
   End;
If MOD(_N_,2)=0 then Do; /* Even numbered Observations = Class effects */
   Effect2=Effect; DF2=DF; ScoreChiSq2=ScoreChiSq; ProbChiSq2=ProbChiSq;
   MODEL_Compare = 1 - probchi(ScoreChiSq2-ScoreChiSq1, DF2-DF1);
   OUTPUT; /* One output observation for each input pair */
   End;
run;
PROC PRINT DATA = Report LABEL SPLIT = "/";
VAR Var_Name DF1 DF2 ScoreChiSq1 ScoreChiSq2 MODEL_Compare;
run;
```

Obs	Var_Name	DF1	DF2	ScoreChiSq1	ScoreChiSq2	MODEL_Compare Significance
1	X1_n	1	4	18.6151	21.5781	0.3974
2	X2_n	1	10	1.5375	12.1026	0.3067
3	X3_n	1	5	0.2256	7.5899	0.1178
4	X4_n	1	6	18.3148	20.2135	0.8630

Table 5: Model Comparison - Class Effect vs Recoded Linear Effect

Inspecting the column "MODEL_Compare Significance", the conclusion is reached that predictors X1_n and X4_n may replace the class usage of X1 and X4. But CLASS X3 is preferred to X3_n (with borderline significance of 0.1178). The decision to drop X2_n and CLASS X2 was discussed earlier. Only 7 non-intercept parameters are needed to fit the model with X1_n, CLASS X3, X4_n.

```
PROC LOGISTIC DATA = Convert2;
CLASS X3;
MODEL Y = X1_n X3 X4_n;
FREQ _freq_;
run;
```

The model c for this simplified model is 0.669 versus 0.690 for the original model with CLASS predictors X1 - X4. This is a noticeable reduction in model c but it comes at the considerable improvement in measures of parsimony such as AIC and SBC (where lower is better) as shown below:

MODEL	AIC	SBC	MODEL c
CLASS X1 - X4; MODEL Y= X1 - X4;	679.25	788.83	0.690
CLASS X3; MODEL Y = X1_n X3 X4_n;	660.71	694.43	0.669

Table 6: Model c versus Parsimony - Class Effect vs Recoded Linear Effect

Of course, the simpler model is also more likely to be validated on a hold-out sample since overfitting would be avoided.

The SAS programming steps given above can be easily modified to extend the processing to more than four predictors.

These steps also provide a skeleton for developing a macro program. More functionality might be added to a macro program by adding a macro parameter which allows the user to perform other recodings of the ordinal, such as “quadratic”, “square root”, “logarithmic”, and to test each of these recodings against the dummy variable coding.

SGF Dallas 2019, v04

REFERENCES

Harrell, Jr, F. (2001). *Regression Modeling Strategies*, New York, NY, Springer-Verlag.

Hinkle, D., Wiersma, W., and Jurs, G. (2003). *Applied Statistics for the Behavioral Sciences* (5th ed.), Houghton Mifflin

Lund B. (2017). SAS® Macros for Binning Predictors with a Binary Target, *Proceedings of the SAS Global Forum 2017 Conference,* Cary, NC, SAS Institute Inc., paper 969.

Lund B. and Brotherton D. (2013). Information Value Statistic, *MWSUG 2013, Proceedings*, Midwest SAS Users Group, paper AA-14.

Siddiqi, N. (2017). Intelligent Credit Scoring, 2nd edition, Hoboken, NJ, John Wiley & Sons, Inc.

ACKNOWLEDGMENTS

Dave Brotherton of OneMagnify of Detroit provided helpful insights and suggestions.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Bruce Lund
Independent Consultant
blund_data@mi.rr.com or blund.data@gmail.com

Contact the author for a copy of the SAS macro program %CUM_LOGIT_SCREEN_2

APPENDIX

The purpose of this appendix is to give a quantitative relationship of Lift (per d.f.) versus monotonicity. An external measurement of monotonicity is Spearman rank correlation. A simulation study was conducted to compare Spearman rank correlation to Lift. Here is an outline of the methodology.

- The population to be sampled is the "K by 2 Tables" where total table cell count is set at N = 5000. The two columns in a table represent the "0's" and the "1's" of the Target. The number of levels of X equals K.

  Here is one example of a 3 by 2 table where N = 5000.

	Y=0	Y=1
X=1	500	800
X=2	1200	500
X=3	900	1100
	TOTAL = 5000	

- For each table, Lift (per d.f.) and Spearman were computed. For the table given above, Lift and Spearman are reported below. Here, LIFT is far from zero (0.1392) and the Spearman rank correlation is negligible.

Predictor	K	C_STAT	MODEL c	LIFT (per d.f)	Spearman
X	3	0.5008	0.6402	0.1392	-0.0015

- All tables with model c > 0.75 were deleted from the simulation. These deletions were made on the grounds that predictors X with model c > 0.75 would be unusual (too strong) for models used in quantitative marketing or credit risk. The number of tables in the sample is an input to the simulation program. For values of K < 10 the number of tables was set at 100,000. After removal of tables with model c > 0.75, there were about 75,000 usable tables. For K ≥ 10 the number of tables must be increased as explained in the outlined dot-points below.

- LIFT (per d.f.) was divided into narrow ranges as shown:

```
if Lift < 0.00001 then Lift_range = "00. 000 - .00001";
else if Lift < 0.001 then Lift_range = "01 .00001 - .001";
else if Lift < 0.002 then Lift_range = "02 .001 - .002";
else if Lift < 0.003 then Lift_range = "03 .002 - .003";
else if Lift < 0.004 then Lift_range = "04 .003 - .004";
else if Lift < 0.005 then Lift_range = "05 .004 - .005";
else if Lift < 0.006 then Lift_range = "06 .005 - .006";
else if Lift < 0.007 then Lift_range = "07 .006 - .007";
else if Lift < 0.008 then Lift_range = "08 .007 - .008";
else if Lift < 0.009 then Lift_range = "09 .008 - .009";
else if Lift < 0.010 then Lift_range = "10 .009 - .010";
else if Lift < 0.011 then Lift_range = "11 .010 - .011";
else if Lift < 0.012 then Lift_range = "12 .011 - .012";
else if Lift < 0.013 then Lift_range = "13 .012 - .013";
else if Lift < 0.014 then Lift_range = "14 .013 - .014";
else if Lift < 0.015 then Lift_range = "15 .014 - .015";
else if Lift < 0.016 then Lift_range = "16 .015 - .016";
else if Lift < 0.017 then Lift_range = "17 .016 - .017";
else if Lift < 0.018 then Lift_range = "18 .017 - .018";
else if Lift < 0.019 then Lift_range = "19 .018 - .019";
else if Lift < 0.020 then Lift_range = "20 .019 - .020";
else if Lift < 0.021 then Lift_range = "21 .020 - .021";
else if Lift < 0.022 then Lift_range = "22 .021 - .022";
else if Lift < 0.023 then Lift_range = "23 .022 - .023";
else if Lift < 0.024 then Lift_range = "24 .023 - .024";
else if Lift < 0.025 then Lift_range = "25 .024 - .025";
else Lift_range = "26 .025 - up";
```

Pearson correlation of 0.30 or less is characterized as “Little if any correlation” by Hinkle, Wiersma, and Jurs (2003). Of course, for this simulation Spearman rank correlation is used, but the 0.30 is still be utilized for the discussion which follows.

- For values of K (i.e. number of levels of X) the average absolute Spearman was computed for each Lift_range. Also computed were the percent of absolute Spearman’s with value above 0.30 as well as the average value of absolute Spearman (approximated by using first LIFT range) for those cases where X was monotonic for that table.
- The Lift_ranges for X2, X3, and X4 are shown in Table 6. The average absolute Spearman, percent absolute Spearman above 0.30, and the average value of absolute Spearman if monotonic were reported. See Table 6.

Predictor	Lift_range from TABLE 3 for predictor	K	Number of Tables in simulation	Ave |Spearman|	% |Spearman| > 0.30	Approx. |Spearman| if monotonic
X2	.006 - .007	11	1,044,477	0.235	19.9%	0.321
X3	.019 - .020	6	72,574	0.179	5.6%	0.304
X4	.00001 - .001	7	71,566	0.308	55.9%	0.318

Table 6: Lift_range versus Spearman Rank Correlation for X2, X3, X4

Using this guideline, average absolute Spearmen of 0.308 for X4 exceeds the 0.30 threshold and, moreover, the average absolution Spearman, if monotonic, is 0.318 which is close to 0.308. (It is possible, however, for an individual non-monotonic absolute Spearman to greatly exceed the 0.318.)

In the cases of X2 and X3 the average absolute Spearman does not come close to the 0.30 threshold and the percentage of the sample exceeding 0.30 is low.

It is interesting that the right-most column is fairly stable over K = 6, 7, 11.

On the basis of this simulation it seems fair to characterize X4 as “near monotonic” and X2, X3 as not “near monotonic”.

By running this simulation for K = 3, …, 20 it would be possible to create an association between “Lift_range” and “Near monotonic”. One issue in running the simulation for K ≥ 10 is that the number of tables in the simulation must be large in order to populate the lower “Lift_range” categories. At K = 20 the number of tables needed for the simulation would be well into the many millions.

Contact the author for the SAS code used in this simulation.