Predictive Modeling Using Logistic Regression

Course Notes

Predictive Modeling Using Logistic Regression Course Notes was developed by Mike Patetta. Additional contributions were made by Jordan Bakerman, Peter Christie, Marc Huber, Lorne Rothman, and David Yeo. Editing and production support was provided by the Curriculum Development and Support Department.

Predictive Modeling Using Logistic Regression Course Notes

Book code E70981, course code LWPMLR42/PMLR42, prepared date 06Apr2017. LWPMLR42_001

ISBN 978-1-63526-175-2

Table of Contents

To learn more…

For information about other courses in the curriculum, contact the SAS Education Division at 1-800-333-7660, or send e-mail to training@sas.com. You can also find this information on the web at http://support.sas.com/training/ as well as in the Training Course Catalog.

For a list of other SAS books that relate to the topics covered in this course notes, USA customers can contact the SAS Publishing Department at 1-800-727-3228 or send e-mail to sasbook@sas.com. Customers outside the USA, please contact your local SAS office.

Also, see the SAS Bookstore on the web at http://support.sas.com/publishing/ for a complete list of books and a convenient order form.

Chapter 1 Predictive Modeling

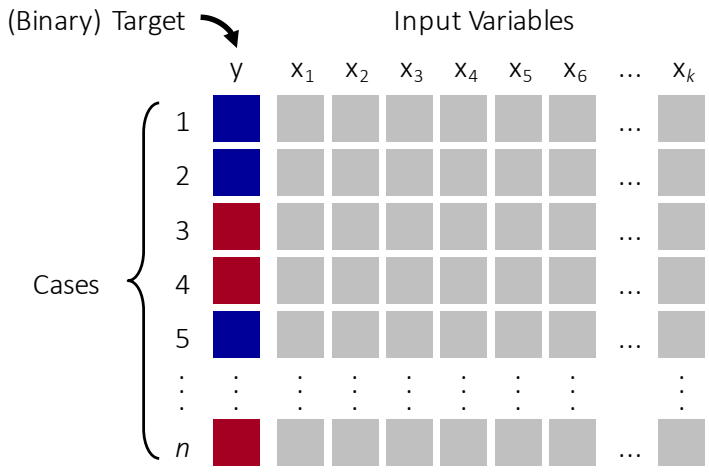
1.1 Introduction

Objectives

- Describe the predictive modeling nomenclature.
- Identify the applications for predictive modeling.
- Illustrate the variable annuity data set.

3

Supervised Classification

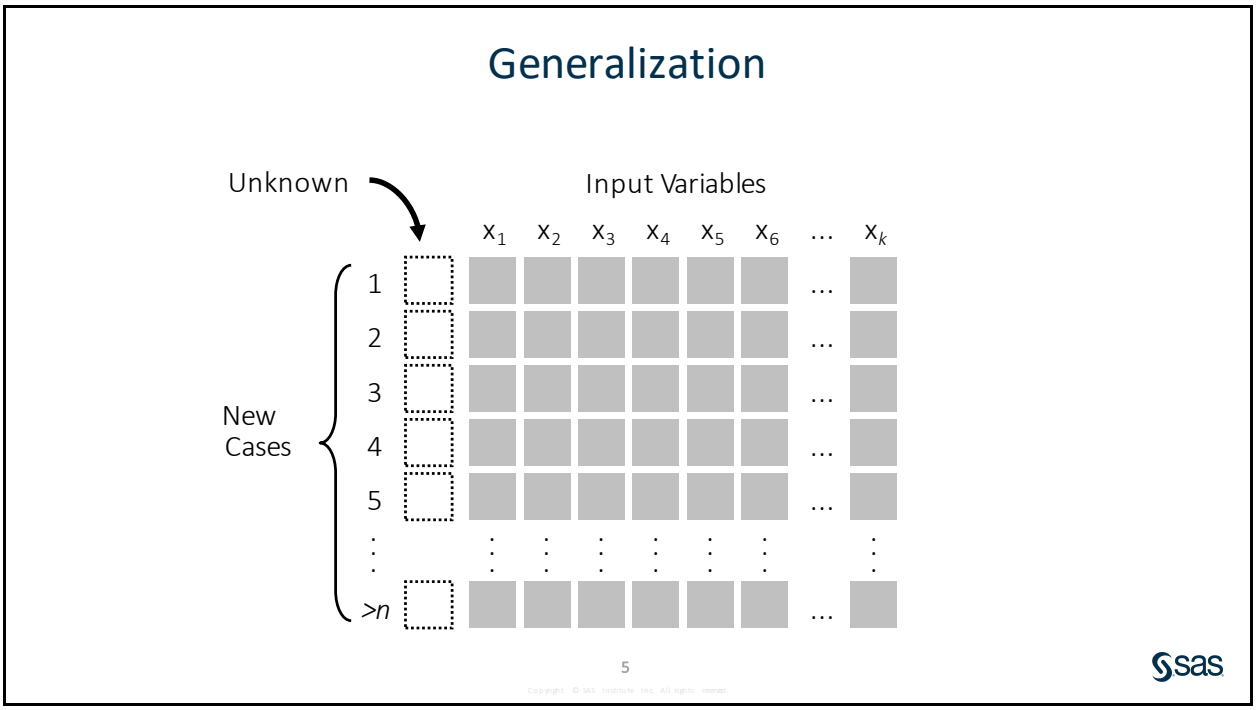


4

The data used to develop a predictive model consists of a set of *cases* (*observations, examples*). Associated with each case is a vector of *input variables* (*predictors, explanatory variables, features*) and a *target variable* (*outcome, response*). A predictive model maps the vector of input variables to the target. The target is the outcome to be predicted. The cases are the units on which the prediction is made.

In *supervised classification* (Hand 1997), the target is a class label. A predictive model assigns, to each case, a score (or a set of scores) that measures the propensity that the case belongs to a particular class. When there are only two possible classes of outcomes, you refer to the target as *binary*. Often, you think of one class in terms of the occurrence of a particular event (such as response to an offer) and the other class is the complement (not responding to the offer).

The term *supervised* is used when the class label is known for each case. If the label is known, then why build a predictive model?



The predictive model is used on new cases where the values of the input variables are known, but the class labels are unknown. The principal aim of predictive modeling is generalization. *Generalization* means the ability to predict the outcome on novel cases.

In contrast, the principal aim of traditional statistical analysis is inference. Confidence intervals, hypothesis tests, and *p*-values are the common inferential tools. Similar methods used by predictive modelers (such as logistic regression) might be used to infer how input variables affect the target. The validity of the inference relies on understanding the statistical properties of methods and applying them correctly.

Understanding the relationships between variables can be important in predictive modeling as well. However, in many practical settings, the discovery of structure is informal and exploratory. The validity of predictive modeling methods is assessed empirically. If a model generalizes well, then this might be the only thing that is considered important. Indeed, some predictive modeling methods (such as neural nets) are difficult to interpret and are not estimated using formal inferential methods, but they are still found to be highly useful.

Applications

Target Marketing

Attrition Prediction

Credit Scoring

Fraud Detection

6

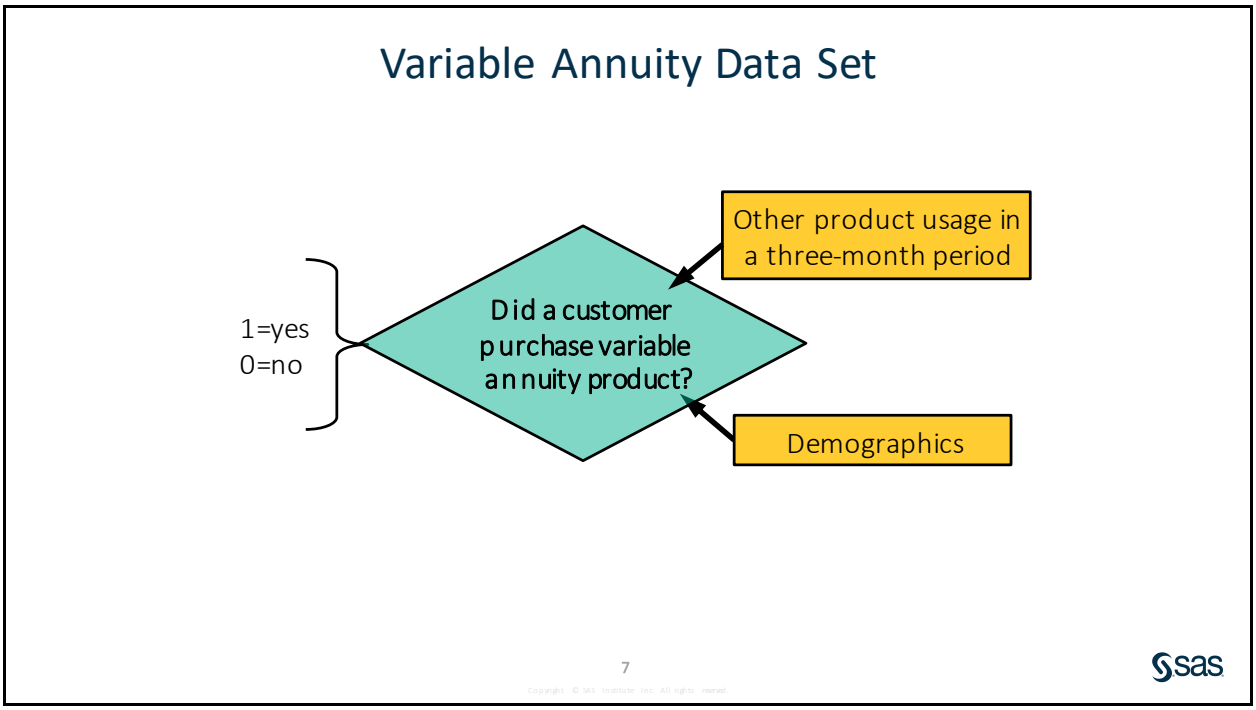
There are many business applications of predictive modeling. *Database marketing* uses customer databases to improve sales promotions and product loyalty. In target marketing, the cases are customers, the inputs are attributes such as previous purchase history and demographics, and the target is often a binary variable indicating a response to a past promotion. The aim is to find segments of customers that are likely to respond to some offer so that they can be targeted.

Historic customer databases can also be used to predict who is likely to switch brands or cancel services (churn). Loyalty promotions can then be targeted at new cases that are at risk.

Credit scoring is used to decide whether to extend credit to applicants. The cases are past applicants. Most input variables come from the credit application or credit reports. A relevant binary target is whether the case defaulted (charged-off) or paid the debt. The aim is to reduce defaults and serious delinquencies on new applicants for credit.

In fraud detection, the cases are transactions (for example, telephone calls, credit card purchases) or insurance claims. The inputs are the particulars and circumstances of the transaction. The binary target is whether that case was fraudulent. The aim is to anticipate fraud or abuse on new transactions or claims so that they can be investigated or impeded.

Supervised classification also has less business-oriented uses. Image classification has applications in areas such as astronomy, nuclear medicine, and molecular genetics (McLachlan 1992; Ripley 1996; Hand 1997).



Example: A target marketing campaign for a bank was undertaken to identify segments of customers who are likely to respond to a variable annuity (insurance product) marketing campaign. A data set was created with 32,264 cases (banking customers) and 47 input variables. The binary target variable, **Ins**, indicates whether the customer bought the variable annuity product. The 47 input variables represent other product usage in a three-month period and demographics. Two of the inputs are nominally scaled. The others are interval or binary. The data are in a SAS data set called **develop**.

These are the variables in the data set:

Ins	purchase variable annuity account (**1**=**yes**, **0**=**no**)
AcctAge	age of oldest account in years
DDA	checking account (**1**=**yes**, **0**=**no**)
DDABal	checking account balance
Dep	number of checking deposits
DepAmt	amount deposited
CashBk	number of times customer received cash back
Checks	number of checks
DirDep	direct deposit (**1**=**yes**, **0**=**no**)
NSF	occurrence of insufficient funds (**1**=**yes**, **0**=**no**)
NSFAmt	amount of insufficient funds
Phone	number of times a customer used telephone banking
Teller	number of teller visits
Sav	savings account (**1**=**yes**, **0**=**no**)

SavBal	savings balance
ATM	used ATM service (**1**=**yes**, **0**=**no**)
ATMAmt	ATM withdrawal amount
POS	number of point of sale transactions
POSAmt	amount in point of sale transactions
CD	has certificate of deposit (**1**=**yes**, **0**=**no**)
CDBal	certificate of deposit balance
IRA	has retirement account (**1**=**yes**, **0**=**no**)
IRABal	retirement account balance
LOC	has line of credit (**1**=**yes**, **0**=**no**)
LOCBal	line of credit balance
Inv	has investment account (**1**=**yes**, **0**=**no**)
InvBal	investment account balance
ILS	has installment loan (**1**=**yes**, **0**=**no**)
ILSBal	installment loan balance
MM	has money market account (**1**=**yes**, **0**=**no**)
MMBal	money market balance
MMCred	number of money market credits
MTG	has mortgage account (**1**=**yes**, **0**=**no**)
MTGBal	mortgage balance
CC	has credit card account (**1**=**yes**, **0**=**no**)
CCBal	credit card balance
CCPurc	number of credit card purchases
SDB	has a safety deposit box (**1**=**yes**, **0**=**no**)
Income	income in thousands of dollars
HMOwn	owns home (**1**=**yes**, **0**=**no**)
LORes	length of residence in years
HMVal	home value in thousands of dollars
Age	age of customer in years
CRScore	credit score
Moved	recent address change (**1**=**yes**, **0**=**no**)
InArea	local address (**1**=**yes**, **0**=**no**)
Res	area classification (R=rural, S=suburb, U=urban)
Branch	branch of bank (B1 – B19)

Exploring the Data

Example: Examine the **develop** data set by generating descriptive statistics and frequency tables.

The DATA step reads in the original data and creates a data set in the **Work** library. This prevents writing over the original data.

```
/* pmlr01d01.sas */
/* For this demonstration program you must have run the initialization
   program pmlr00d01.sas */

data work.develop;
   set pmlr.develop;
run;
```

The %LET statement enables you to define a macro variable and assign it a value. The statement below creates a macro variable named **inputs** and assigns it a string that contains all the numeric input variable names. This reduces the amount of text that you need to enter in other programs.

```
%global inputs;
%let inputs=acctage dda ddabal dep depamt cashbk checks
            dirdep nsf nsfamt phone teller atm atmamt pos posamt
            cd cdbal ira irabal loc locbal inv invbal ils ilsbal
            mm mmbal mmcred mtg mtgbal sav savbal cc ccbal
            ccpurc sdb income hmown lores hmval age crscore
            moved inarea;
```

The MEANS procedure generates descriptive statistics for the numeric variables. The statistics requested below are the number of observations, the number of missing values, the mean, the minimum value, and the maximum value. The macro variable, **inputs**, is referenced in the VAR statement by prefixing an ampersand (&) to the macro variable name. The FREQ procedure examines the values of the target variable and the nominal input variables.

```
proc means data=work.develop n nmiss mean min max;
   var &inputs;
run;

proc freq data=work.develop;
   tables ins branch res;
run;
```

Variable	Label	N	N Miss	Mean	Minimum	Maximum
AcctAge	Age of Oldest Account	30194	2070	5.9086772	0.3000000	61.5000000
DDA	Checking Account	32264	0	0.8156459	0	1.0000000
DDABal	Checking Balance	32264	0	2170.02	-774.8300000	278093.83
Dep	Checking Deposits	32264	0	2.1346082	0	28.0000000
DepAmt	Amount Deposited	32264	0	2232.76	0	484893.67
CashBk	Number Cash Back	32264	0	0.0159621	0	4.0000000
Checks	Number of Checks	32264	0	4.2599182	0	49.0000000
DirDep	Direct Deposit	32264	0	0.2955616	0	1.0000000

NSF	Number Insufficient Fund	32264	0	0.0870630	0	1.0000000
NSFAmt	Amount NSF	32264	0	2.2905464	0	666.8500000
Phone	Number Telephone Banking	28131	4133	0.4056024	0	30.0000000
Teller	Teller Visits	32264	0	1.3652678	0	27.0000000
Sav	Saving Account	32264	0	0.4668981	0	1.0000000
SavBal	Saving Balance	32264	0	3170.60	0	700026.94
ATM	ATM	32264	0	0.6099368	0	1.0000000
ATMAmt	ATM Withdrawal Amount	32264	0	1235.41	0	427731.26
POS	Number Point of Sale	28131	4133	1.0756816	0	54.0000000
POSAmt	Amount Point of Sale	28131	4133	48.9261782	0	3293.49
CD	Certificate of Deposit	32264	0	0.1258368	0	1.0000000
CDBal	CD Balance	32264	0	2530.71	0	1053900.00
IRA	Retirement Account	32264	0	0.0532792	0	1.0000000
IRABal	IRA Balance	32264	0	617.5704550	0	596497.60
LOC	Line of Credit	32264	0	0.0633833	0	1.0000000
LOCBal	Line of Credit Balance	32264	0	1175.22	-613.0000000	523147.24
Inv	Investment	28131	4133	0.0296826	0	1.0000000
InvBal	Investment Balance	28131	4133	1599.17	-2214.92	8323796.02
ILS	Installment Loan	32264	0	0.0495909	0	1.0000000
ILSBal	Loan Balance	32264	0	517.5692344	0	29162.79
MM	Money Market	32264	0	0.1148959	0	1.0000000
MMBal	Money Market Balance	32264	0	1875.76	0	120801.11
MMCred	Money Market Credits	32264	0	0.0563786	0	5.0000000
MTG	Mortgage	32264	0	0.0493429	0	1.0000000
MTGBal	Mortgage Balance	32264	0	8081.74	0	10887573.28
CC	Credit Card	28131	4133	0.4830969	0	1.0000000
CCBal	Credit Card Balance	28131	4133	9586.55	-2060.51	10641354.78
CCPurc	Credit Card Purchases	28131	4133	0.1541716	0	5.0000000
SDB	Safety Deposit Box	32264	0	0.1086660	0	1.0000000
Income	Income	26482	5782	40.5889283	0	233.0000000
HMOwn	Owns Home	26731	5533	0.5418802	0	1.0000000
LORes	Length of Residence	26482	5782	7.0056642	0.5000000	19.5000000
HMVal	Home Value	26482	5782	110.9121290	67.0000000	754.0000000
Age	Age	25907	6357	47.9283205	16.0000000	94.0000000
CRScore	Credit Score	31557	707	666.4935197	509.0000000	820.0000000
Moved	Recent Address Change	32264	0	0.0296305	0	1.0000000
InArea	Local Address	32264	0	0.9602963	0	1.0000000

The results PROC MEANS results show that several variables have missing values and several variables are binary.

Ins	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	21089	65.36	21089	65.36
1	11175	34.64	32264	100.00

Branch of Bank

Branch	Frequency	Percent	Cumulative Frequency	Cumulative Percent
B1	2819	8.74	2819	8.74
B10	273	0.85	3092	9.58
B11	247	0.77	3339	10.35
B12	549	1.70	3888	12.05
B13	535	1.66	4423	13.71
B14	1072	3.32	5495	17.03

B15	2235	6.93	7730	23.96
B16	1534	4.75	9264	28.71
B17	850	2.63	10114	31.35
B18	541	1.68	10655	33.02
B19	285	0.88	10940	33.91
B2	5345	16.57	16285	50.47
B3	2844	8.81	19129	59.29
B4	5633	17.46	24762	76.75
B5	2752	8.53	27514	85.28
B6	1438	4.46	28952	89.73
B7	1413	4.38	30365	94.11
B8	1341	4.16	31706	98.27
B9	558	1.73	32264	100.00

Area Classification

Res	Frequency	Percent	Cumulative Frequency	Cumulative Percent
R	8077	25.03	8077	25.03
S	11506	35.66	19583	60.70
U	12681	39.30	32264	100.00

The results of PROC FREQ show that 34.6 percent of the observations acquired the insurance product. There are 19 levels of **Branch** and three levels under **Res** (**R**=rural, **S**=suburb, **U**=urban).

End of Demonstration

1.2 Analytical Challenges

Objectives

- Describe the analytical challenges that predictive modelers face.
- Offer some solutions to these challenges.

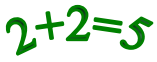
11

sas

Opportunistic Data

Operational / Observational

Massive



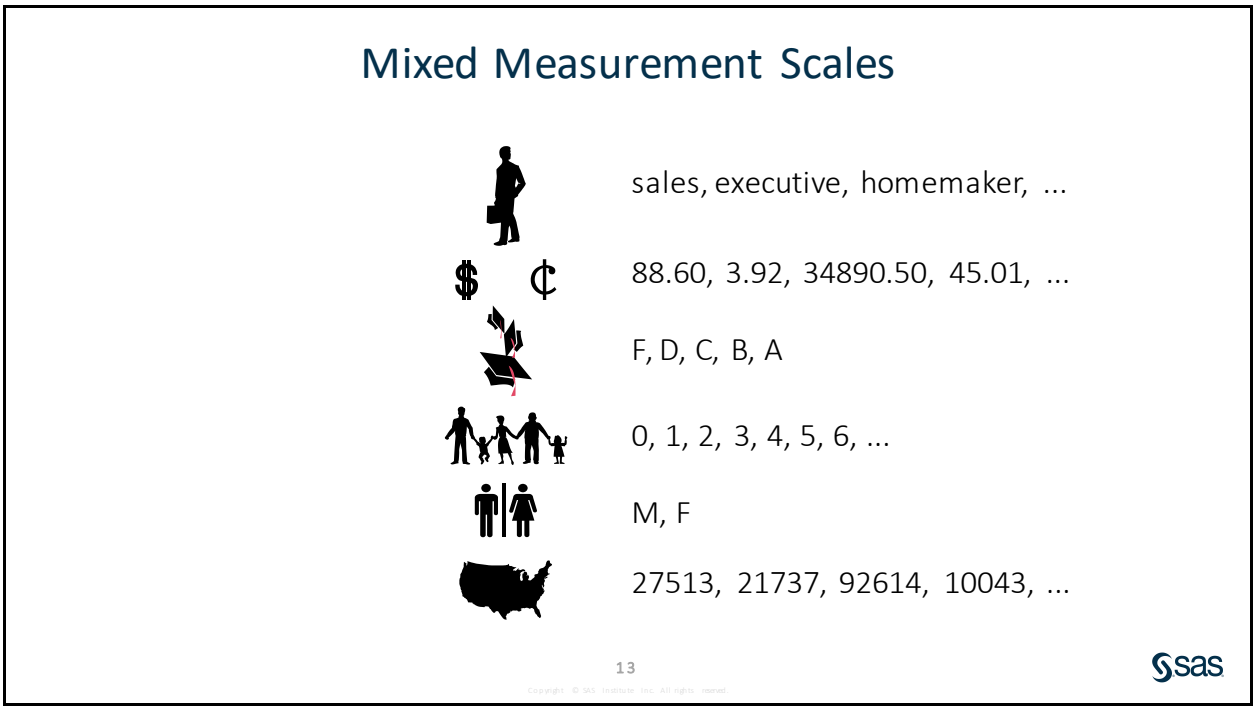
Errors and Outliers

Missing Values

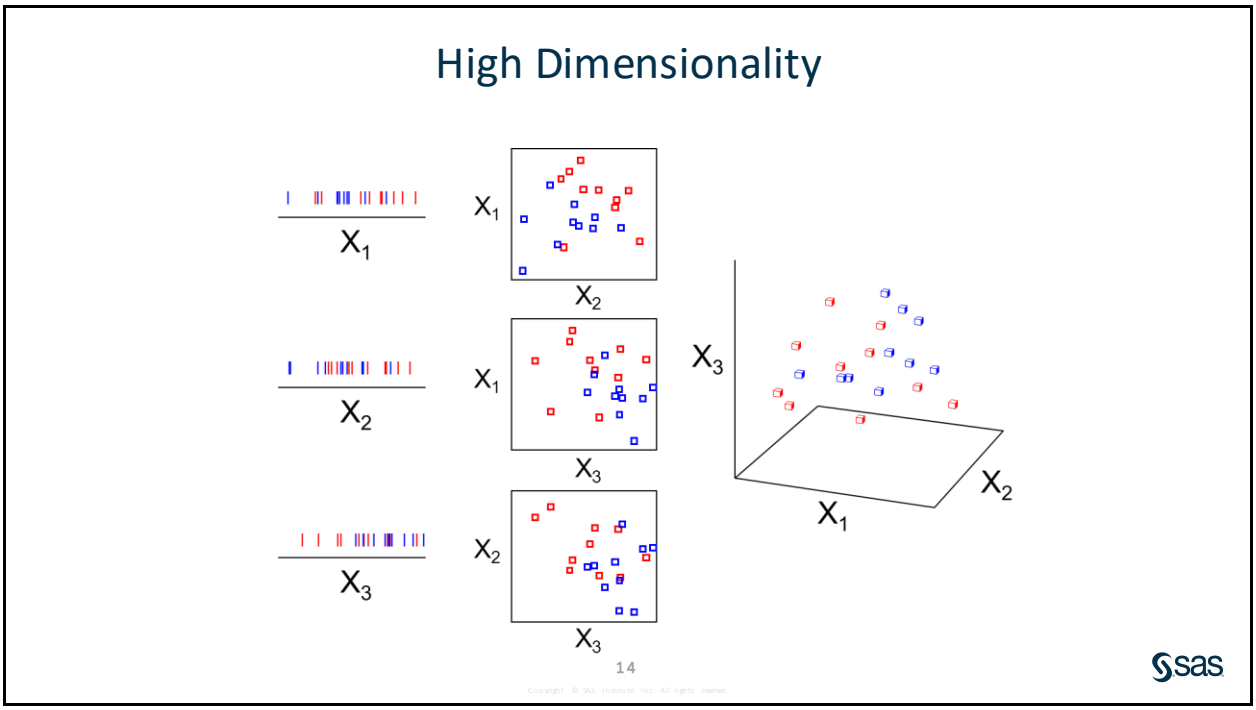
12

sas

The data that are typically used to develop predictive models can be characterized as *opportunistic*. The data were collected for operational purposes unrelated to statistical analysis. Such data are usually massive, dynamic, and dirty. Preparing data for predictive modeling is often very difficult. For example, the parts of the data that are relevant to the analysis need to be acquired. Furthermore, the relevant inputs usually need to be created from the raw operational fields. Many statistical methods do not scale well to massive data sets because most methods were developed for small data sets generated from designed experiments.



When there are large numbers of input variables, there is usually a variety of measurement scales represented. The input variable might be scaled (amounts) as an interval, binary, nominally scaled (names), ordinally scaled (grades), or counts. Nominal input variables with a large number of levels (such as
ZIP code) are commonplace and present complications for regression analysis.

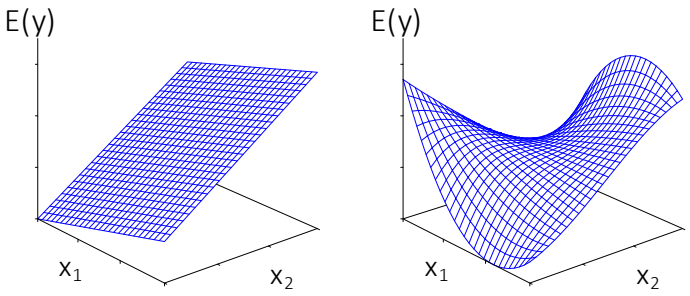


The *dimension* refers to the number of input variables (actually, input degrees of freedom). Predictive modelers consider large numbers (hundreds) of input variables. The number of variables often has a greater effect on computational performance than the number of cases. High dimensionality limits the ability to explore and model the relationships among the variables. This is known as the *curse of dimensionality*, which was defined by Breiman et al. (1984) as the following:

> *The complexity of a data set increases rapidly with increasing dimensionality.*

The remedy is dimension reduction, that is, ignore irrelevant and redundant dimensions without inadvertently ignoring important ones.

Nonlinearities and Interactions



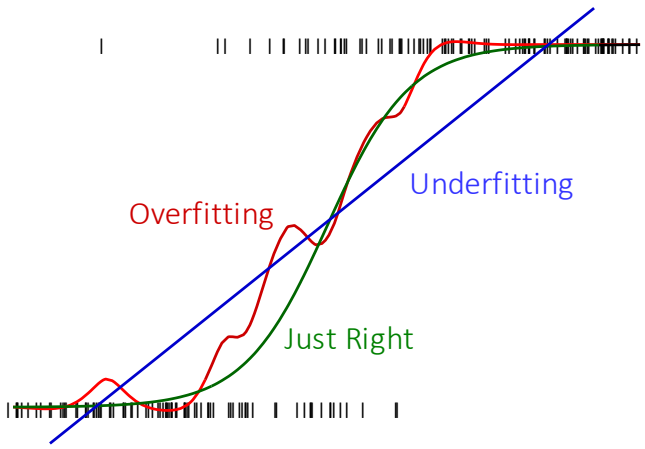
Linear
Additive

Nonlinear
Nonadditive

15

Predictive modeling is a multivariate problem. Each important dimension might affect the target in complicated ways. Moreover, the effect of each input variable might depend on the values of other input variables. The curse of dimensionality makes this difficult to untangle. Many classical modeling methods (including standard logistic regression) were developed for inputs with effects that have a constant rate of change and do not depend on any other inputs.

Model Selection



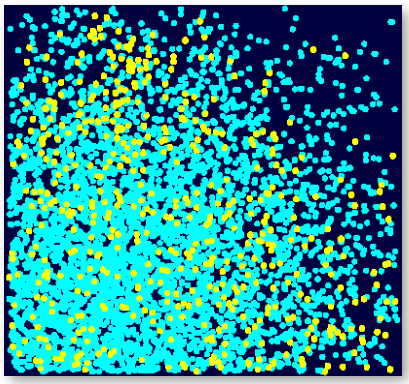
16

Predictive modeling typically involves choices from among a set of models. These might be different types of models. These might be different complexities of models of the same type. A common pitfall is to overfit the data (to use too complex a model). An overly complex model might be too sensitive to peculiarities in the sample data set and not generalize well to new data. However, using too simple a model can lead to *underfitting*, where true features are disregarded.

In predictive modeling, the event of interest is often rare. With rare events, the effective sample size for building a reliable prediction model is closer to three times the number of event cases than to the nominal size of the data set (Harrell 1997). A seemingly massive data set might have the predictive potential of one that is much smaller.

Outcome Overrepresentation

A common predictive modeling practice is to build models from a sample with a primary outcome proportion different from the original population.

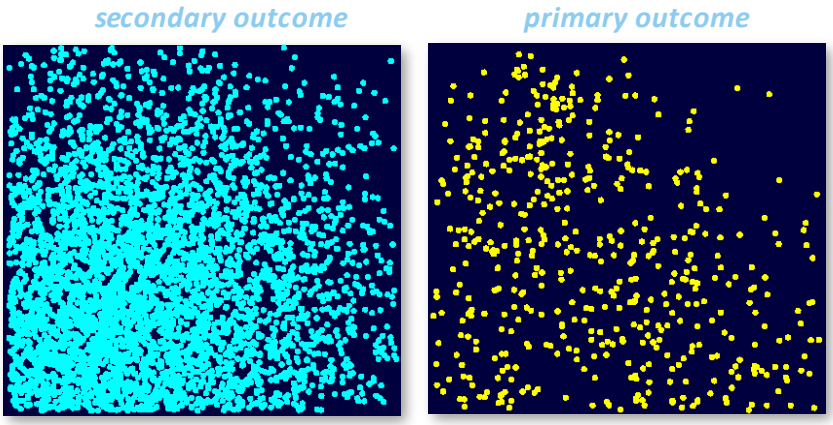


18

...

One widespread strategy for predicting rare events is to build a model on a sample that disproportionately over-represents the event cases (for example, an equal number of events and non-events). This is typically done when the ratio of events to non-events is very small.

Separate Sampling



Target-based samples are created by considering the primary outcome cases separately from the secondary outcome cases.

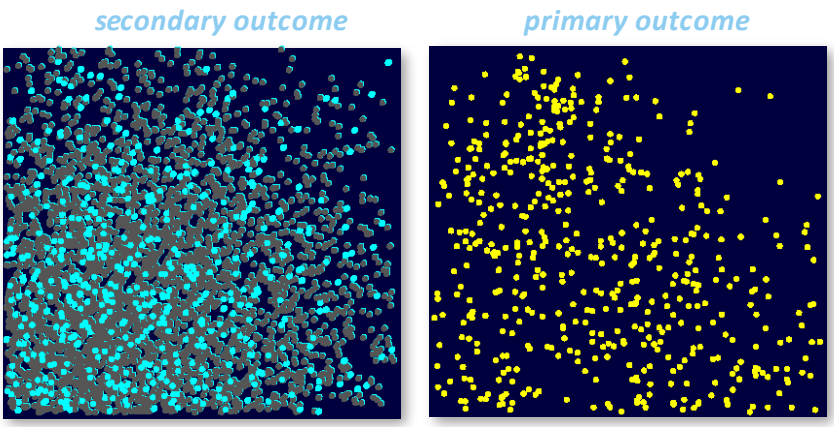
21

sas

...

Separate sampling derives its name from the technique that is used to generate the modeling data. In this case, samples are drawn separately based on the target outcome.

Separate Sampling



Select some cases. **Select all cases.**

22

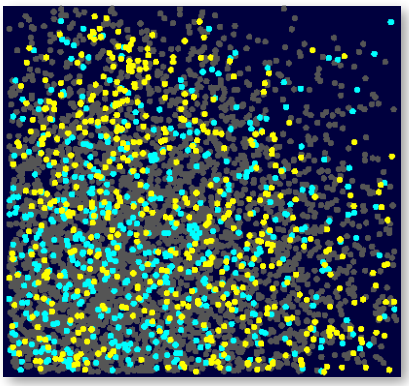
sas

...

In the case of the rare target event, usually all the target events and a random sample of the non-events are selected.

The Modeling Sample

+ Similar predictive power with smaller case count
- Must adjust assessment statistics and graphics
- Must adjust prediction estimates for bias



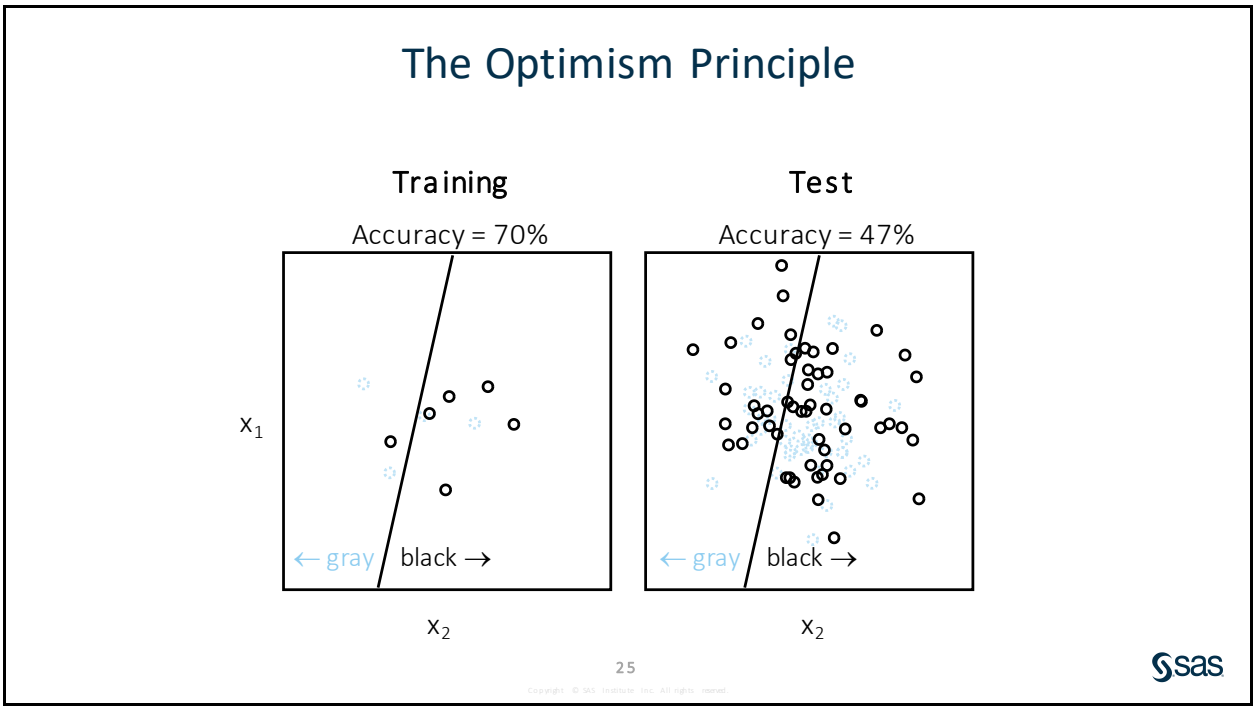
24

...

The advantage of separate sampling is that you can obtain (on the average) a model of similar predictive power with a smaller overall case count. This is in accordance with the idea that the amount of information in a data set with a categorical outcome is determined not by the total number of cases in the data set itself, but by the number of cases in the rarest outcome category (Harrell 2006).

This advantage might seem of minimal importance in the age of extremely fast computers. However, the model-fitting process occurs only after the completion of a long, tedious, and error-prone data preparation process. Therefore, oversampling rare events is generally done for data efficiency purposes because smaller sample sizes for data preparation are usually preferred.

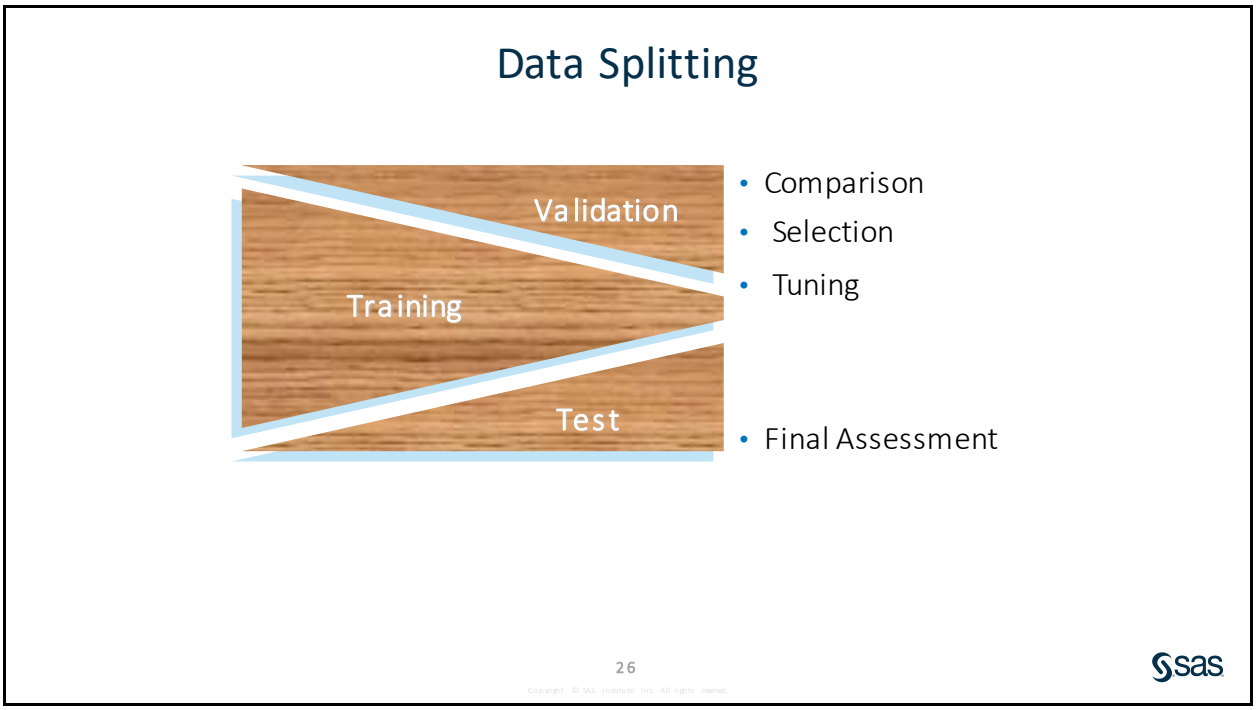
Although it reduces analysis time, separate sampling also introduces some analysis complications. (Biases and the adjustments for separate sampling are addressed in a later section.)



Evaluating the performance of a classifier on the same data that is used to train the classifier usually leads to an optimistically biased assessment. For example, the above classifier was fit (or more properly, overfit) to a 10-case data set. It correctly classified 70% of the cases. However, when the same classification rule was applied to 100 new cases from the same distribution, only 47% were correctly classified. This is called *overfitting*. The model was overly sensitive to peculiarities of the particular training data, in addition to the true features of their joint distribution.

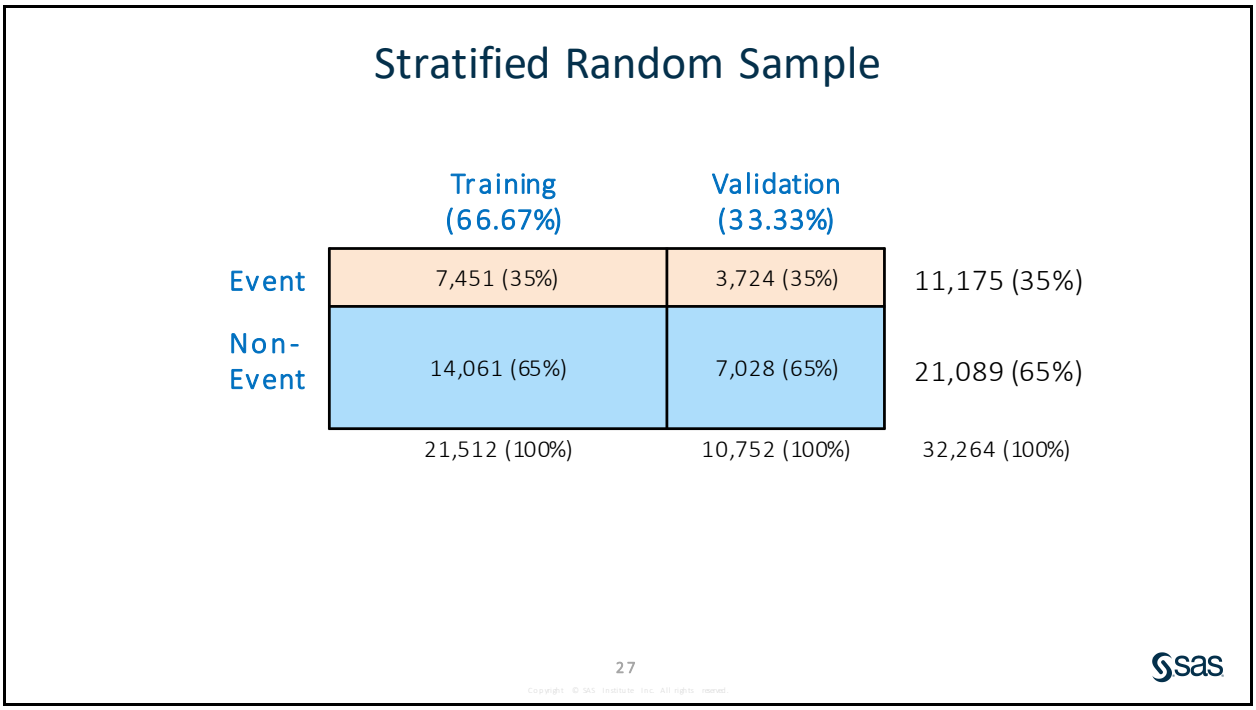
When the underlying model is more flexible and the data is less plentiful, overfitting becomes more of a problem. When a relatively inflexible model, such as a (linear) logistic regression, is fitted to massive amounts of data, overfitting might not be a problem (Hand 1997). However, the chance of overfitting is increased by variable selection methods and supervised input preparation (such as collapsing levels of nominal variables based on associations with the target). It is prudent to assume that overfitting is a problem until proven otherwise.

Large differences between the performance on the training and test data sets usually indicate overfitting.



The simplest strategy for correcting the optimism bias is to isolate a portion of the development data for assessment. The model is fit to the remainder (training data set) and performance is evaluated on the separated portion (test data set). Usually, from one-fourth to one-half of the development data is used as a test set (Picard and Berk 1990).

When the detached data are used for comparing, selecting, and tuning models, and the chosen model is assessed on the same data set that was used for comparison, then the optimism principle again applies. In this situation, the sample is more correctly called a validation data set, not a test set. The test set is used for a final assessment of a fully specified classifier (Ripley 1996). If model tuning and a final assessment are both needed, then the data should be split three ways into training, validation, and test sets. In some applications, the test set is gathered from a different time or location.



To ensure that an equal percentage of events are in the training and validation data sets, a stratified random sample could be selected. In stratified random sampling, you divide the observations into non-overlapping groups and then select a sample from each group. In this case, the strata are the target values, and the resulting samples are the training and validation data sets.

Data Splitting

Example: Split the data into training and validation data sets.

The SURVEYSELECT procedure can be used to select the records for the training and validation data sets. To create a stratified sample, the data must be sorted by the stratum variable. The SAMPRATE= option specifies what proportion of the develop data set should be selected. The default behavior of PROC SURVEYSELECT is to write the sample to output, not the entire data set, so the OUTALL option can be used to return the initial data set augmented by a flag to indicate selection in the sample. Of course, this flag indicates membership in the training and validation data sets in this context. The SEED= option enables the user to control what series of pseudo-random numbers is generated to do the partitioning. A particular number, greater than zero, produces the same split each time PROC SURVEYSELECT is run. If the seed is zero, then the data are split differently each time the procedure is run.

Beginning in SAS/STAT 14.1, PROC SURVEYSELECT uses a different method to determine stratum initial seeds for the Mersenne Twister random number generator. STRATUMSEED=RESTORE reproduces the stratum initial seeds that PROC SURVEYSELECT uses in releases prior to SAS/STAT 14.1 for the Mersenne Twister random number generator. To reproduce a stratified sample that PROC SURVEYSELECT selects in releases prior to SAS/STAT 14.1, you can specify STRATUMSEED=RESTORE along with the same sample selection parameters and input data set.

```
/*  pmlr01d02.sas  */

proc sort data=work.develop out=work.develop_sort;
   by ins;
run;

proc surveyselect noprint data = work.develop_sort samprate=.6667
     out=work.develop_sample seed=44444 outall stratumseed=restore;
   strata ins;
run;
```

PROC FREQ verifies the stratification.

```
proc freq data = work.develop_sample;
   tables ins*selected;
run;
```

Table of Ins by Selected

Ins
Selected(Selection Indicator)

Frequency Percent Row Pct Col Pct	0	1	Total
0	7028 21.78 33.33 65.36	14061 43.58 66.67 65.36	21089 65.36
1	3724 11.54 33.32 34.64	7451 23.09 66.68 34.64	11175 34.64
Total	10752 33.33	21512 66.67	32264 100.00

Notice that the proportion of responders is the same for both levels of selected, and the data are partitioned into two-thirds for training and one-third for validation.

The next DATA step creates the two data sets, **train** and **valid**. Several variables that are not needed in the analysis are dropped.

```
data work.train(drop=selected SelectionProb SamplingWeight)
     work.valid(drop=selected SelectionProb SamplingWeight);
   set work.develop sample;
   if selected then output work.train;
   else output work.valid;
run;
```

End of Demonstration

SAS Studio

SAS Studio is the new browser-based SAS programming environment.

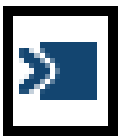
30

SAS Studio is the new browser-based SAS programming environment that you can use for data exploration and analysis. A tutorial about SAS Studio can be found at the following site: https://support.sas.com/training/tutorial/studio/get-started.html.

Interactive Mode

Some SAS procedures, such as PROC GLM, are interactive. That means that they remain active until you submit a QUIT statement, or until you submit a new PROC or DATA step.

In SAS Studio, you can use the code editor to run these procedures, as well as other SAS procedures, in interactive mode.



By default, SAS Studio does not run in interactive mode.
This icon in SAS Studio toggles interactive mode on and off.

31

Some procedures, such as PROC GLM, are interactive. This means that they remain active until you submit a QUIT statement or a new PROC or DATA step. You can use the code editor to run these procedures interactively in SAS Studio. However, you must use the icon shown above to enable and activate the Interactive mode.

Considerations for Running in Interactive Mode

- Interactive mode starts a new SAS session.
- Librefs and macro variables used in the course must be defined for each new SAS session.

SAS Studio Documentation:

http://support.sas.com/software/products/sasstudio/#s1=2

Running SAS Studio in Interactive mode starts a new SAS session. This means that library references and macro variables must be defined for each new session.

Exercises

A national veterans' organization seeks to better target its solicitations for donation. By soliciting only the most likely donors, less money is spent on solicitation efforts and more money is available for charitable concerns. Solicitations involve sending a small gift to an individual with a request for donation. Gifts include mailing labels and greeting cards.

The organization has more than 3.5 million individuals in its mailing database. These individuals were classified by their response behaviors to previous solicitation efforts. Of particular interest is the class of individuals who were identified as lapsing donors. These individuals made their most recent donations between 12 and 24 months ago. The organization found that by predicting the response behavior of this group, they could use the model to rank all 3.5 million individuals in their database. With this ranking, a decision can be made to either solicit or ignore an individual in the current solicitation campaign. The current campaign refers to a greeting card mailing sent in June of 1997. It is identified in the raw data as the 97NK campaign.

The raw analysis data were reduced for this course. A subset of slightly more than 19,000 records was selected for modeling. This subset was not chosen arbitrarily. In addition, the 481 fields were reduced to 47. Some of the fields were eliminated when their potential association with the analysis objective was considered. (For example, it is doubtful that CD player ownership is strongly correlated with donation potential.) Other fields were combined to form summaries of a specific customer behavior. In general, if the variable has **PROM** in its name, then the variable is related to the number of items that the organization sent to the customer. However, if the variable has **GIFT** in its name, then the variable is related to the amount of money that the customer sent to the organization.

The data are in a data set called **pva_raw_data**. The following table details the variables and descriptions:

Variable	Description
CARD_PROM_12	Count of card promotions in the past 12 months
CLUSTER_CODE	Socio-Economic Cluster Code
CONTROL_NUMBER	ID
DONOR_AGE	Donor Age
DONOR_GENDER	Donor Gender
FREQUENCY_STATUS_97NK	Count of Donations between June 1995 and June 1996 (capped at 4)
HOME_OWNER	Home Owner flag
INCOME_GROUP	Income Bracket, from 1 to 7
IN_HOUSE	Flag for *In-House* donor program
LAST_GIFT_AMT	Amount of most recent donation

Variable	Description
LIFETIME_AVG_GIFT_AMT	Average donation amount, ever
LIFETIME_CARD_PROM	Number of card promotions, ever
LIFETIME_GIFT_AMOUNT	Total donation amount, ever
LIFETIME_GIFT_COUNT	Total number of donations, ever
LIFETIME_GIFT_RANGE	Maximum gift amount less minimum gift amount
LIFETIME_MAX_GIFT_AMT	Maximum gift amount, ever
LIFETIME_MIN_GIFT_AMT	Minimum gift amount, ever
LIFETIME_PROM	Count of solicitations ever sent
MEDIAN_HOME_VALUE	Census data
MEDIAN_HOUSEHOLD_INCOME	Census data
MONTHS_SINCE_FIRST_GIFT	Months since first donation
MONTHS_SINCE_LAST_GIFT	Months since most recent donation
MONTHS_SINCE_ORIGIN	Months since entry onto the file
MOR_HIT_RATE	Data recorded by a third-party Mail-Order Response rate
NUMBER_PROM_12	Count of promotions in the past 12 months
OVERLAY_SOURCE	Source of Demographic overlay
PCT_MALE_MILITARY	Census data
PCT_MALE_VETERANS	Census data
PCT_OWNER_OCCUPIED	Census data
PCT_VIETNAM_VETERANS	Census data
PCT_WWII_VETERANS	Census data
PEP_STAR	Flag to identify consecutive donors
PER_CAPITA_INCOME	Census data
PUBLISHED_PHONE	Flag
RECENCY_STATUS_96NK	Categorization of donation patterns
RECENT_AVG_CARD_GIFT_AMT	Average donation amount to card promotions since June 1994

Variable	Description
RECENT_AVG_GIFT_AMT	Average donation amount to promotions since June 1994
RECENT_CARD_RESPONSE_COUNT	Count of responses to card promotions since June 1994
RECENT_CARD_RESPONSE_PROP	Proportion of responses to card promotions since June 1994
RECENT_RESPONSE_COUNT	Count of responses to promotions since June 1994
RECENT_RESPONSE_PROP	Proportion of responses to promotions since June 1994
RECENT_STAR_STATUS	STAR status flag, since June 1994
SES	A clustering of the levels of **CLUSTER_CODE**
TARGET_B	B=Binary, flag for response to 97NK—Target Variable
TARGET_D	Dollar amount of response to 97NK
URBANICITY	Categorization of residency
WEALTH_RATING	Measures wealth relative to others within state

1. Data Splitting

Submit the program **pmlr00e01.sas**. Use PROC SURVEYSELECT to split the **pva** data set into training and validation data sets. Use 50% of the data for each data set role. Stratify on the target variable, use a seed of *27513*, and use the STRATUMSEED=RESTORE option.

How many observations are in the training data set?

End of Exercises

1.3 Chapter Summary

The principal objective of predictive modeling is to predict the outcome on new cases. Some of the business applications include target marketing, attrition prediction, credit scoring, and fraud detection. One challenge in building a predictive model is that the data usually were not collected for purposes of data analysis. Therefore, the data are usually massive, dynamic, and dirty. For example, the data usually have large numbers of input variables. This limits the ability to explore and model the relationships among the variables. Thus, detecting interactions and nonlinearities becomes a cumbersome problem.

When the target is rare, a widespread strategy is to build a model on a sample that disproportionately over-represents the events. The results are biased, but they can be easily corrected to represent the population.

A common pitfall in building a predictive model is to overfit the data. An overfit model is too sensitive to the nuances in the data and does not generalize well to new data. However, a model that underfits the data systematically misses the true features in the data. One of the most important steps in predictive modeling is to assess the performance of the model. To correct for the optimism bias, a common strategy is to isolate a portion of the development data for assessment.

1.4 Solutions

Solutions to Exercises

1. Data Splitting

Submit the program **pmlr00e01.sas**. Use PROC SURVEYSELECT to split the **pva** data set into training and validation data sets. Use 50% of the data for each data set role. Stratify on the target variable, use a seed of *27513*, and use the STRATUMSEED=RESTORE option.

Note: An electronic copy of the solution program is in **pmlr01s01.sas**.

```
proc sort data=pmlr.pva out=work.pva_sort;
   by target_b;
run;

proc surveyselect noprint data=work.pva_sort samprate=0.5
     out=work.pva sample seed=27513 outall stratumseed=restore;
   strata target_b;
run;

data pmlr.pva train(drop=selected SelectionProb SamplingWeight)
     pmlr.pva_valid(drop=selected SelectionProb SamplingWeight);
   set work.pva_sample;
   if selected then output pmlr.pva_train;
   else output pmlr.pva_valid;
run;
```

```
NOTE: There were 19372 observations read from the data set WORK.PVA_SAMPLE.
NOTE: The data set PMLR.PVA_TRAIN has 9687 observations and 59 variables.
NOTE: The data set PMLR.PVA_VALID has 9685 observations and 59 variables.
```

How many observations are in the training data set? **There are 9687 observations in the training data set.**

End of Solutions

Chapter 2 Fitting the Model

2.1 The Model

Objectives

- Define the concepts of logistic regression.
- Fit a logistic regression model in the LOGISTIC procedure.
- Score new cases in PROC LOGISTIC.

3

Functional Form

posterior probability

$$\text{logit}(p_i) = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_k x_{ki}$$

parameter

input

4

The data set consists of i=1,2,…,n cases. Each case belongs to one of two classes. A binary indicator variable represents the class label for each case.

$$y_i = \begin{cases} 1 & \text{target event for case } i \\ 0 & \text{non-event for case } i \end{cases}$$

Associated with each case is k-vectors of input variables.

$$\mathbf{x}_i = (x_{1i}, x_{2i}, \ldots, x_{ki})$$

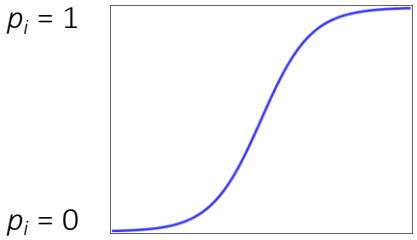
The posterior probability of the target event given the inputs is calculated by the following formula:

$$p_i = E(y_i \mid \mathbf{x}_i) = \Pr(y_i = 1 \mid \mathbf{x}_i)$$

The standard logistic regression model assumes that the logit of the posterior probability is a linear combination of the input variables. The parameters, $\beta_0, \ldots, \beta_k$, are unknown constants that must be estimated from the data.

The Logit Link Function

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \eta \quad \Leftrightarrow \quad p_i = \frac{1}{1+e^{-\eta}}$$



5

sas

A linear combination can take any value. The probability must be between zero and one. The logit transformation (which is the log of the odds) is a device for constraining the posterior probability to be between zero and one. The logit function transforms the probability scale to the real line $(-\infty, +\infty)$. Therefore, modeling the logit with a linear combination gives estimated probabilities that are constrained to be between zero and one.

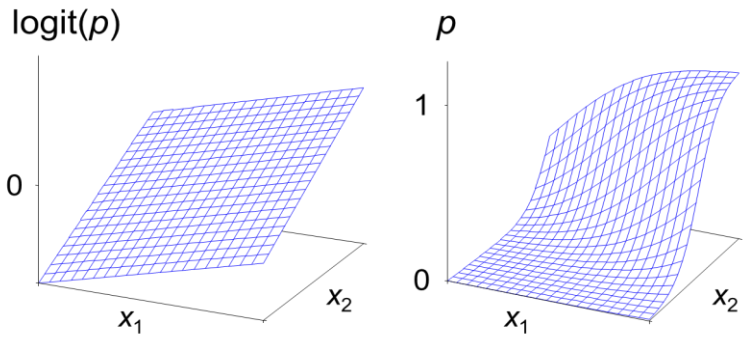
Logistic regression is a special case of the *generalized linear model*:

$$g(E(y \mid \mathbf{x})) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$$

where the expected value of the target is linked to the linear predictor by the function $g(\bullet)$.

The link function depends on the scale of the target.

The Fitted Surface



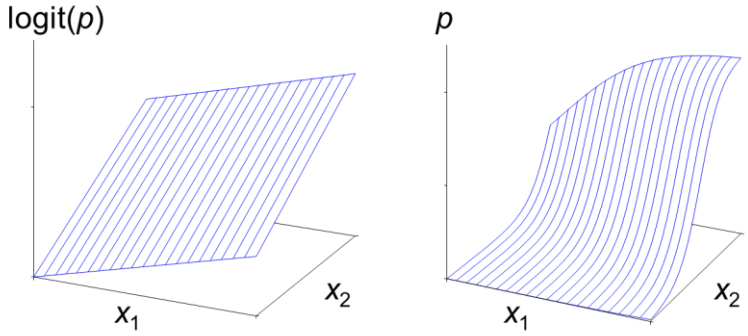
6

The graph of a linear combination on the logit scale is a hyperplane. Different parameter values give different surfaces with different slopes and different orientations.

On the probability scale, it becomes a sigmoidal surface. The nonlinearity of the model is used to deal with the constrained scale of the target.

Interpretation

Unit change in $x_2 \Rightarrow$



β_2 change in logit

100(exp(β_2)-1)% change in the odds

7

A linear-additive model is particularly easy to interpret because each input variable affects the logit linearly. The coefficients are the slopes. Exponentiating each parameter estimate gives the odds ratios, which compares the odds of the event in one group to the odds of the event in another group.

Odds Ratio from a Logistic Regression Model

Estimated logistic regression model:

$\text{logit}(p) = -.7567 + .4373*(\textbf{gender})$

where females are coded 1 and males are coded 0

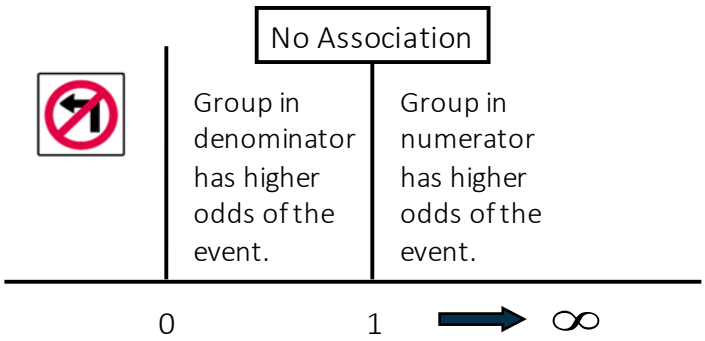
Estimated odds ratio (Females to Males):

$\text{odds ratio} = (e^{-.7567+.4373})/(e^{-.7567}) = 1.55$

8

The slide above shows how odds ratios are simple functions of the parameters. For example, suppose you want to examine the relationship between **gender** and the target variable. The odds ratio for **gender** compares the predicted odds of females to have the outcome to the predicted odds of males. If **gender** is coded 1 for females and 0 for males, the odds ratio is simply the exponentiation of the parameter estimate for **gender**. An odds ratio of 1.55 means that females have 1.55 times the odds of having the outcome compared to males.

Properties of the Odds Ratio

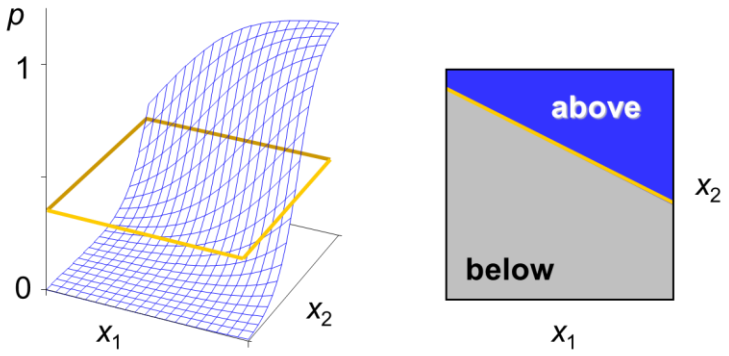


The odds ratio shows the strength of the association between the predictor variable and the response variable. If the odds ratio is 1, then there is no association between the predictor variable and the response. If the odds ratio is greater than 1, then the group in the numerator has higher odds of having the event. If the odds ratio is less than 1, then the group in the denominator has higher odds of having the event. For example, an odds ratio of 3 indicates that the odds of getting the event in the group in the numerator are three times that in the group in the denominator.

The odds ratio represents the multiplicative effect of each input variable. Moreover, the effect of each input variable does not depend on the values of the other inputs (additivity).

However, this simple interpretation depends on the model being correctly specified. In predictive modeling, you should not presume that the true posterior probability has such a simple form. Think of the model as an approximating hyperplane. Consequently, you can determine the extent that the inputs are important in representing the approximating hyperplane.

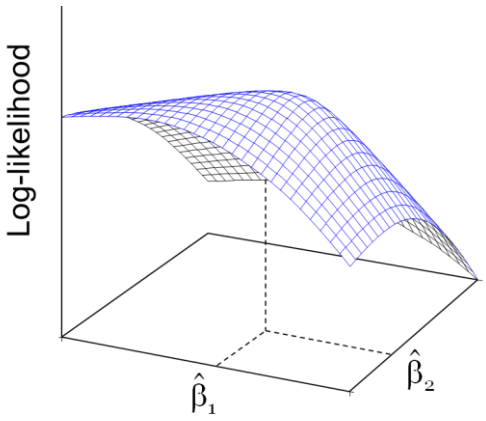
Logistic Discrimination



10

In supervised classification, the ultimate use of logistic regression is to allocate cases to classes. This is more correctly termed *logistic discrimination* (McClachlan 1989). An allocation rule is merely an assignment of a cutoff probability, where cases above the cutoff are allocated to class 1 and cases below the cutoff are allocated to class 0. The standard logistic discrimination model separates the classes by a linear surface (hyperplane). The decision boundary is always linear. Determining the best cutoff is a fundamental concern in logistic discrimination.

Maximum Likelihood Estimation



11

The method of maximum likelihood (ML) is usually used to estimate the unknown parameters in the logistic regression model. The likelihood function is the joint probability density function of the data treated as a function of the parameters. The maximum likelihood estimates are the values of the parameters that maximize the probability of obtaining the sample data.

It is usually computationally more convenient to work with the log of the likelihood rather than the likelihood itself. The parameter estimates that maximize the log of the likelihood also maximize the likelihood. Then, if you assume that the y_i values independently have Bernoulli distributions (the probability distribution of a random variable with only two possible values, usually 0 and 1) with probability p_i (which is a function of the parameters), then the log of the likelihood is given by the following formula:

$$\sum_{i=1}^{n}\left(y_i \ln(p_i)+(1-y_i)\ln(1-p_i)\right) \quad = \quad \sum_{i|y=1}^{n_1}\ln(p_i)+\sum_{i|y=0}^{n_0}\ln(1-p_i)$$

where n_0 and n_1 are the numbers of class 0 and class 1, respectively. The form of the log-likelihood shows the intuitively appealing result that the ML estimates be chosen so that p_i is large when y_i=1 and small when y_i=0.

In ML estimation, the combination of parameter values that maximize the likelihood (or log-likelihood) is pursued. There is, in general, no closed-form analytical solution for the ML estimates, as there is for linear regression with normally distributed response error. They must be determined using an iterative optimization algorithm. Consequently, logistic regression is considerably more computationally expensive than linear regression.

Software for ML estimation of the logistic model is commonplace. Many SAS procedures can be used. Most notable are the LOGISTIC, GENMOD, CATMOD, and DMREG procedures (in SAS Enterprise Miner).

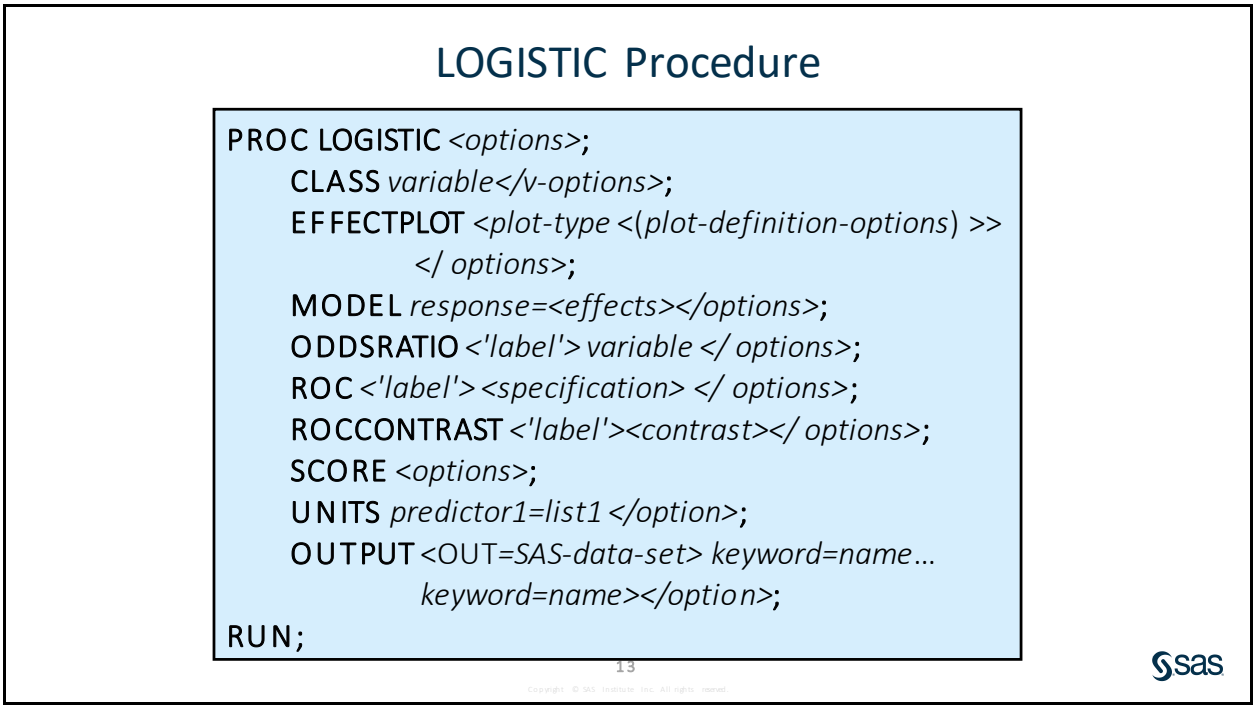
Concordant versus Discordant

		Customer Bought Variable Annuity Product	
	Predicted Outcome Probability	High	Low
Customer Did Not Buy Variable Annuity Product	High	Tie	Discordant Pair
	Low	Concordant Pair	Tie

12

In logistic regression, several measures that assess the predictive accuracy of the model are reported. These measures are calculated from the percent concordant, discordant, and tied pairs. For all possible pairs of observations with different outcomes (in this example, buying variable annuity products versus not buying variable annuity products), a comparison is made of the predicted outcome probabilities.

If the observation with the outcome has a higher predicted positive outcome probability compared to an observation with the negative outcome, the pair is *concordant*. However, if the observation with the positive outcome has a lower predicted outcome probability compared to the predicted outcome probability of an observation with the negative outcome, the pair is *discordant*. If the predicted outcome probabilities are the same, the pair is *tied*.



The LOGISTIC procedure fits logistic regression models for binary, ordinal, or nominal response data.

Selected LOGISTIC procedure statements:

CLASS	specifies the classification variables to be used in the analysis. The CLASS statement must precede the MODEL statement.
EFFECTPLOT	produces a display of the fitted model and provides options for changing and enhancing the displays.
MODEL	specifies the response variable and the predictor variables (which can be character or numeric). The MODEL statement is required, and only one is allowed with each invocation of PROC LOGISTIC.
ODDSRATIO	produces odds ratios for variables even when the variables are involved in interactions with other covariates, and for classification variables that use any parameterization. You can specify several ODDSRATIO statements.
ROC	specifies models to be used in the ROC comparisons. You can specify more than one ROC statement. ROC statements are identified by their label.

ROCCONTRAST	compares the different ROC models. You can specify only one ROCCONTRAST statement.
SCORE	creates a data set that contains all the data in the DATA= data set together with posterior probabilities and as an option, prediction confidence intervals. You can specify several SCORE statements.
UNITS	enables you to obtain an odds ratio estimate for a specified change in a predictor variable. The unit of change can be a number, standard deviation (SD), or a number times the standard deviation (2*SD).

2.01 Multiple Choice Poll

The odds ratio for a $1000 increase in income is 1.074. What does this mean for every $1000 increase in income?

a. The probability of the event increases by 7.4%.
b. The logit increases by 7.4%.
c. The odds of the event increases by 7.4%.
d. The log of the odds of the event increases by 7.4%.

14

Introduction to the LOGISTIC Procedure

Example: Fit a logistic regression model to the **work.train** data set. Specify **Ins** as the target variable, and **DDA**, **DDABal**, **Dep**, **DepAmt**, **CashBk**, **Checks**, and **Res** as the input variables. Use the EVENT=option to model the probability of buying the variable annuity product. Specify **Res** as a CLASS variable and use reference cell coding and S as the reference cell. Use the UNITS statement to obtain an odds ratio estimate for a 1000-unit change in **DDABal** and **DepAmt**. Create an effect plot for **DDABal**, **DepAmt**, **Checks**, and **Res** with confidence bands, an effect plot of **DDABal** and **DepAmt** grouped by **DDA**, and an odds ratio plot that displays a horizontal orientation and the statistics. Use the ODDSRATIO statement to compute the odds ratios for all the pairwise comparisons in **Res**. Specify profile likelihood confidence intervals, and request standardized estimates for the parameters for the continuous input variables.

```
/* pmlr02d01.sas  */
title1 "Logistic Regression Model for the Variable Annuity Data Set";
proc logistic data=work.train plots(only maxpoints=none) =
      (effect(clband x=(ddabal depamt checks res))
      oddsratio (type=horizontalstat));
   class res (param=ref ref='S') dda (param=ref ref='0');
   model ins(event='1')=dda ddabal dep depamt cashbk checks res
      / stb clodds=pl;
   units ddabal=1000 depamt=1000 / default=1;
   oddsratio 'Comparisons of Residential Classification' res
      / diff=all cl=pl;
   effectplot slicefit(sliceby=dda x=ddabal) / noobs;
   effectplot slicefit(sliceby=dda x=depamt) / noobs;
run;
```

Selected PROC LOGISTIC statement option:

PLOTS= controls the plots produced through ODS Statistical Graphics.

Selected global plot option:

ONLY suppresses the default plots. Only specifically requested plot-requests are displayed.

MAXPOINTS=NONE
requests that no observations be displayed in the effect plots.

Selected plot requests:

EFFECT displays and enhances the effect plots for the model.

ODDSRATIO displays and enhances the odds ratio plots for the model when the CLODDS= option or ODDSRATIO statements are also specified.

Selected effect plot options:

CLBAND displays confidence limits on the plots.

X= specifies effects to be used on the X axis of the effect plots. You can specify several X axes. Continuous variables must be specified as main effects. CLASS variables can be crossed.

Selected ODDSRATIO plot option:

TYPE controls the look of the graphic. The default TYPE=HORIZONTAL option places the odds ratio values on the X axis, and the TYPE=HORIZONTALSTAT option displays the values of the odds ratios and their confidence limits on the right side of the graphic. The TYPE=VERTICAL option places the odds ratio values on the Y axis, but the TYPE=VERTICALBLOCK option (available only with the CLODDS= option) places the odds ratio values on the Y axis and puts boxes around the labels.

Selected CLASS statement option:

PARAM= specifies the parameterization method for the classification variable or variables. The default is PARAM=EFFECT.

REF=*'level' | keyword*
specifies the reference level for PARAM=EFFECT, PARAM=REF, and their orthogonalizations. For an individual variable, you can specify the level of the variable to use as the reference level. For a global or individual variable, you can use one of the following keywords:

FIRST designates the first ordered level as the reference.

LAST designates the last ordered level as the reference. This is the default.

Selected MODEL statement options:

EVENT=*'category' | keyword*
specifies the event category for the binary response model. PROC LOGISTIC models the probability of the event category. The EVENT= option has no effect when there are more than two response categories. You can specify the value (formatted if a format is applied) of the event category in quotation marks or you can specify one of the following keywords:

FIRST designates the first ordered category as the event. This is the default.

LAST designates the last ordered category as the event.

CLODDS= requests confidence intervals for the odds ratios. Computation of these confidence intervals is based on the profile likelihood (CLODDS=PL, which is desirable for small sample sizes) or based on individual Wald tests (CLODDS=WALD). If you specify CLODDS=BOTH, the procedure computes two sets of confidence intervals for the odds ratios. One is based on the profile likelihood and the other is based on the Wald tests.

STB displays the standardized estimates for the parameters in the Analysis of Maximum Likelihood Estimates table.

Selected UNITS statement option:

DEFAULT= gives a list of units of change for all predictor variables that are not specified in the UNITS statement. If the DEFAULT= option is not specified, PROC LOGISTIC does not produce customized odds ratio estimates for any predictor variable that is not listed in the UNITS statement.

Selected ODDSRATIO statement options:

CL specifies whether to create Wald or profile-likelihood confidence limits, or both. By default, Wald confidence limits are produced.

DIFF= specifies whether the odds ratios for a classification variable are computed against the reference level, or all pairs of *variable* are compared. By default, DIFF=ALL. The DIFF= option is ignored when *variable* is continuous.

Selected EFFECTPLOT statement options:

SLICEFIT displays a curve of predicted values versus a continuous variable, grouped by the levels of a CLASS effect.

NOOBS suppresses the display of observations.

Selected plot definition options:

SLICEBY= displays the fitted values at the different levels of the specified variable.

X= specifies the values to be displayed on the X axis.

```
     Logistic Regression Model for the Variable Annuity Data Set

                        The LOGISTIC Procedure

                          Model Information

          Data Set                      WORK.TRAIN
          Response Variable             Ins
          Number of Response Levels     2
          Model                         binary logit
          Optimization Technique        Fisher's scoring

              Number of Observations Read       21512
              Number of Observations Used       21512

                       Response Profile

            Ordered                        Total
              Value          Ins     Frequency

                 1            0         14061
                 2            1          7451

                Probability modeled is Ins=1.
```

The results consist of a number of tables. The Response Profile table shows the target variable values listed according to their ordered values. By default, the target-variable values are ordered alphanumerically and PROC LOGISTIC always models the probability of the first ordered value. Because you used the EVENT= option, in this example, the model is based on the probability of buying the variable annuity product (**Ins**=1).

Class Level Information

Class	Value	Design Variables	
Res	R	1	0
	S	0	0
	U	0	1
DDA	0	0	
	1	1	

The Class Level Information table shows that reference cell coding and the level S as the reference level were used to dummy-code the **Res** variable into two design variables.

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	27759.675	26284.098
SC	27767.651	26355.885
-2 Log L	27757.675	26266.098

The Model Fit Statistics table contains the Akaike information criteria (AIC) and the Schwarz criterion (SC). These are goodness-of-fit measures that you can use to compare one model to another. When the values of the goodness-of-fit statistics are smaller, the fit of the model to the data is better.

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	1491.5772	8	<.0001
Score	1315.6105	8	<.0001
Wald	1256.8282	8	<.0001

The Likelihood Ratio, Wald, and Score tests test the null hypothesis that all regression coefficients of the model other than the intercept are 0.

Type 3 Analysis of Effects

Effect	DF	Wald Chi-Square	Pr > ChiSq
DDA	1	484.0020	<.0001
DDABal	1	317.1284	<.0001
Dep	1	26.0277	<.0001
DepAmt	1	10.1271	0.0015
CashBk	1	19.8706	<.0001
Checks	1	0.0309	0.8604
Res	2	0.1229	0.9404

The Type 3 Analysis of Effects table shows which input variables are statistically significant and controlling for all of the other input variables in the model.

Analysis of Maximum Likelihood Estimates

Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq	Standardized Estimate
Intercept		1	0.1706	0.0374	20.8591	<.0001	
DDA	1	1	-1.0410	0.0473	484.0020	<.0001	-0.2226
DDABal		1	0.000075	4.188E-6	317.1284	<.0001	0.3135
Dep		1	-0.0682	0.0134	26.0277	<.0001	-0.0648
DepAmt		1	0.000012	3.819E-6	10.1271	0.0015	0.0460
CashBk		1	-0.6393	0.1434	19.8706	<.0001	-0.0468
Checks		1	-0.00068	0.00384	0.0309	0.8604	-0.00193
Res	R	1	-0.0129	0.0388	0.1106	0.7395	-0.00308
Res	U	1	-0.00191	0.0343	0.0031	0.9557	-0.00051

The parameter estimates measure the rate of change in the logit (log odds) corresponding to a one-unit change in input variable, adjusted for the effects of the other inputs. The parameter estimates are difficult to compare because they depend on the units in which the variables are measured. The standardized estimates convert them to standard deviation units. The absolute value of the standardized estimates can be used to give an approximate ranking of the relative importance of the input variables on the fitted logistic model.

Association of Predicted Probabilities and Observed Responses

Percent Concordant	67.2	Somers' D	0.357
Percent Discordant	31.5	Gamma	0.362
Percent Tied	1.3	Tau-a	0.162
Pairs	104768511	c	0.679

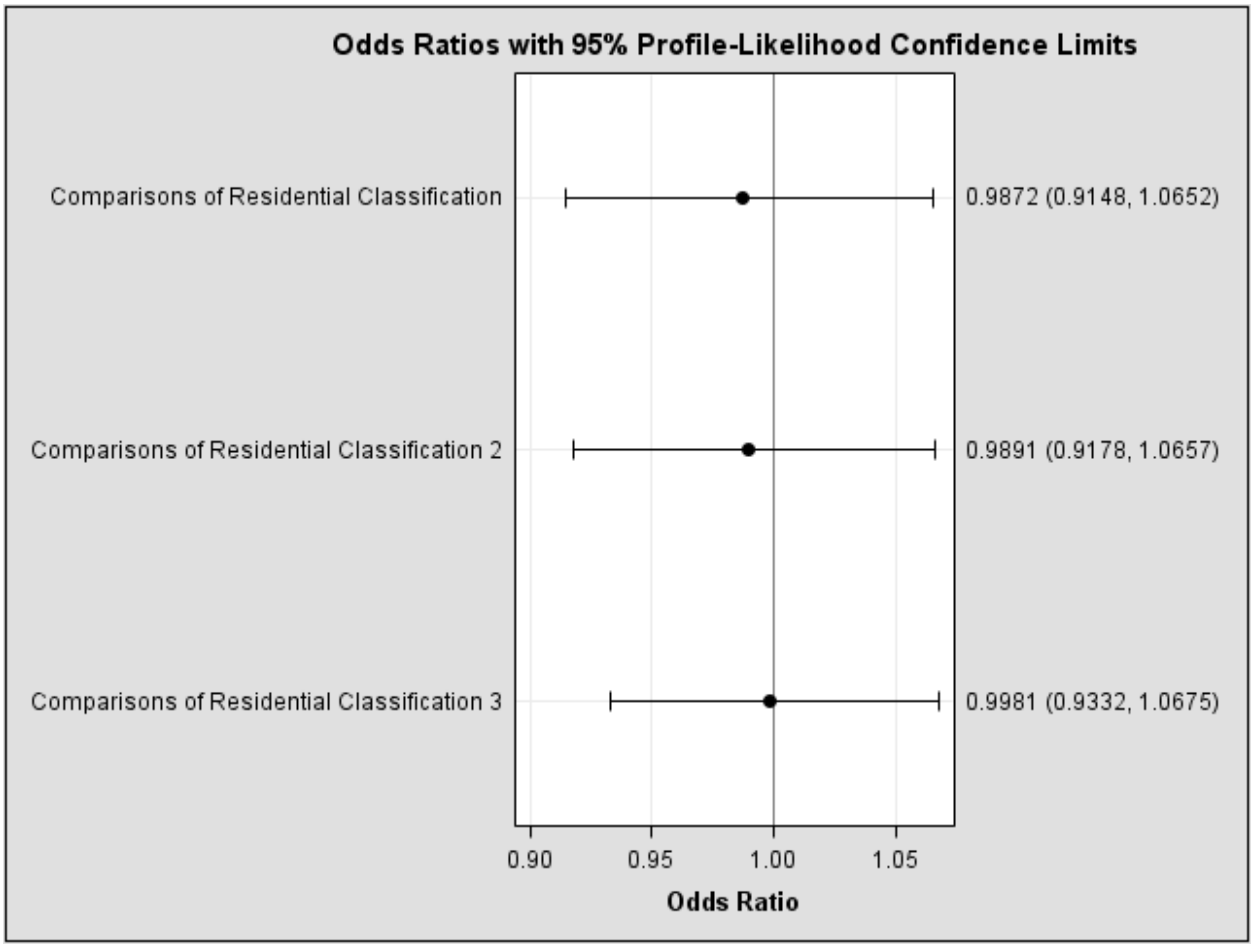
The Association of Predicted Probabilities and Observed Responses table lists several measures that assess the predictive ability of the model. Recall that for all pairs of observations with different values of the target variable, a pair is *concordant* if the observation with the outcome has a higher predicted outcome probability (based on the model) than the observation without the outcome. A pair is *discordant* if the observation with the outcome has a lower predicted outcome probability than the observation without the outcome.

The four rank correlation indexes (Somer's D, Gamma, Tau-a, and c) are computed from the numbers of concordant and discordant pairs of observations. In general, a model with higher values for these indexes (the maximum value is 1) has better predictive ability than a model with lower values for these indexes.

Odds Ratio Estimates and Profile-Likelihood Confidence Intervals

Label	Odds Ratio	Estimate	95% Confidence Limits	
Comparisons of Residential Classification	Res R vs S	0.987	0.915	1.065
Comparisons of Residential Classification 2	Res R vs U	0.989	0.918	1.066
Comparisons of Residential Classification 3	Res U vs S	0.998	0.933	1.068

The table produced from the ODDSRATIO statement shows the odds ratios that compare rural residence to suburb residence, rural residence to urban residence, and urban residence to suburb residence. The 95% confidence limits show that none of the comparisons is statistically significant at the 0.05 level.



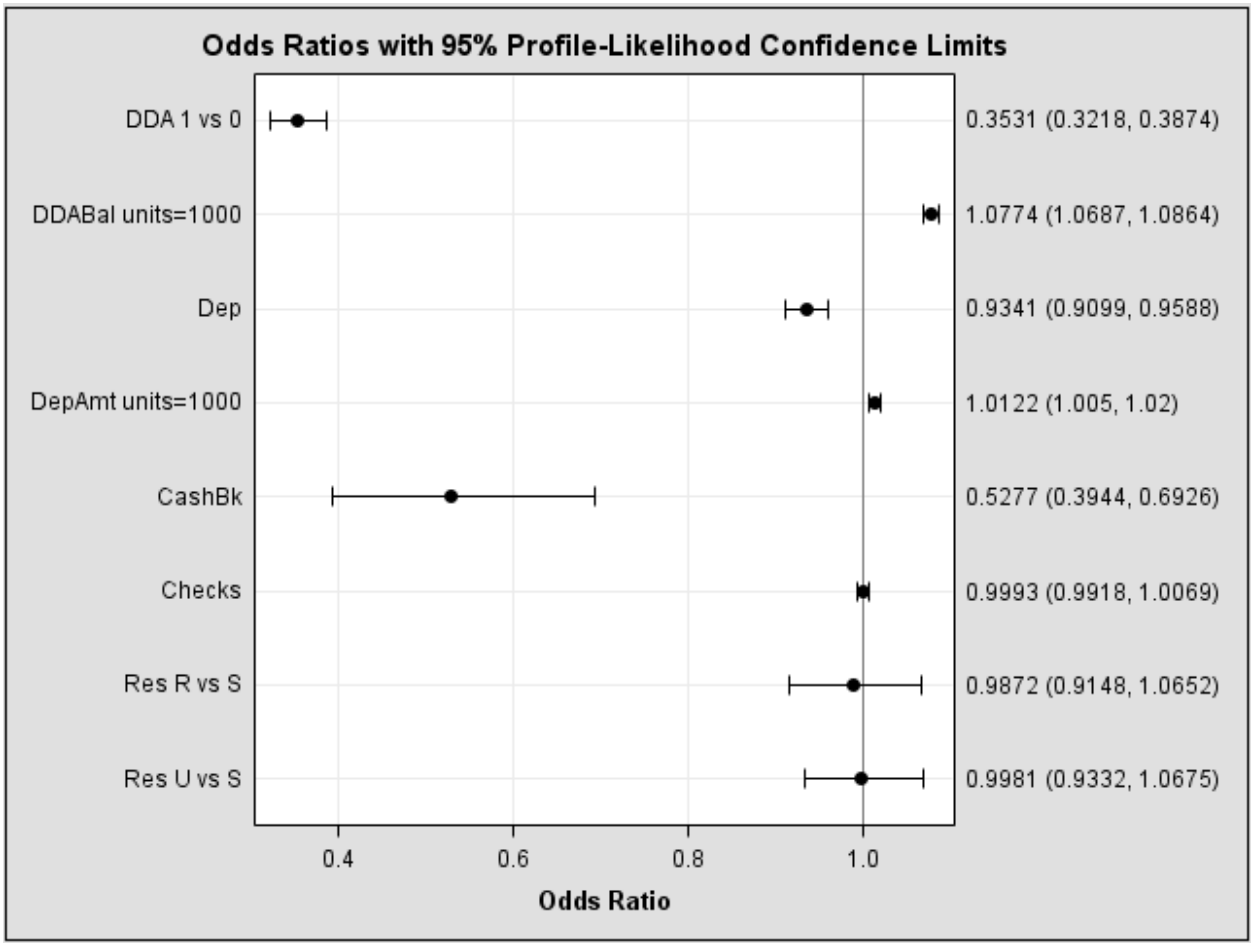
The odds ratio plot that corresponds to the ODDSRATIO statement shows the odds ratios that compare rural residence to suburb residence (0.9872), rural residence to urban residence (0.9891), and urban residence to suburb residence (0.9981).

Odds Ratio Estimates and Profile-Likelihood Confidence Intervals

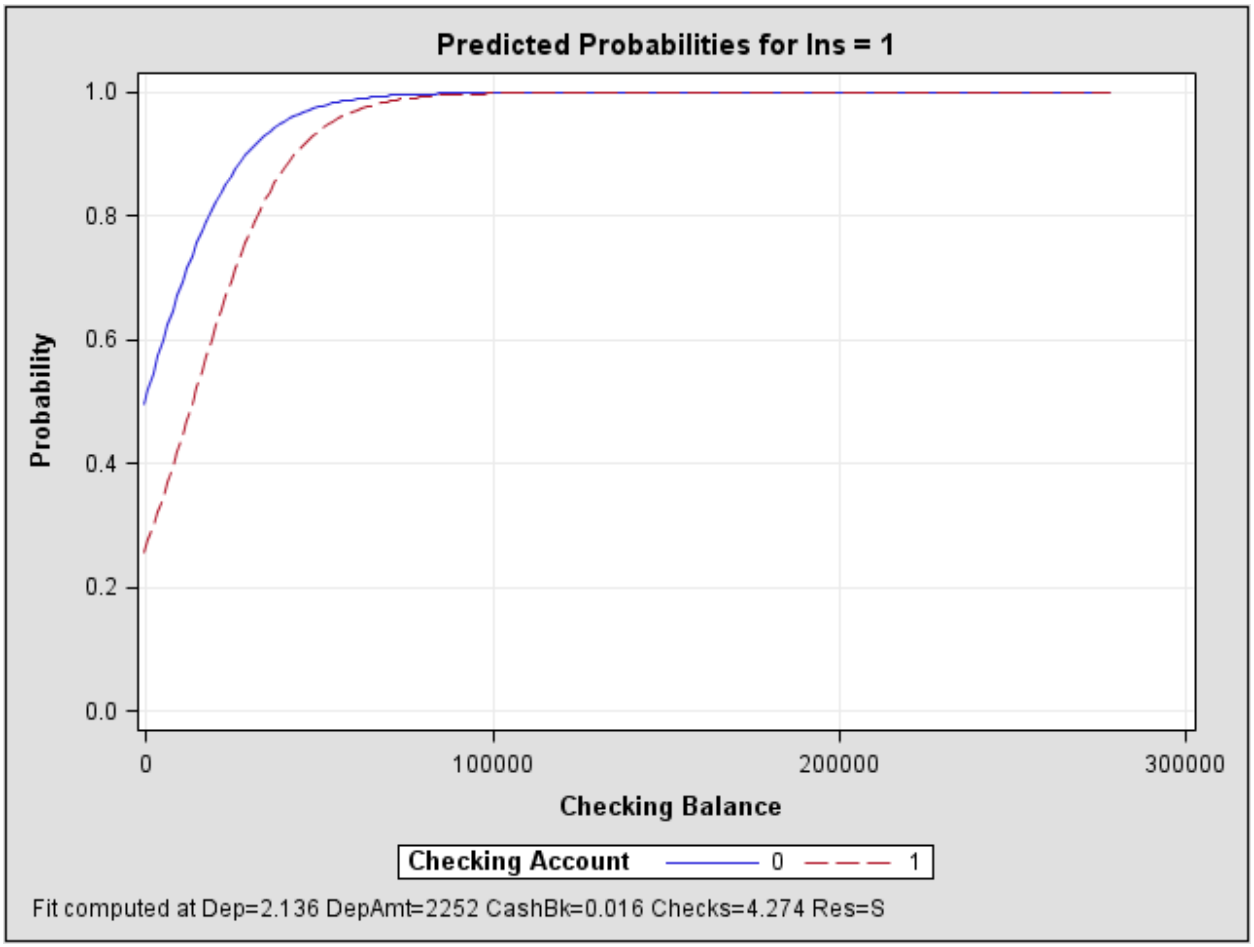
Effect		Unit	Estimate	95% Confidence Limits	
DDA	1 vs 0	1.0000	0.353	0.322	0.387
DDABal		1000.0	1.077	1.069	1.086
Dep		1.0000	0.934	0.910	0.959
DepAmt		1000.0	1.012	1.005	1.020
CashBk		1.0000	0.528	0.394	0.693
Checks		1.0000	0.999	0.992	1.007
Res	R vs S	1.0000	0.987	0.915	1.065
Res	U vs S	1.0000	0.998	0.933	1.068

The odds ratio measures the effect of the input variable on the target adjusted for the effect of the other input variables. For example, the odds of acquiring an insurance product for **DDA** (checking account) customers are .353 times the odds for non-**DDA** customers. Equivalently, the odds of acquiring an insurance product are 1/.353 or 2.83 times more likely for non-**DDA** customers compared to **DDA** customers.

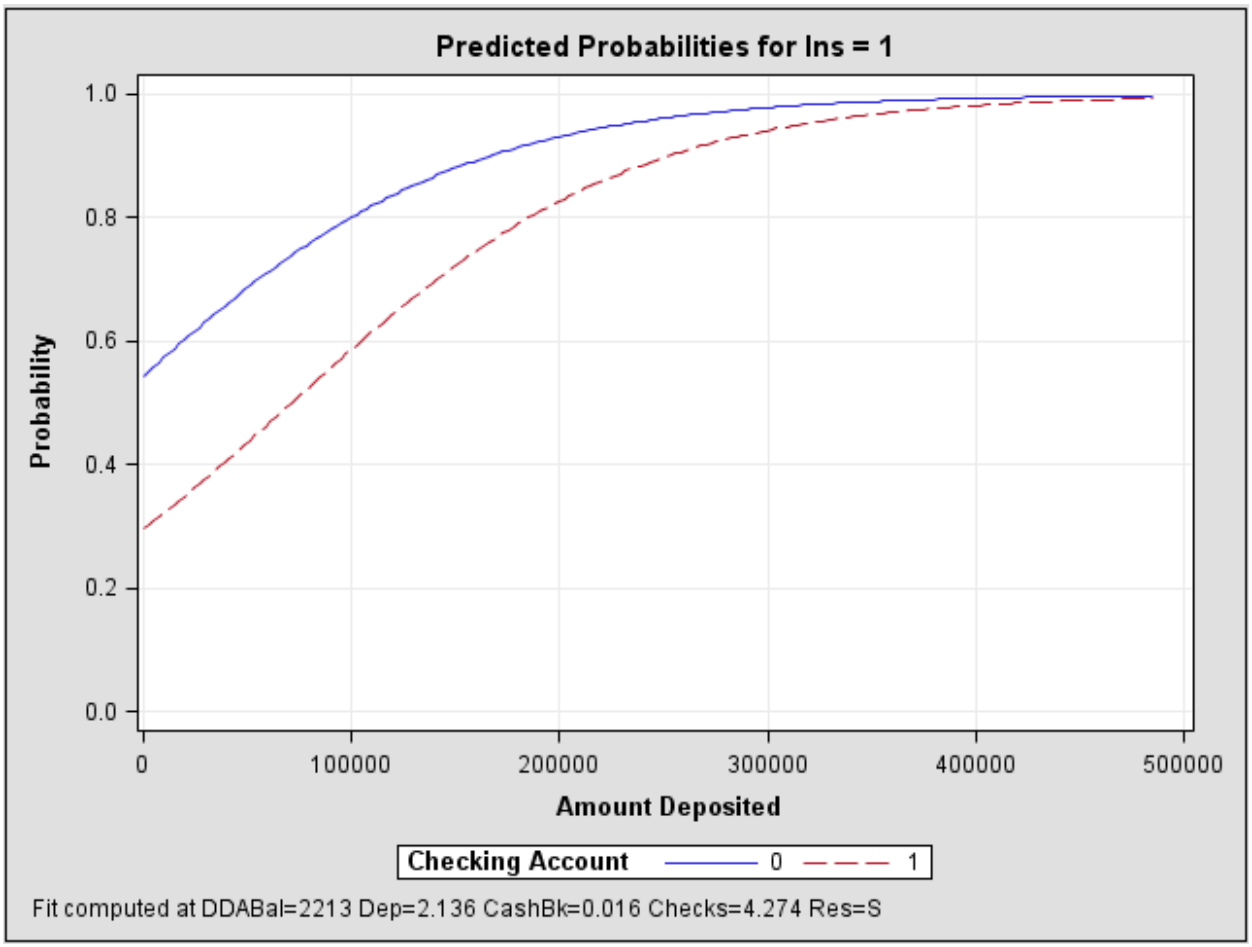
For continuous variables, it might be useful to convert the odds ratio to a percentage increase or decrease in odds. For example, the odds ratio for a 1000-unit change in **DDABal** is 1.077. Consequently, the odds of acquiring the insurance product increase 7.7% (calculated as 100(1.077–1)) for every thousand-dollar increase in the checking balance, assuming that the other variables do not change.



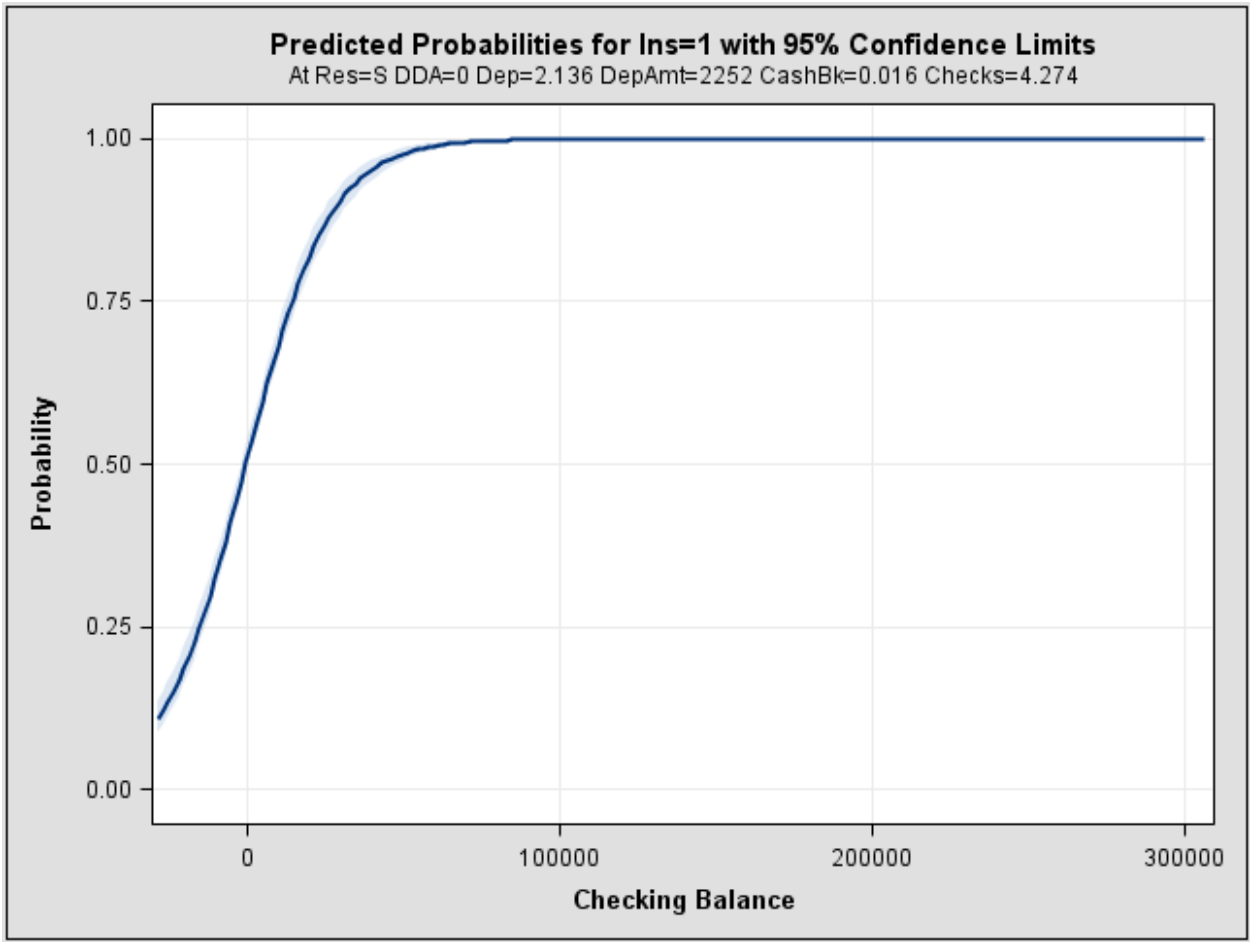
The odds ratio plot incorporates the information from the Profile Likelihood Confidence Interval for the Odds Ratio table.



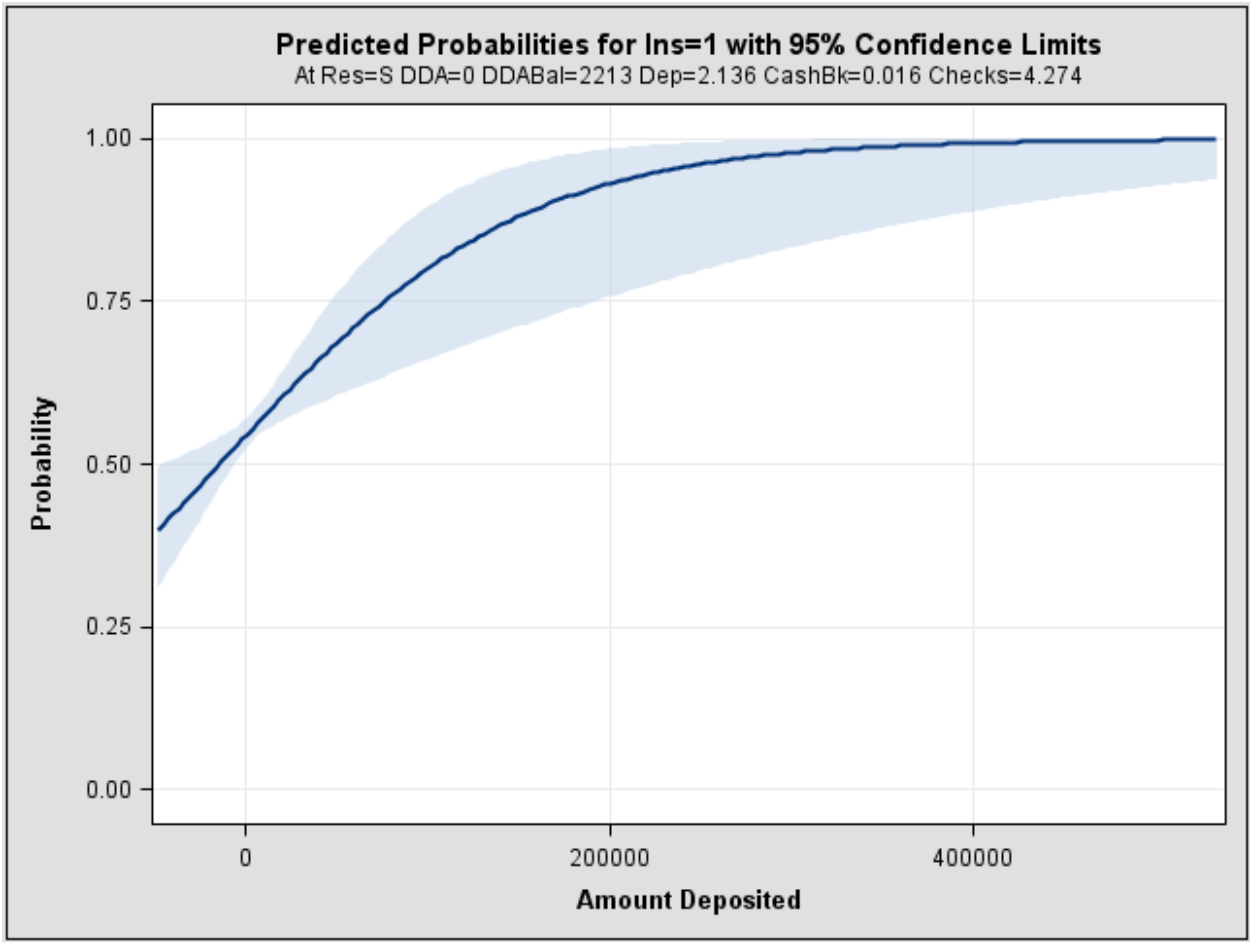
The effect plot of Checking Balance by checking account shows that the customers without checking accounts have higher probabilities of buying the variable annuity product than those customers with checking account balances less than approximately 90,000 dollars. The fitted lines are computed by setting the other continuous predictor variables at their means, and the categorical predictor variable (**Res**) at its reference level. The effect plot also shows a strong positive relationship between the probability of buying the variable annuity product and checking account balance.



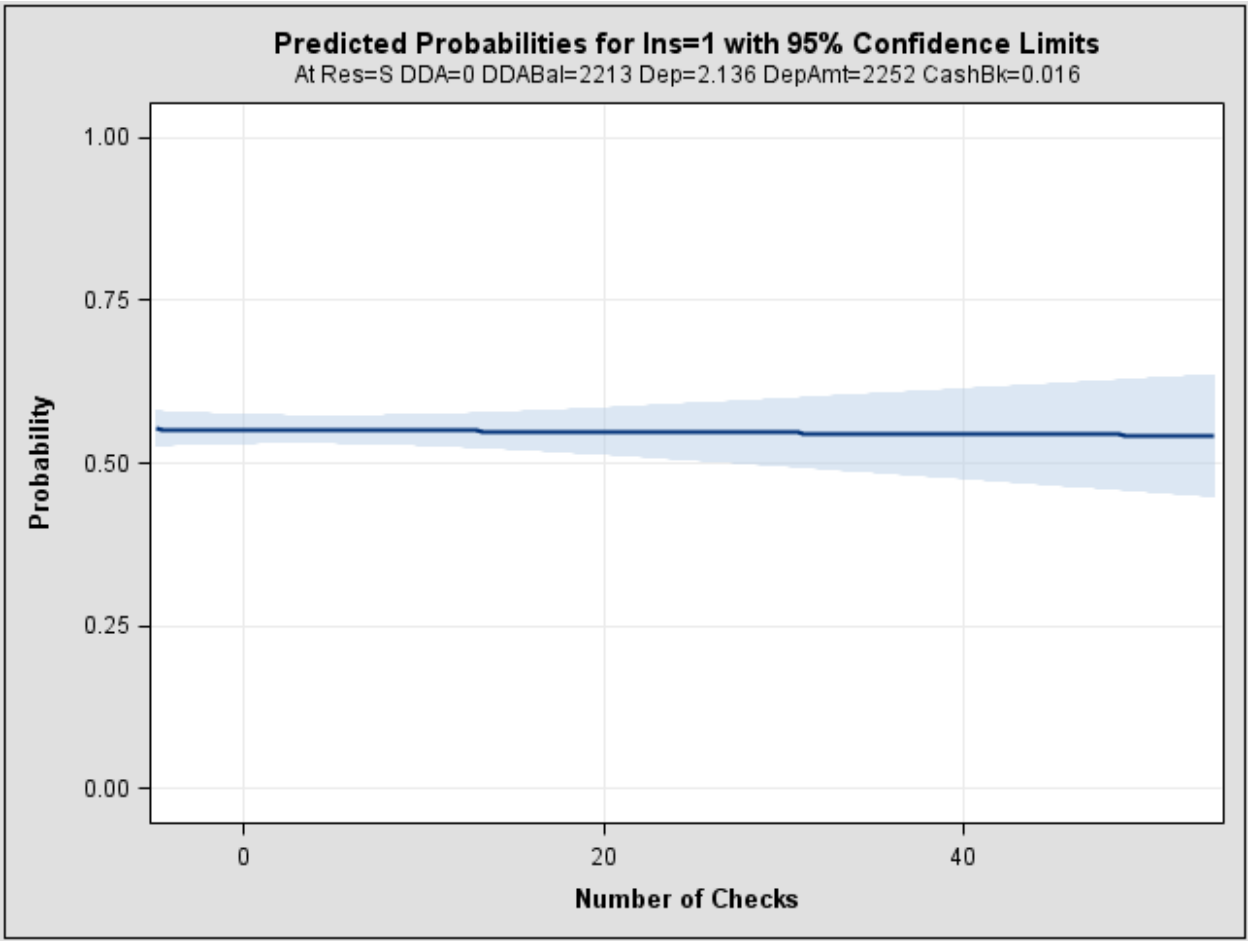
The effect plot of Amount Deposited by checking account shows that the customers without checking accounts have higher probabilities of buying the variable annuity product across the range of amounts deposited. The fitted lines are computed by setting the other continuous predictor variables at their means, and the categorical predictor variable (**Res**) at its reference level. The effect plot also shows a strong positive relationship between the probability of buying the variable annuity product and the amount deposited.



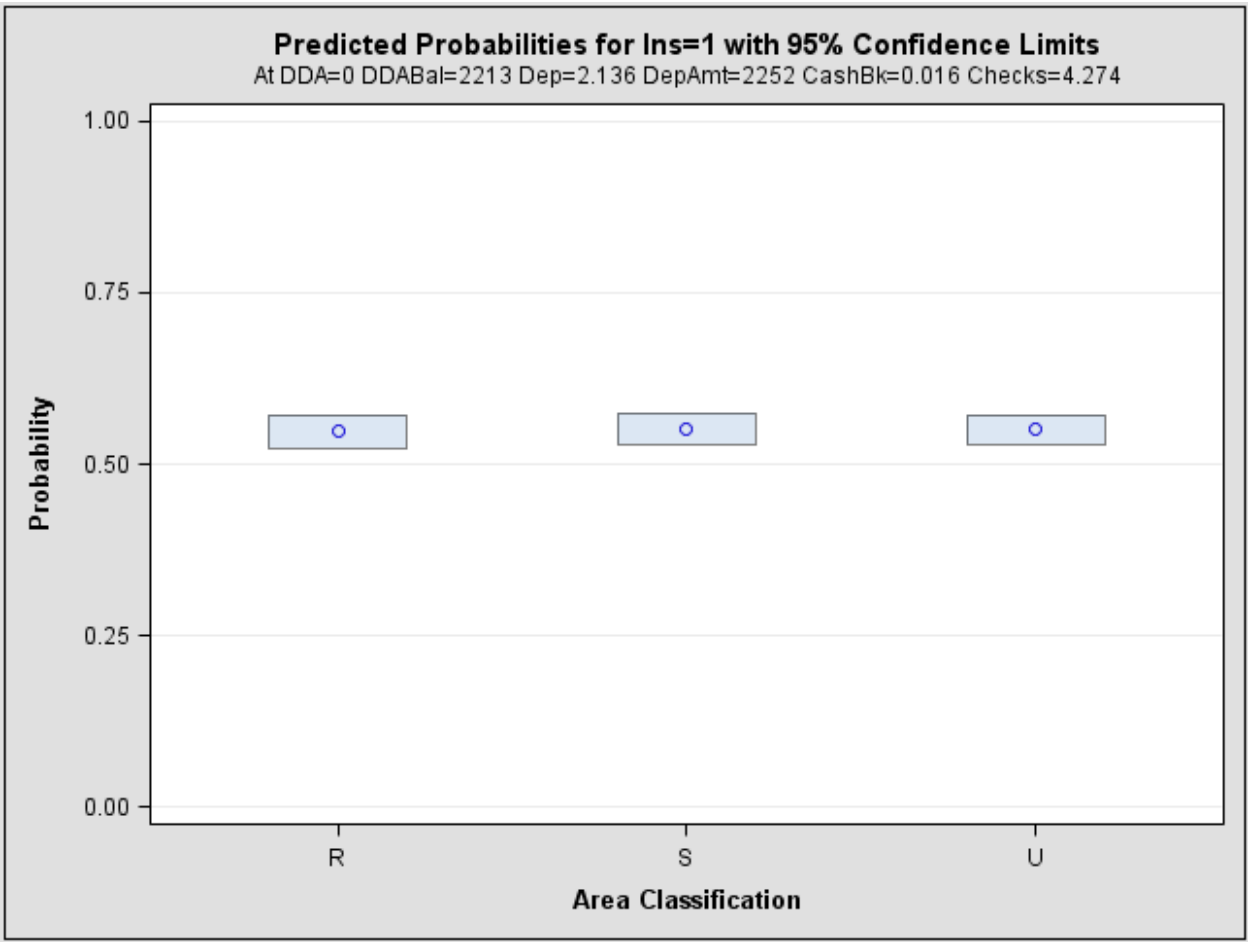
The effect plot for Checking Balance shows the predicted probabilities of the event by the values of checking account balance setting the other continuous predictor variables at their means, and the categorical predictor variable (**Res**) at its reference level. The plot shows a strong positive relationship between the probability of buying the variable annuity product and checking account balance.



The effect plot for Amount Deposited shows a strong positive relationship between the probability of buying the variable annuity product and the amount deposited. Confidence bands around the regression line show that the precision of the predicted probabilities decreases as you move farther away from the mean probability and the mean of amount deposited.



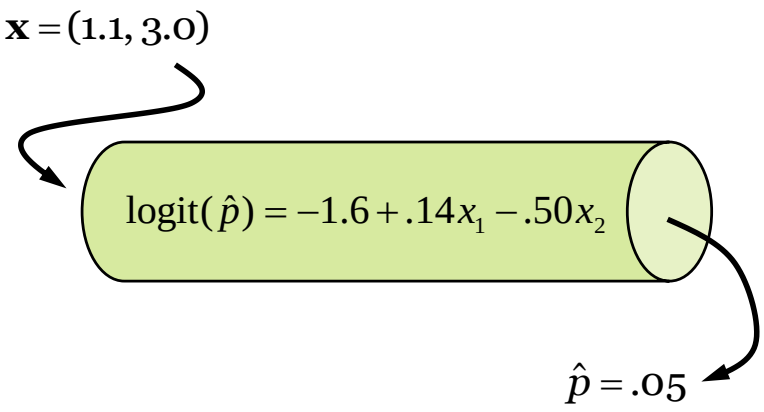
The effect plot for Number of Checks shows a weak negative relationship between the probability of buying the variable annuity product and number of checks.



The effect plot for Area Classification shows no relationship between the probability of buying the variable annuity product and the residential area of the customer.

End of Demonstration

Scoring New Cases



17

The overriding purpose of predictive modeling is to score new cases. Predictions can be made by simply plugging in the new values of the inputs.

Scoring New Cases

Example: Use the SCORE statement in PROC LOGISTIC to score the **pmlr.new** data set.

```
/* pmlr02d02.sas */
proc logistic data=work.train noprint;
   class res (param=ref ref='S');
   model ins(event='1')= res dda ddabal dep depamt cashbk checks;
   score data = pmlr.new out=work.scored1;
run;
```

Selected SCORE statement options:

DATA= names the data set you want to score. It is not necessary for the data set to contain the target variable unless you are specifying the FITSTAT or OUTROC= option.

OUT= names the data set that contains all the information in the DATA= data set together with the posterior probabilities and, as an option, prediction confidence intervals.

```
title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored1(obs=10);
   var p_1 dda ddabal dep depamt cashbk checks res;
run;
```

Predicted Probabilities from Scored Data Set

Obs	P_1	DDA	DDABal	Dep	DepAmt	Cash Bk	Checks	Res
1	0.27023	1	56.29	2	955.51	0	1	U
2	0.32036	1	3292.17	2	961.60	0	1	U
3	0.29822	1	1723.86	2	2108.65	0	2	U
4	0.54208	0	0.00	0	0.00	0	0	U
5	0.26946	1	67.91	2	519.24	0	3	S
6	0.32228	1	2554.58	1	501.36	0	2	S
7	0.27038	1	0.00	2	2883.08	0	12	R
8	0.30399	1	2641.33	3	4521.61	0	8	S
9	0.54255	0	0.00	0	0.00	0	0	S
10	0.27956	1	52.22	1	75.59	0	0	R

The predicted probability that **Ins** is 1 is named **P_1**.

```
title1 "Mean of Predicted Probabilities from Scored Data Set";
proc means data=work.scored1 mean nolabels;
   var p_1;
run;
```

Mean of Predicted Probabilities from Scored Data Set

Analysis Variable : P_1

Mean
0.3448203

The mean of the predicted probabilities is approximately the same as the mean of the target variable, **ins** (0.3464). This is not a coincidence.

Example: Use the OUTMODEL= and INMODEL= options in PROC LOGISTIC to score the **pmlr.new** data set.

```
proc logistic data=work.train outmodel=work.scoredata noprint;
   class res (param=ref ref='S');
   model ins(event='1')= res dda ddabal dep depamt cashbk checks;
run;
```

Selected PROC LOGISTIC statement option:

OUTMODEL= specifies the name of the SAS data set that contains the information about the fitted model. This data set contains sufficient information to score new data without needing to refit the model. It is solely used as the input to the INMODEL= option in a subsequent PROC LOGISTIC call.

```
proc logistic inmodel=work.scoredata noprint;
   score data = pmlr.new out=work.scored2;
run;
```

Selected PROC LOGISTIC statement option:

INMODEL= specifies the name of the SAS data set that contains the model information that is needed for scoring new data. This INMODEL= data set is the OUTMODEL= data set that was saved in a previous PROC LOGISTIC call. The OUTMODEL= data set should not be modified before its use as an INMODEL= data set.

```
title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored2(obs=10);
   var p_1 dda ddabal dep depamt cashbk checks res;
run;
```

Predicted Probabilities from Scored Data Set

Obs	P_1	DDA	DDABal	Dep	DepAmt	Cash Bk	Checks	Res
1	0.27023	1	56.29	2	955.51	0	1	U
2	0.32036	1	3292.17	2	961.60	0	1	U
3	0.29822	1	1723.86	2	2108.65	0	2	U
4	0.54208	0	0.00	0	0.00	0	0	U
5	0.26946	1	67.91	2	519.24	0	3	S
6	0.32228	1	2554.58	1	501.36	0	2	S
7	0.27038	1	0.00	2	2883.08	0	12	R
8	0.30399	1	2641.33	3	4521.61	0	8	S
9	0.54255	0	0.00	0	0.00	0	0	S
10	0.27956	1	52.22	1	75.59	0	0	R

The predicted probabilities are the same as the previous demonstration.

Example: Use the CODE statement and a DATA step to score the **pmlr.new** data set.

```
proc logistic data=work.train noprint;
   class res (param=ref ref='S');
   model ins(event='1')= res dda ddabal dep depamt cashbk checks;
   code file="&PMLRfolder\pmlr_score.txt";
run;
```

Selected PROC LOGISTIC statement:

CODE writes SAS DATA step code for computing predicted values of the fitted model either to a file or to a catalog entry. This code can then be included in a DATA step to score new data.

Selected CODE statement option:

FILE= names the file where the generated code is saved.

```
data work.scored3;
   set pmlr.new;
   %include "&PMLRfolder\pmlr_score.txt";
run;

title1 "Predicted Probabilities from Scored Data Set";
proc print data=work.scored3(obs=10);
   var p_ins1 dda ddabal dep depamt cashbk checks res;
run;
```

Predicted Probabilities from Scored Data Set

Obs	P_Ins1	DDA	DDABal	Dep	DepAmt	Cash Bk	Checks	Res
1	0.27023	1	56.29	2	955.51	0	1	U
2	0.32036	1	3292.17	2	961.60	0	1	U
3	0.29822	1	1723.86	2	2108.65	0	2	U
4	0.54208	0	0.00	0	0.00	0	0	U
5	0.26946	1	67.91	2	519.24	0	3	S
6	0.32228	1	2554.58	1	501.36	0	2	S
7	0.27038	1	0.00	2	2883.08	0	12	R
8	0.30399	1	2641.33	3	4521.61	0	8	S
9	0.54255	0	0.00	0	0.00	0	0	S
10	0.27956	1	52.22	1	75.59	0	0	R

The predicted probabilities are the same as the previous two demonstrations.

When an item store is created in PROC LOGISTIC, PROC PLM can also be used to score new cases without rerunning the model. It can also generate scoring code for use in a DATA step.

End of Demonstration

2.2 Adjustments for Oversampling

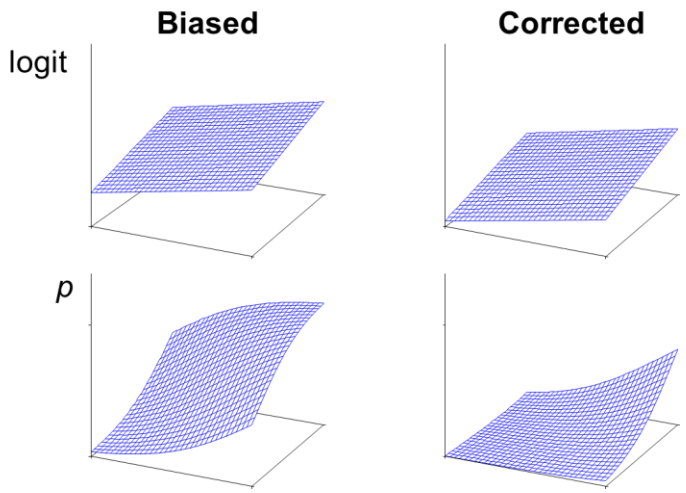
Objectives

- Explain the effect of oversampling.
- Illustrate the offset method.
- Adjust the model for oversampling in PROC LOGISTIC.

20

sas

The Effect of Oversampling



21

sas

The maximum likelihood estimates were derived under the assumption that y_i values have independent Bernoulli distributions. This assumption is appropriate for joint sampling but not for separate sampling. However, the effects of violating this assumption can be easily corrected. In logistic regression, only the estimate of the intercept, β_0, is affected by using Bernoulli ML on data from a separate sampling design (Prentice and Pike 1979). If the standard model:

$$\text{logit}(p_i) = \beta_o + \beta_1 x_1 + \cdots + \beta_k x_k$$

is appropriate for joint sampling, then ML estimates of the parameters under separate sampling can be determined by fitting the *pseudo model* (Scott and Wild 1986, 1997):

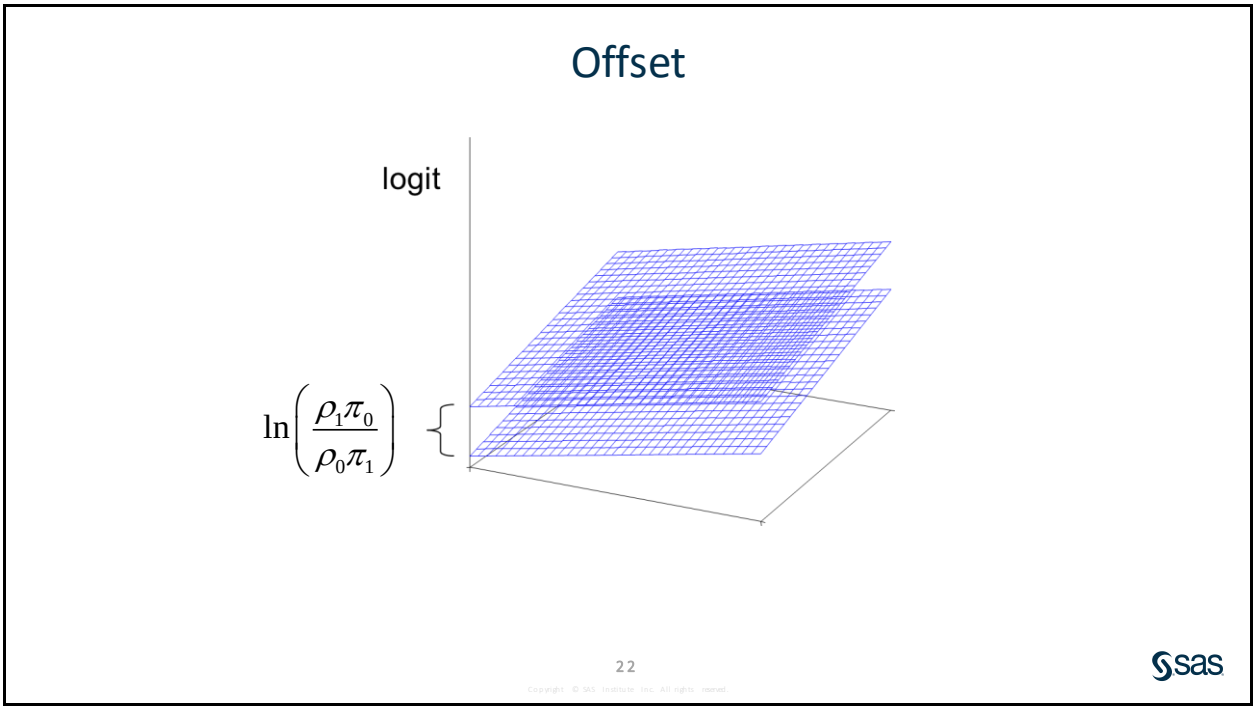
$$\text{logit}(p_i^*) = \ln\left(\frac{\rho_1 \pi_0}{\rho_0 \pi_1}\right) + \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k$$

where p^* is the posterior probability corresponding to the biased sample. Consequently, the effect of oversampling is to shift the logits by a constant amount – the *offset*:

$$\ln\left(\frac{\rho_1 \pi_o}{\rho_o \pi_1}\right)$$

The *priors*, π_0 and π_1, represent the population proportions of class 0 and 1, respectively. The proportions of the target classes in the sample are denoted ρ_0 and ρ_1. In separate sampling (nonproportional) $\pi_0 \neq \rho_0$ and $\pi_1 \neq \rho_1$, the adjustments for oversampling require the priors be known *a priori*. Notice that if $\pi_1 = \rho_1$, then $\pi_0 = \rho_0$ and the offset is 0, as you would expect, because the sample proportions and population proportions of responders and non-responders are the same.

When rare events are oversampled, $\pi_0 > \rho_0$ and $\pi_1 < \rho_1$, the offset is positive. (The logit is too large.) This vertical shift of the logit affects the posterior probability in a corresponding fashion.



The pseudo model can be fitted directly by incorporating the offset into the model. Alternatively, the offset could be applied *after* the standard model is fitted. Subtracting the offset from the predicted values and solving for the posterior probability is solved by the following:

$$\hat{p}_i = \frac{\hat{p}_i^* \rho_0 \pi_1}{(1-\hat{p}_i^*)\,\rho_1 \pi_0 + \hat{p}_i^* \rho_0 \pi_1}$$

where $\hat{p}_i^*$ is the unadjusted estimate of the posterior probability. Both approaches give identical results. For both types of adjustments, the population priors, π_0 and π_1, need to be known *a priori*, but the sample priors, ρ_0 and ρ_1, can be estimated from the data.

Because only the intercept is affected, the adjustments might not be necessary. If the goal of the analysis is to understand the relationships between the inputs and the target, or to rank order the population, then the adjustment is not critical. If the predicted probabilities are important, and not just necessary for rank ordering or classification, then the correction for oversampling is necessary.

Correcting for Oversampling

Example: Separate sampling was used to create the data set for analysis. The proportion of the target event in the population was .02, not .346 as appears in the sample. First, use the %LET statement to define the macro variable **pi1** for the population prior for class 1 (**Ins**=1). Then, use the SCORE statement and the PRIOREVENT= option to correct the predicted probabilities back to the population scale.

```
/* pmlr02d03.sas */
%global pi1;
%let pi1=.02;          /* supply the prior for class 1 */

proc logistic data=work.train noprint;
   class res (param=ref ref='S');
   model ins(event='1')=dda ddabal dep depamt cashbk checks res;
   score data = pmlr.new out=work.scored4 priorevent=&pi1;
run;
```

Selected SCORE statement option:

PRIOREVENT= specifies the prior event probability for a binary response model.

```
title1 "Adjusted Predicted Probabilities from Scored Data Set";
proc print data=scored4(obs=10);
   var p_1 dda ddabal dep depamt cashbk checks res;
run;
```

Adjusted Predicted Probabilities from Scored Data Set

Obs	P_1	DDA	DDABal	Dep	DepAmt	Cash Bk	Checks	Res
1	0.014060	1	56.29	2	955.51	0	1	U
2	0.017830	1	3292.17	2	961.60	0	1	U
3	0.016102	1	1723.86	2	2108.65	0	2	U
4	0.043602	0	0.00	0	0.00	0	0	U
5	0.014007	1	67.91	2	519.24	0	3	S
6	0.017985	1	2554.58	1	501.36	0	2	S
7	0.014071	1	0.00	2	2883.08	0	12	R
8	0.016543	1	2641.33	3	4521.61	0	8	S
9	0.043682	0	0.00	0	0.00	0	0	S
10	0.014724	1	52.22	1	75.59	0	0	R

```
title1 "Mean of Adjusted Predicted Probabilities from Scored Data "
       " Set";
proc means data=scored4 mean nolabels;
   var p_1;
run;
```

```
Mean of Adjusted Predicted Probabilities from Scored Data Set

Analysis Variable : P_1

        Mean
_____________
   0.0249733
_____________
```

The mean of the adjusted predicted probabilities is approximately equal to pi1.

Example: Use the CODE statement and correct the predicted probabilities back to the population scale in the DATA Step.

```
proc logistic data=work.train noprint;
   class res (param=ref ref='S');
   model ins(event='1')=dda ddabal dep depamt cashbk checks res;
   code file="&PMLRfolder\pmlr_score_adj.txt";
run;

%global rho1;
proc SQL noprint;
  select mean(INS) into :rho1 from work.train;
quit;

data new;
   set pmlr.new;
   off=log(((1-&pi1)*&rho1)/(&pi1*(1-&rho1)));
run;

data work.scored5;
   set work.new;
   %include "&PMLRfolder\pmlr_score_adj.txt";
   eta=log(p ins1/p ins0) - off;
   prob=1/(1+exp(-eta));
run;
```

To correct the bias caused by the separate sampling, the **offset** variable needs to be created on the data set to be scored. The offset value is then subtracted from the logits, and the adjusted probabilities are computed in a DATA step.

```
title1 "Adjusted Predicted Probabilities from Scored Data Set";
proc print data=scored5(obs=10);
   var prob dda ddabal dep depamt cashbk checks res;
run;
```

Adjusted Predicted Probabilities from Scored Data Set

Obs	prob	DDA	DDABal	Dep	DepAmt	Cash Bk	Checks	Res
1	0.014060	1	56.29	2	955.51	0	1	U
2	0.017830	1	3292.17	2	961.60	0	1	U
3	0.016102	1	1723.86	2	2108.65	0	2	U
4	0.043602	0	0.00	0	0.00	0	0	U
5	0.014007	1	67.91	2	519.24	0	3	S
6	0.017985	1	2554.58	1	501.36	0	2	S
7	0.014071	1	0.00	2	2883.08	0	12	R
8	0.016543	1	2641.33	3	4521.61	0	8	S
9	0.043682	0	0.00	0	0.00	0	0	S
10	0.014724	1	52.22	1	75.59	0	0	R

The adjusted probabilities are the same as the previous demonstration.

End of Demonstration

2.02 Multiple Choice Poll

If the value for the offset is 3.2567, then the model corrected for oversampling has which of the following?

a. An intercept that is 3.2567 lower in value compared to the model fitted to the biased sample

b. Probabilities that are 3.2567% higher than the probabilities from the model fitted to the biased sample

c. Probabilities that are 3.2567% lower than the probabilities from the model fitted to the biased sample

d. Parameter estimates that are 3.2567% lower than the parameter estimates from the model fitted to the biased sample

Exercises

1. Fitting a Logistic Regression Model

a. Submit the program **pmlr01s01.sas** or use the **PMLR_UpToDate** macro. Create a global macro variable to store π_1, the proportion of responders in the population. This value is 0.05.

b. Fit a logistic regression model. Use the **pva_train** data set with **TARGET_B** as the target variable and **PEP_STAR**, **RECENT_AVG_GIFT_AMT**, and **FREQUENCY_STATUS_97NK** as the input variables. Model the probability that the target variable equals 1 and request 95% profile likelihood confidence intervals for the odds ratio.

c. Use the SCORE statement to append the predicted probability to the data, and correct for oversampling.

d. Create the following plots:

- effect plots with confidence bands for the three input variables
- effect plots of **RECENT_AVG_GIFT_AMT** and **FREQUENCY_STATUS_97NK** by **PEP_STAR**
- an odds ratio plot with a horizontal orientation and displayed statistics

e. Print the first 10 adjusted probabilities.

- Interpret the c statistic.
- Interpret the odds ratio for **PEP_STAR**.
- Interpret the effect plot of **RECENT_AVG_GIFT_AMT** by **PEP_STAR**.
- Interpret the effect plot for **RECENT_AVG_GIFT_AMT**.

End of Exercises

2.03 Multiple Choice Poll

The c statistic was 0.604. This can be interpreted as the probability of which of the following?

a. A customer donating to the national veterans' organization

b. A customer not donating to the national veterans' organization

c. If one person is randomly selected from the group of donors and another person is randomly selected from the non-donors, 60.4% of the time, the donor has the higher predicted probability of the target event as given by this model.

2.3 Chapter Summary

The standard logistic regression model assumes that the logit of the posterior probability is a linear combination of the input variables. The logit transformation is used to constrain the posterior probability to be between zero and one. The parameter estimates use the method of maximum likelihood for estimation. This method finds the parameter estimates that are most likely, given the data. When you exponentiate the parameter estimates, you obtain the odds ratio, which compares the odds of the event in one group to the odds of the event in another group.

When you oversample rare events, you can use the SCORE statement and the PRIOREVENT= option to adjust the model so that the posterior probabilities reflect the population.

General form of the LOGISTIC procedure:

```
PROC LOGISTIC <options>;
      CLASS variable</v-options>;
      MODEL response=<effects></options>;
      ODDSRATIO <'label'> variable </ options>;
      ROC <'label'> <specification> </ options>;
      ROCCONTRAST <'label'><contrast>
                        </ options>;
      SCORE <options>;
      UNITS predictor1=list1 </option>;
      OUTPUT <OUT=SAS-data-set> keyword=name...
                keyword=name></option>;
RUN;
```

2.4 Solutions

Solutions to Exercises

1. Fitting a Logistic Regression Model

a. Submit the program pmlr01s01.sas or use the PMLR_UpToDate macro. Create a global macro variable to store π_1, the proportion of responders in the population. This value is 0.05.

b. Fit a logistic regression model. Use the **pva_train** data set with **TARGET_B** as the target variable and **PEP_STAR**, **RECENT_AVG_GIFT_AMT**, and **FREQUENCY_STATUS_97NK** as the input variables. Model the probability that the target variable equals 1 and request 95% profile likelihood confidence intervals for the odds ratio.

c. Use the SCORE statement to append the predicted probability to the data, and correct for oversampling.

d. Create the following plots:

- effect plots with confidence bands for the three input variables
- effect plots of **RECENT_AVG_GIFT_AMT** and **FREQUENCY_STATUS_97NK** by **PEP_STAR**
- an odds ratio plot with a horizontal orientation and displayed statistics

e. Print the first 10 adjusted probabilities.

An electronic copy of the solution program is in **pmlr02s01.sas**.

```
%global ex_pi1;
%let ex_pi1=0.05;

title1 "Logistic Regression Model of the Veterans' Organization "
       "Data";
proc logistic data=pmlr.pva_train plots(only)=(effect(clband
     x=(pep star recent avg gift amt frequency_status_97nk)) oddsratio
          (type=horizontalstat));
   class pep_star (param=ref ref='0');
   model target b(event='1') = pep star recent_avg_gift_amt
            frequency_status_97nk / clodds=pl;
   effectplot slicefit(sliceby=pep star x=recent avg gift amt)/ noobs;
   effectplot slicefit(sliceby=pep_star x=frequency_status_97nk)
      / noobs;
   score data=pmlr.pva_train out=scopva_train priorevent=&ex_pi1;
run;
```

Logistic Regression Model of the Veterans' Organization Data

The LOGISTIC Procedure

Model Information

Data Set	PMLR.PVA_TRAIN
Response Variable	TARGET_B
Number of Response Levels	2
Model	binary logit
Optimization Technique	Fisher's scoring

Number of Observations Read	9687
Number of Observations Used	9687

Response Profile

Ordered Value	TARGET_B	Total Frequency
1	0	7265
2	1	2422

Probability modeled is TARGET_B=1.

Class Level Information

Class	Value	Design Variables
PEP_STAR	0	0
	1	1

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	10897.230	10663.061
SC	10904.409	10691.776
-2 Log L	10895.230	10655.061

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	240.1690	3	<.0001
Score	242.9486	3	<.0001
Wald	237.2875	3	<.0001

Type 3 Analysis of Effects

Effect	DF	Wald Chi-Square	Pr > ChiSq
PEP_STAR	1	43.4902	<.0001
RECENT_AVG_GIFT_AMT	1	3.9559	0.0467
FREQUENCY_STATUS_97N	1	83.8209	<.0001

Analysis of Maximum Likelihood Estimates

Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept		1	-1.6454	0.0831	392.4480	<.0001
PEP_STAR	1	1	0.3371	0.0511	43.4902	<.0001
RECENT_AVG_GIFT_AMT		1	-0.00579	0.00291	3.9559	0.0467
FREQUENCY_STATUS_97N		1	0.2179	0.0238	83.8209	<.0001

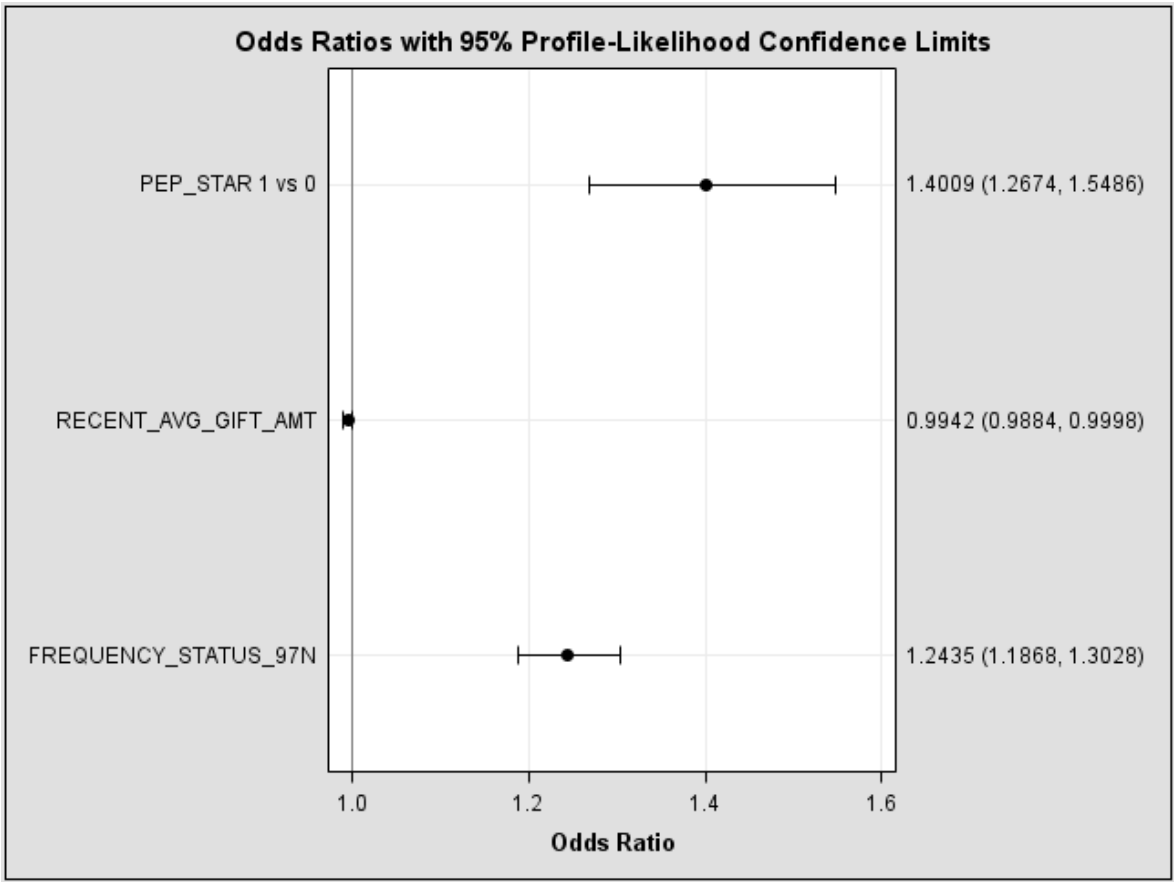
Association of Predicted Probabilities and Observed Responses

Percent Concordant	59.9	Somers' D	0.208
Percent Discordant	39.0	Gamma	0.211
Percent Tied	1.1	Tau-a	0.078
Pairs	17595830	c	0.604

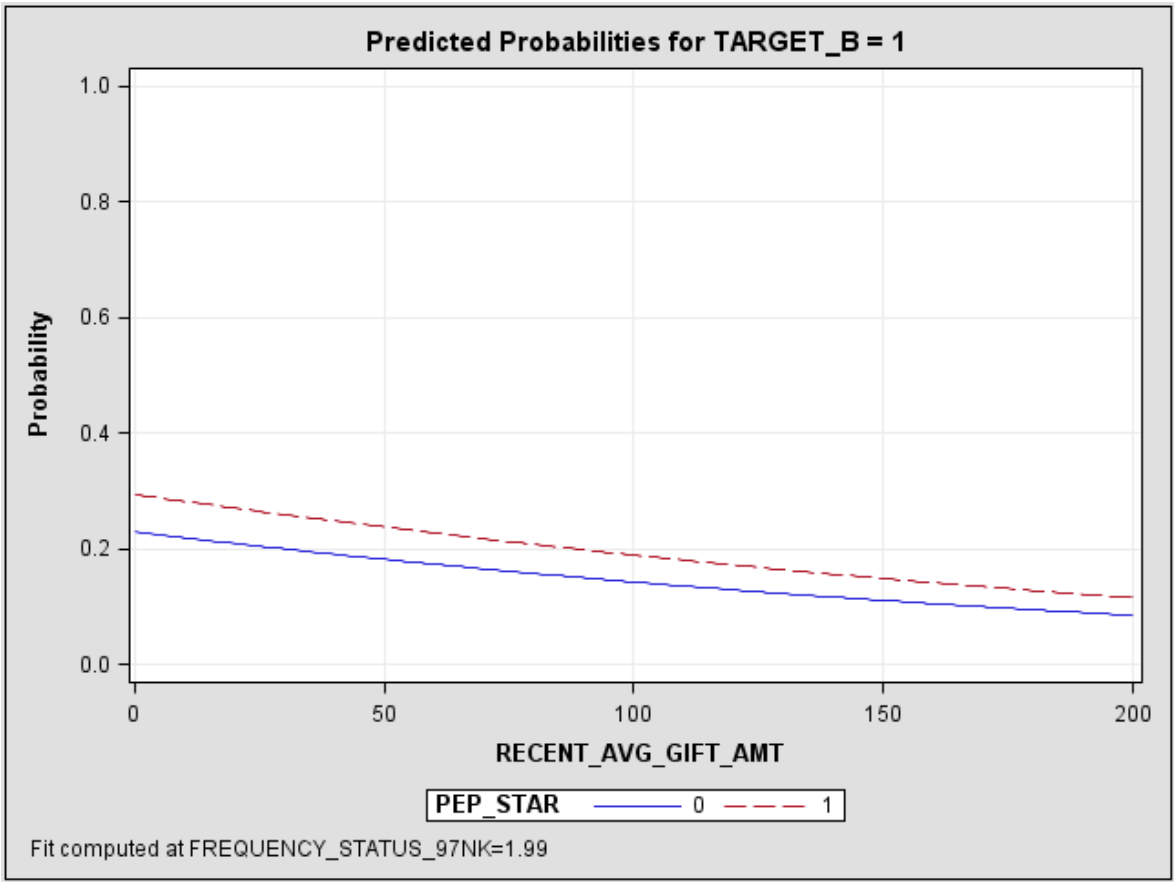
- Interpret the c statistic.
  The c statistic of 0.604 can be interpreted as the probability of a customer who donated to the national veterans' organization having a higher predicted probability (of donating to the organization) compared to a customer who did not donate.

Odds Ratio Estimates and Profile-Likelihood Confidence Intervals

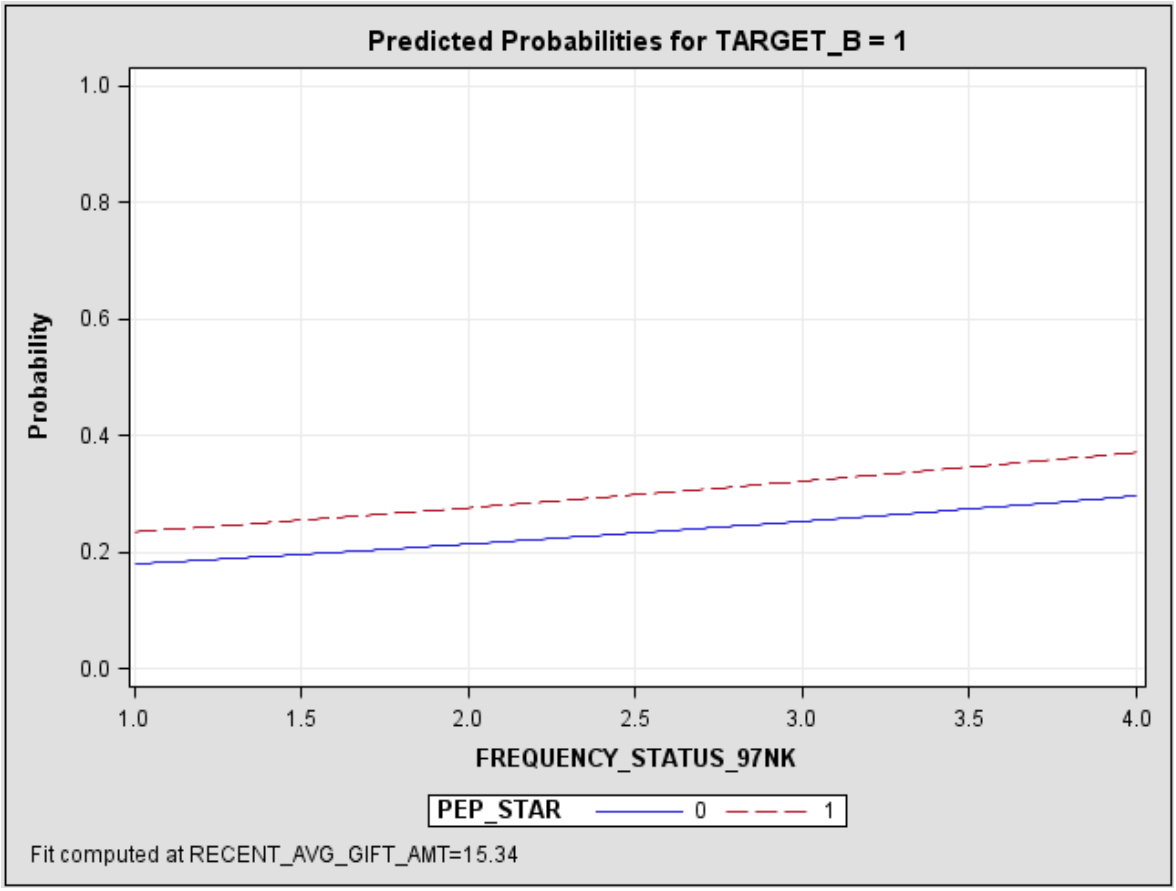
Effect	Unit	Estimate	95% Confidence Limits	
PEP_STAR 1 vs 0	1.0000	1.401	1.267	1.549
RECENT_AVG_GIFT_AMT	1.0000	0.994	0.988	1.000
FREQUENCY_STATUS_97N	1.0000	1.243	1.187	1.303

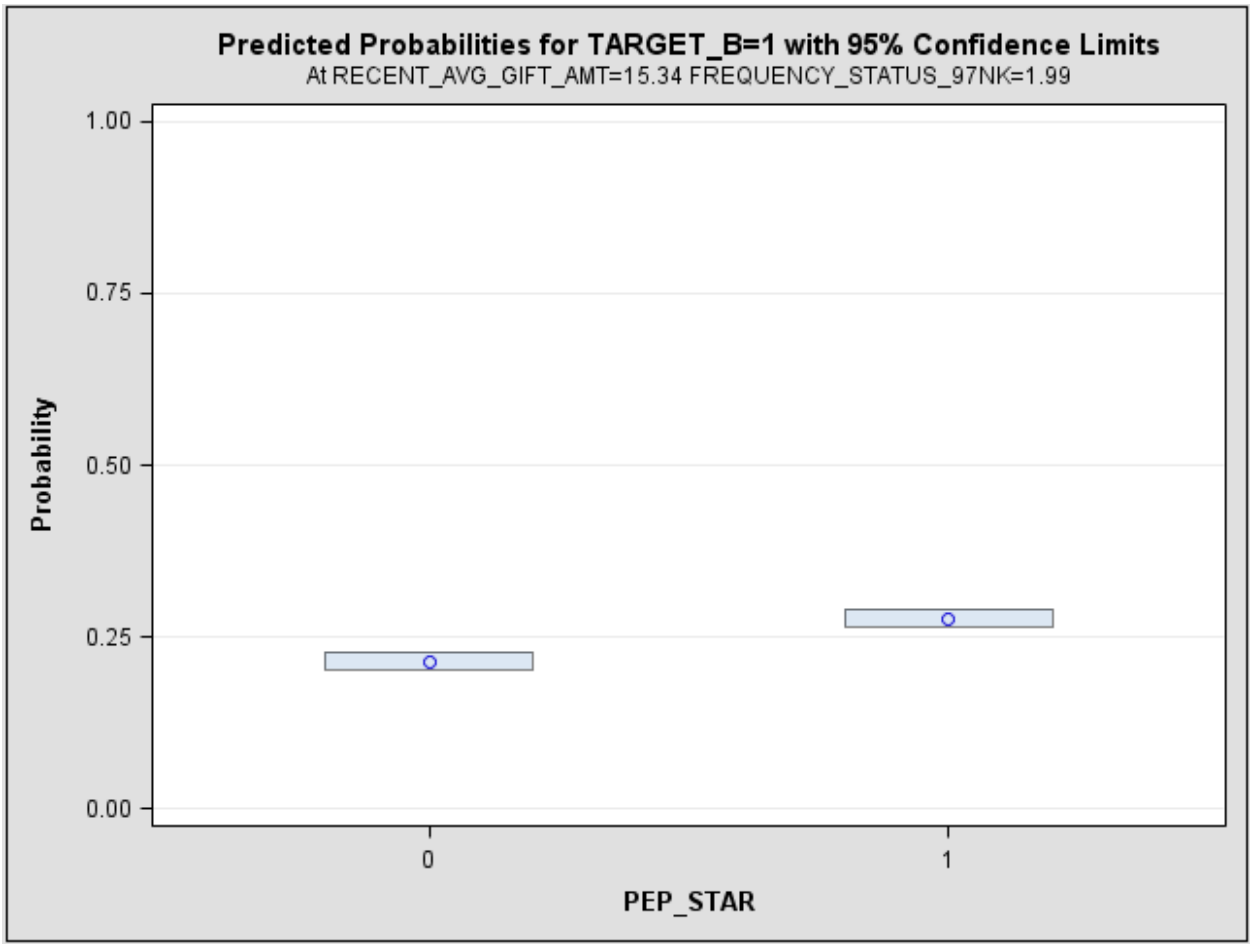


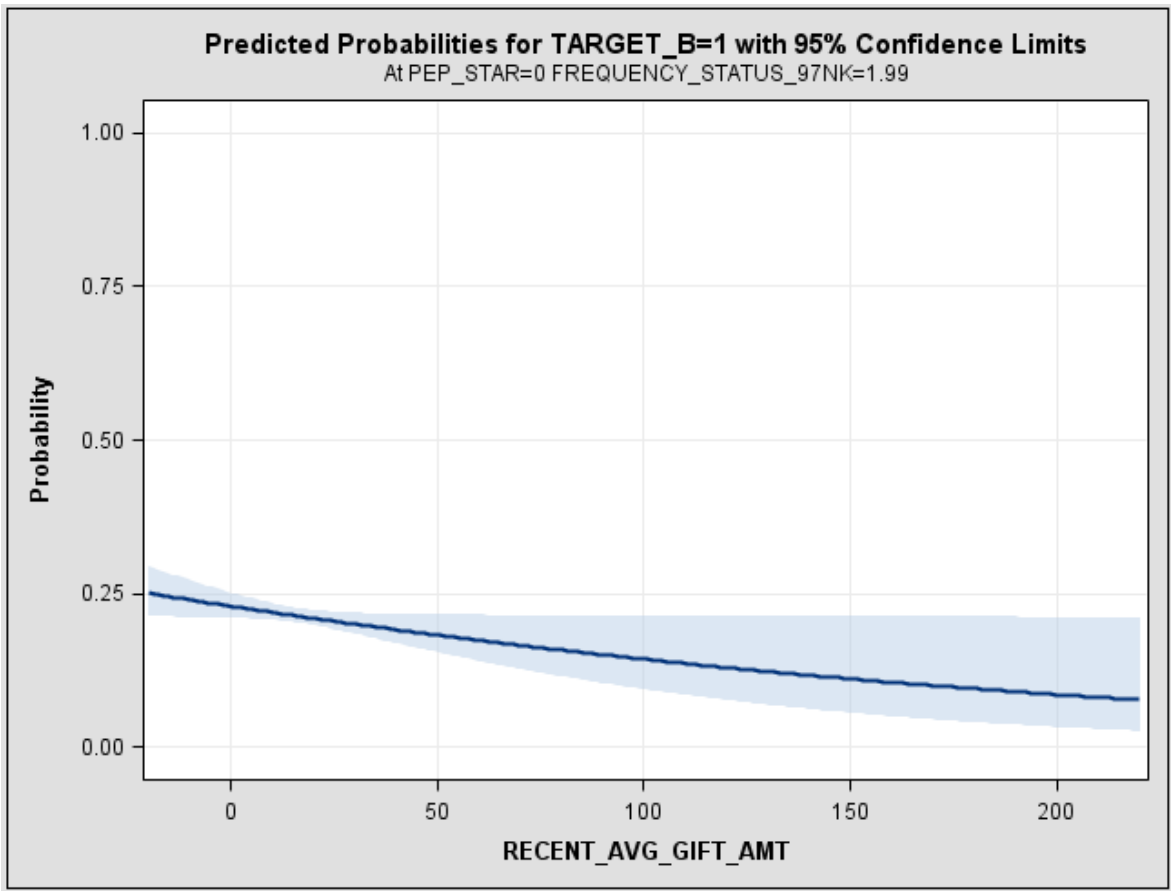
- Interpret the odds ratio for **PEP_STAR**.
  The odds ratio for **PEP_STAR** indicates that customers who are consecutive donors have 1.40 times the odds of responding to a solicitation compared to customers who are not consecutive donors.



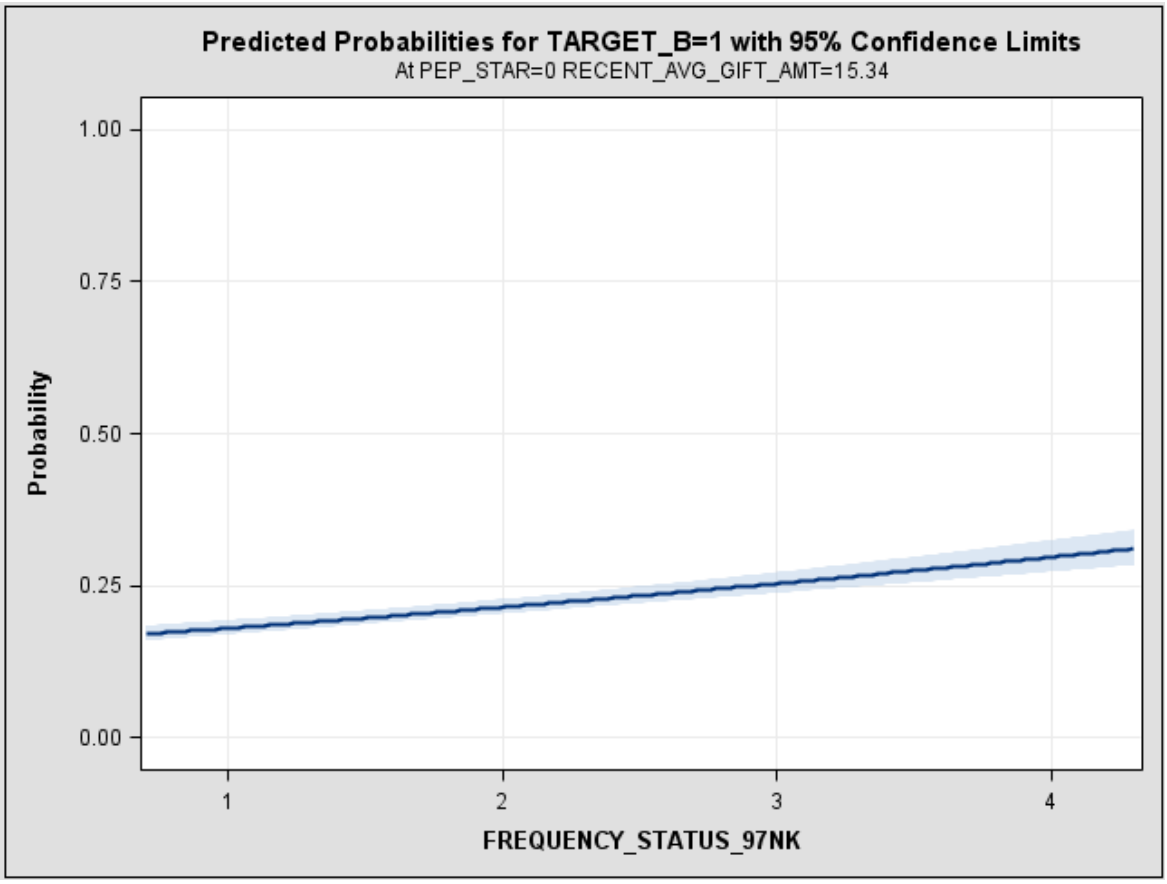
- Interpret the effect plot of **RECENT_AVG_GIFT_AMT** by **PEP_STAR**.
  The effect plot for **RECENT_AVG_GIFT_AMT** by **PEP_STAR** shows a negative relationship between the average donation amount to promotions since June 1994 and the predicted probabilities of responding to the solicitation in June of 1997. The consecutive donors have the higher probabilities across all values of **RECENT_AVG_GIFT_AMT**.







- Interpret the effect plot for **RECENT_AVG_GIFT_AMT**. The effect plot for **RECENT_AVG_GIFT_AMT** shows a negative relationship between the average donation amount to promotions since June 1994 and the predicted probabilities of responding to the solicitation in June of 1997 with the widest confidence intervals corresponding to the largest values of **RECENT_AVG_GIFT_AMT**.




```
title  "Adjusted Predicted Probabilities of the Veteran's "
       "Organization Data";
proc print data=scopva train(obs=10);
   var p_1 pep_star recent_avg_gift_amt frequency_status_97nk;
run;
```

Adjusted Predicted Probabilities of the Veteran's Organization Data

Obs	P_1	PEP_STAR	RECENT_ AVG_GIFT_ AMT	FREQUENCY_ STATUS_ 97NK
1	0.046390	1	15.00	1
2	0.033094	0	17.50	1
3	0.064890	0	8.33	4
4	0.090167	1	5.00	4
5	0.059152	1	8.33	2
6	0.058117	1	11.57	2
7	0.046941	1	12.86	1
8	0.031733	0	25.00	1
9	0.045126	1	20.00	1
10	0.032091	0	23.00	1

End of Solutions

Solutions to Student Activities (Polls/Quizzes)

2.01 Multiple Choice Poll – Correct Answer

The odds ratio for a $1000 increase in income is 1.074. What does this mean for every $1000 increase in income?

a. The probability of the event increases by 7.4%.
b. The logit increases by 7.4%.
c. The odds of the event increases by 7.4%.
d. The log of the odds of the event increases by 7.4%.

15

2.02 Multiple Choice Poll – Correct Answer

If the value for the offset is 3.2567, then the model corrected for oversampling has which of the following?

a. An intercept that is 3.2567 lower in value compared to the model fitted to the biased sample
b. Probabilities that are 3.2567% higher than the probabilities from the model fitted to the biased sample
c. Probabilities that are 3.2567% lower than the probabilities from the model fitted to the biased sample
d. Parameter estimates that are 3.2567% lower than the parameter estimates from the model fitted to the biased sample

25

2.03 Multiple Choice Poll – Correct Answer

The c statistic was 0.604. This can be interpreted as the probability of which of the following?

a. A customer donating to the national veterans' organization

b. A customer not donating to the national veterans' organization

c. If one person is randomly selected from the group of donors and another person is randomly selected from the non-donors, 60.4% of the time, the donor has the higher predicted probability of the target event as given by this model.

Chapter 3 Preparing the Input Variables

3.1 Missing Values

Objectives

- Explain the problem of missing values.
- Describe the usefulness of missing indicator variables.
- Demonstrate how to impute missing values in the STDIZE procedure.

3

Does Pr(missing) Depend on the Data?

- No
  - MCAR
- Yes
  - that unobserved value
  - other unobserved values
  - other observed values
    - including the target

14	2	2
67	1	4
?	3	1
33	1	7
18	2	1
6	0	1
31	3	8
51	1	8

4

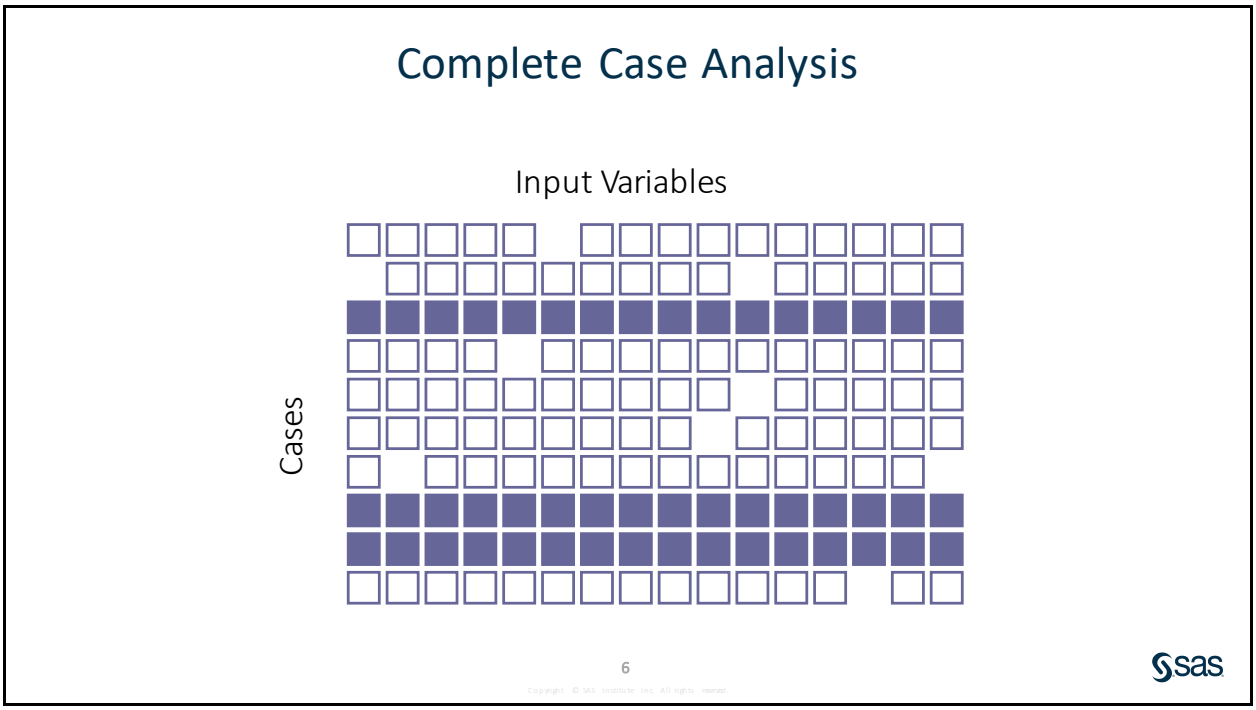
Missing values of the input variables can arise from several mechanisms (Little 1992). A value is *missing completely at random* (MCAR) if the probability that it is missing is independent of the data. MCAR is a particularly easy mechanism to manage but is unrealistic in most predictive modeling applications.

The probability that a value is missing might depend on the unobserved value. Credit applicants with fewer years at their current job might be less inclined to provide this information.

An even more problematic missing-value mechanism occurs when the probability that a value is missing depends on values of unobserved (lurking) predictors. Transient customers might have missing values on several variables.

The probability that a value is missing might depend on observed values of other input variables. Customers with longer tenures might be less likely to have certain historic transactional data. Missingness might depend on a combination of values of correlated inputs.

A fundamental concern for predictive modeling is that the missingness is related to the target. Customers that are more transient might be the best prospects for a new offer.



The default method for treating missing values in most SAS modeling procedures (including the LOGISTIC procedure) is complete-case analysis. In *complete-case analysis*, only those cases without any missing values are used in the analysis.

Complete-case analysis has some moderately attractive theoretical properties even when the missingness depends on the observed values of other inputs (Donner 1982, Jones 1996). However, complete-case analysis has serious practical shortcomings regarding predictive modeling. Even a small amount of missing values can cause an enormous loss of data in high dimensions. For example, suppose each of the *k* input variables can be MCAR with probability α. In this situation, the expected proportion of complete cases is $(1-\alpha)^k$.

Therefore, a 1% probability of missing (α=.01) for 100 inputs leaves only 37% of the data for analysis, 200 leaves 13%, and 400 leaves 2%. If the missingness were increased to 5% (α=.05), then <1% of the data is available with 100 inputs.

New Missing Values

Fitted Model:

$$\text{logit}(\hat{p}) = -2.1 + .072\,x_1 - .89\,x_2 - 1.4\,x_3$$

New Case: $(x_1,\ x_2,\ x_3) = (2,\ ?,\ -.5)$

Predicted Value:

$$\text{logit}(\hat{p}) = -2.1 + .144 - .89(\ ) + .7$$

7

Another practical consideration of any treatment of missing values is *scorability* (the practicality of the method when it is deployed). The purpose of predictive modeling is scoring new cases. How does a model built on the complete cases score a new case if it has a missing value? To decline to score new incomplete cases is practical only if there were a very small number of missing values.

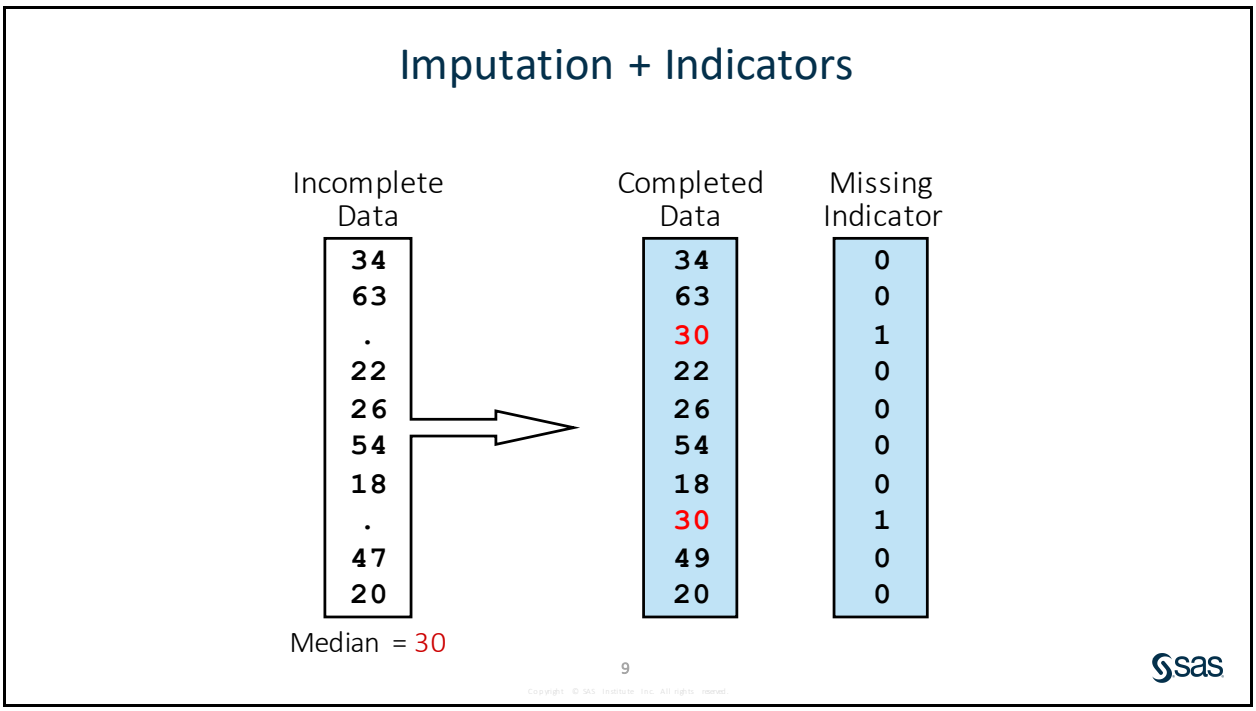
Missing Value Imputation

6	03	2.6	0	8.3	42	66	C03
12	04	1.8	0	0.5	86	65	C14
6.5	01	2.3	.33	4.8	37	66	C00
8	01	2.1	1	4.8	37	64	C08
6	01	2.8	1	9.6	22	66	C99
3	01	2.7	0	1.1	28	64	C00
2	02	2.1	1	5.9	21	63	C03
10	03	2.0	0	0.8	0	63	C99
7	01	2.5	0	5.5	62	67	C12
6.5	01	2.4	0	0.9	29	63	C05

8

Because of the previously mentioned drawbacks of complete-case analysis, some type of missing value imputation is necessary. Imputation means filling in the missing values with some reasonable value. Many methods were developed for imputing missing values (Little 1992). The principal consideration for most methods is the ability to obtain valid statistical inference on the imputed data, not generalization.

Subject-matter knowledge is often important for proper handling of missing value imputation. For example, it is somewhat common to see missing values incorrectly coded as zeros.



One reasonable strategy for handling missing values in predictive modeling is to do the following steps.

1. Create missing indicators for each variable with missing values

$$MI_j = \begin{cases} 1 & \text{if } x_j \text{ is missing} \\ 0 & \text{otherwise} \end{cases}$$

   and treat them as new input variables in the analysis.

2. Use median imputation for numeric inputs. Fill the missing value of x_j with the median of the complete cases for that variable.

3. Create a new level that represents missing (unknown) for categorical inputs.

If a very large percentage of values is missing (>50%), then the variable might be better handled by omitting it from the analysis or by creating the missing indicator only. If a very small percentage of the values is missing (<0.1%), then the missing indicator is probably of little value.

This strategy is somewhat unsophisticated but satisfies the two most important considerations in predictive modeling:

- efficient scorability
- capturing the relationship of missingness with the target

A new case is easily scored.

1. Replace the missing values with the medians from the development data.
2. Apply the prediction model.

There is a large amount of statistical literature about different missing value imputation methods, including discussions of the drawbacks of mean and median imputation and missing indicators (Donner 1982, Jones 1997). However, most of the advice is based on considerations that are peripheral to predictive modeling. There is little advice in these situations:

- when the functional form of the model is not assumed to be perfectly specified
- when the goal is to obtain good predictions that can be practically applied to new cases
- when *p*-values and hypothesis tests are of secondary importance
- when the missingness might be highly pathological, in other words, depending on lurking predictors

3.01 Multiple Choice Poll

Which of the following statements is ***true*** regarding missing values in predictive modeling applications?

a. In complete case analysis, the loss of data is inconsequential when the missing values are spread across many variables.

b. Observations with missing values are scored in PROC LOGISTIC.

c. Missing indicator variables can be used to capture the relationship between the target variable and missing inputs.

d. The missing completely at random assumption is valid in most predictive modeling applications.

10

Imputing Missing Values

Example: Create missing value indicator variables and replace missing values with the variable median.

```
/*  pmlr03d01.sas  */

title1 "Variables with Missing Values";
proc print data=work.train(obs=15);
   var ccbal ccpurc income hmown;
run;
```

Variables with Missing Values

Obs	CCBal	CCPurc	Income	HMOwn
1	0.00	1	4	1
2	65.76	0	125	1
3	85202.99	0	55	1
4	.	.	20	0
5	0.00	0	25	1
6	0.00	0	8	1
7	0.00	0	100	1
8	323.13	0	13	1
9	32366.86	0	.	1
10	0.00	0	9	0
11	1378.46	1	60	1
12	.	.	25	0
13	0.00	0	54	0
14	1466.87	0	45	0
15	.	.	31	0

Fifteen of the input variables were selected for imputation. Two arrays are created. One is called *MI*, which contains the missing value indicator variables. The other is called *X*, which contains the input variables. It is critical that the order of the variables in the array MI match the order of the variables in array X. Defining the dimension with an asterisk causes the array elements to be automatically counted. In the DO loop, the DIM function returns the dimension of the array. Thus, the DO loop executes 15 times in this example. The assignment statement inside the DO loop causes the entries of MI to be 1 if the corresponding entry in X is missing and 0 otherwise.

```
data work.train mi(drop=i);
   set work.train;
   /* name the missing indicator variables */
   array mi{*} MIAcctAg MIPhone MIPOS MIPOSAmt
               MIInv MIInvBal MICC MICCBal
               MICCPurc MIIncome MIHMOwn MILORes
               MIHMVal MIAge MICRScor;
   /* select variables with missing values */
   array x{*} acctage phone pos posamt
              inv invbal cc ccbal ccpurc income
              hmown lores hmval age crscore;
   do i=1 to dim(mi);
      mi{i}=(x{i}=.);
      nummiss+mi{i};
   end;
run;
```

The STDIZE procedure with the REPONLY option can be used to replace only the missing values. A variable value remains unchanged if it was not missing. The METHOD= option enables you to choose several location measures such as the mean, median, and midrange. The output data set created by the OUT= option contains all the variables in the input data set where the variables listed in the VAR statement are imputed. Only numeric input variables should be used in PROC STDIZE.

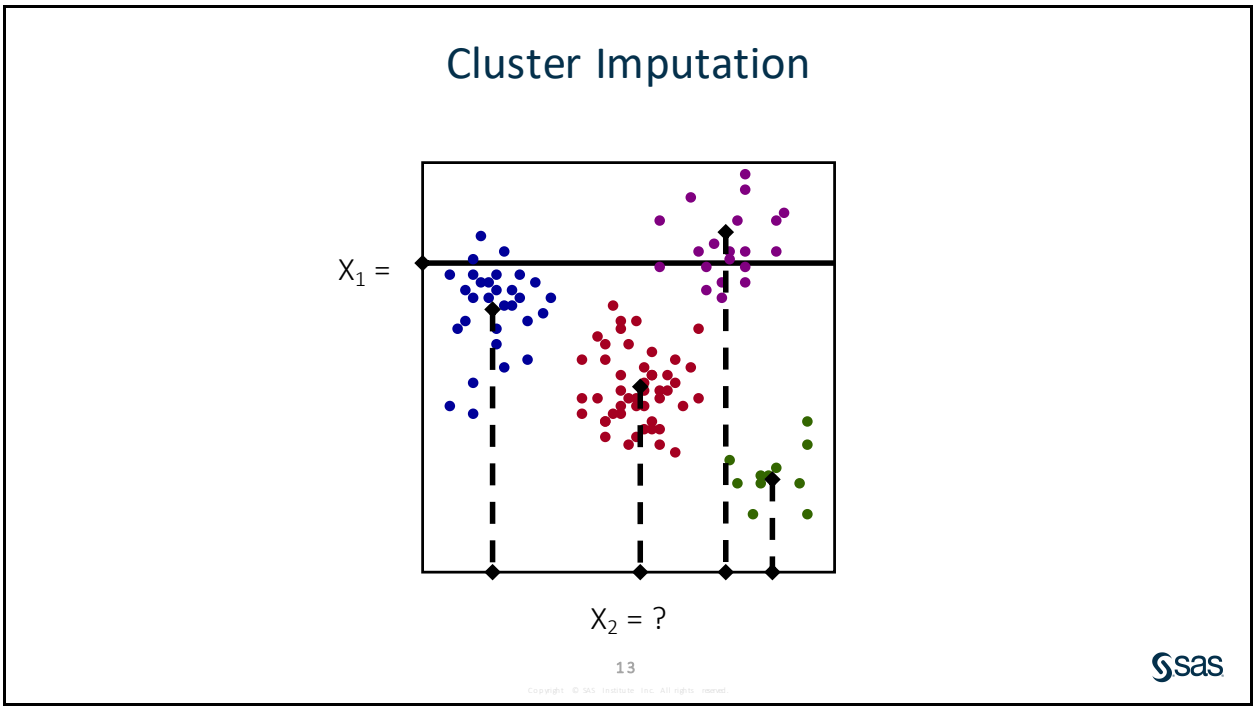
```
proc stdize data=work.train_mi
            reponly
            method=median
            out=work.train_imputed;
   var &inputs;
run;

title1 "Imputed Values with Missing Indicators";
proc print data=work.train imputed(obs=12);
   var ccbal miccbal ccpurc miccpurc
       income miincome hmown mihmown nummiss;
run;
```

Note: The REPLACE option in PROC STANDARD can be used to replace missing values with the mean of that variable on the nonmissing cases.

Imputed Values with Missing Indicators

Obs	CCBal	MICCBal	CCPurc	MICCPurc	Income	MIIncome	HMOwn	MIHMOwn	nummiss
1	0.00	0	1	0	4	0	1	0	0
2	65.76	0	0	0	125	0	1	0	0
3	85202.99	0	0	0	55	0	1	0	0
4	0.00	1	0	1	20	0	0	0	8
5	0.00	0	0	0	25	0	1	0	8
6	0.00	0	0	0	8	0	1	0	9
7	0.00	0	0	0	100	0	1	0	9
8	323.13	0	0	0	13	0	1	0	9
9	32366.86	0	0	0	35	1	1	0	13
10	0.00	0	0	0	9	0	0	0	13
11	1378.46	0	1	0	60	0	1	0	13
12	0.00	1	0	1	25	0	0	0	21

End of Demonstration



Mean imputation uses the unconditional mean of the variable. An attractive extension would be to use the mean conditional on the other inputs. This is referred to as regression imputation. Regression imputation would usually give better estimates of the missing values. Specifically, *k* linear regression models could be built, one for each input variable using the other inputs as predictors. This would presumably give better imputations and be able to accommodate missingness that depends on the values of the other inputs. However, a complication of regression imputation is that the other inputs might themselves have missing values. Consequently, the *k* imputation regressions also need to accommodate missing values.

Cluster-mean imputation is a somewhat more practical alternative.

1. Cluster the cases into relatively homogenous subgroups.
2. Conduct mean-imputation within each group.
3. For new cases with multiple missing values, use the cluster mean that is closest in all the nonmissing dimensions.

This method can accommodate missingness that depends on the other input variables. This method is implemented in SAS Enterprise Miner. For large data sets, the FASTCLUS procedure might also be appropriate. (See the appendix.)

A simpler but sometimes useful alternative is to define *a priori* segments (for example, high, middle, low, and unknown income), and then do mean or median imputation within each segment.

Exercises

1. Missing Value Imputation

a. Include the program **pmlr01s01.sas** or use the **PMLR_UpToDate** macro. Using the **pmlr.pva_train** data set, create missing value indicators for the inputs **DONOR_AGE**, **INCOME_GROUP**, and **WEALTH_RATING**. Call the new data set **pmlr.pva_train_mi**.

b. Submit the program below to group the variables **RECENT_RESPONSE_PROP** and **RECENT_AVG_GIFT_AMT** into three groups.

```
proc rank data=pmlr.pva_train_mi  out=pva_train_rank  groups=3;
   var recent response prop recent_avg_gift_amt;
   ranks grp_resp grp_amt;
run;
```

c. Sort **pva_train_rank** by **GRP_RESP** and **GRP_AMT** and name the output data set **pva_train_rank_sort**.

d. Use PROC STDIZE with a BY statement to impute missing values for each BY group and write the completed data set to output. Name the output data set **pva_train_imputed**.

e. Use the MEANS procedure to determine the values that replaced the missing values.

For the cell **GRP_RESP**=0 and **GRP_AMT**=0, what replaced the missing value for **DONOR_AGE**?

Note: The RANK procedure with the GROUPS= option bins variables into quantiles. The VAR statement lists the variables to be grouped. The RANKS statement names the group indicators in the OUT= data set.

End of Exercises

3.02 Multiple Choice Poll

For the cell GRP_RESP=0 and GRP_AMT=0, what replaced the missing value for DONOR_AGE?

a. 57
b. 65
c. 58
d. 68

15

3.2 Categorical Inputs

Objectives

- Explain the problem of having numerous levels in a categorical input.
- Illustrate how to cluster levels of a categorical input using the CLUSTER procedure.
- Demonstrate how to use smoothed weight of evidence coding to convert a categorical predictor to a continuous predictor.

19

Dealing with Categorical Inputs

- When a categorical input has many levels, expanding the input into dummy variables can greatly increase the dimension of the input space.
- Including categorical inputs in the model can cause quasi-complete separation.

20

The typical approach when you work with categorical inputs is to dummy-code the levels. In this way, a predictive model can adjust the predicted response to accurately reflect the response difference between levels. However, expanding categorical inputs into dummy variables can greatly increase the dimension of the input space and produces many redundant and irrelevant inputs in the model. Creating many dummy-coded inputs might lead to the problem of over-fitting, where the predictive model generalizes poorly to a new data set.

Quasi-Complete Separation

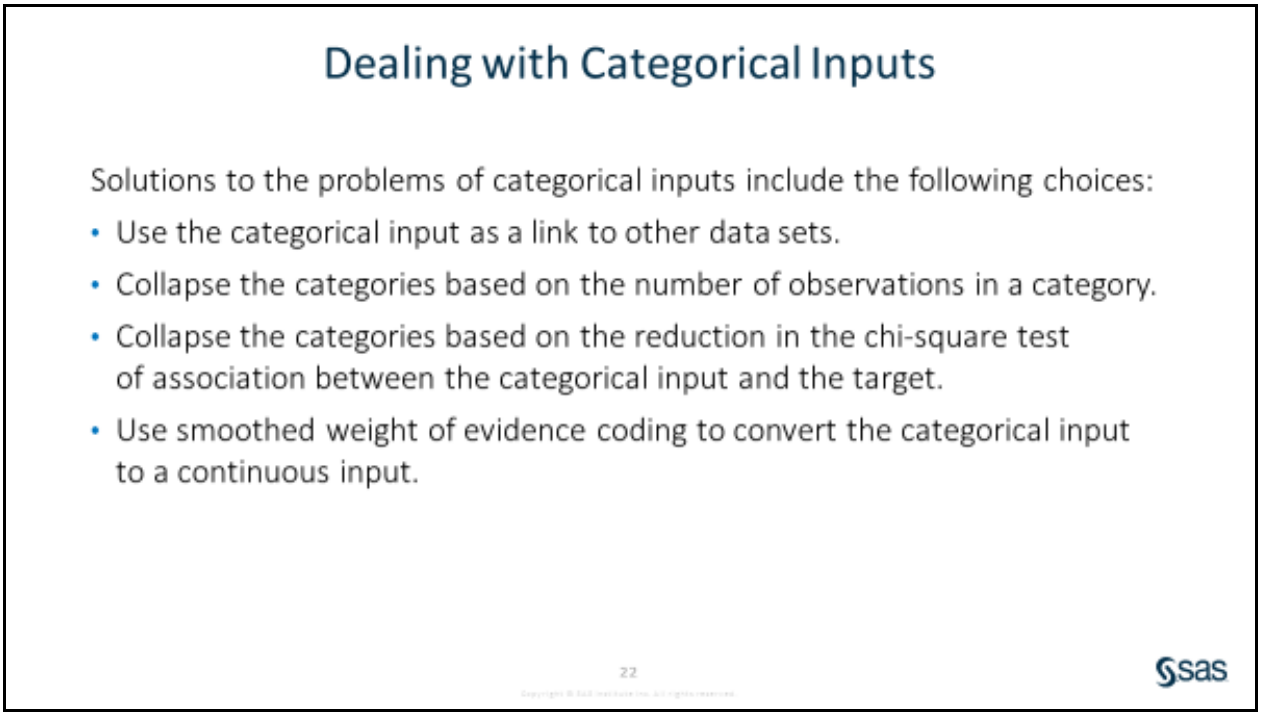
	0	1	D_A	D_B	D_C	logit
A	28	7	1	0	0	-1.39
B	16	0	0	1	0	-Infinity
C	94	11	0	0	1	-2.14
D	23	21	0	0	0	-0.08

21

Quasi-complete separation occurs when a level of the categorical input has a target event rate of 0 or 100%. The coefficient of a dummy variable represents the difference in the logits between that level and the reference level. When quasi-complete separation occurs, one or more of the logits is infinite. The likelihood does not have a maximum in at least one dimension, so the ML estimate of that coefficient is infinite. If the zero-cell category is the reference level, then all the coefficients for the dummy variables are infinite.

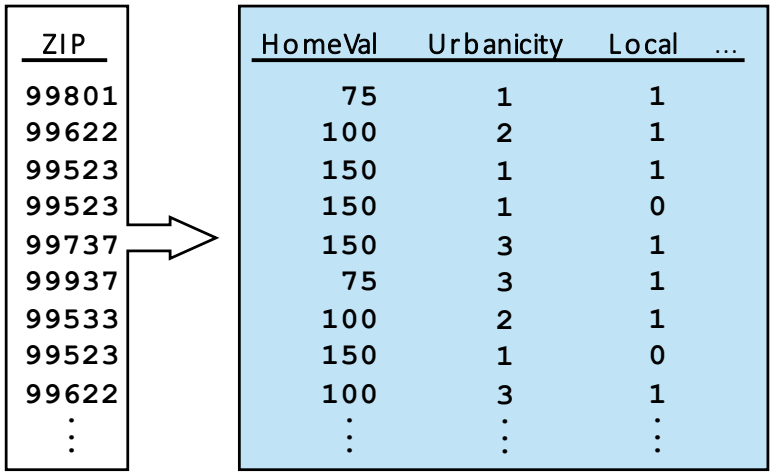
Quasi-complete separation complicates model interpretation. It can also affect the convergence of the estimation algorithm. Furthermore, it might lead to incorrect decisions regarding variable selection.

The most common cause of quasi-complete separation in predictive modeling is categorical inputs with rare categories.



One solution to the problem of categorical inputs, which is not recommended, is to enumerate the levels as 1, 2, 3, and so on. Most models assume a linear association between the continuous input and the target. If the categorical levels were enumerated, two adjacent levels can have completely different response behavior, which violates the linearity assumption.

Smarter Variables



24

A preferable method is to use subject-matter information to create new inputs that represent relevant sources of variation. A categorical input might be best considered as a link to other data sets. For example, geographic areas are often mapped to several relevant demographic variables.

Thresholding

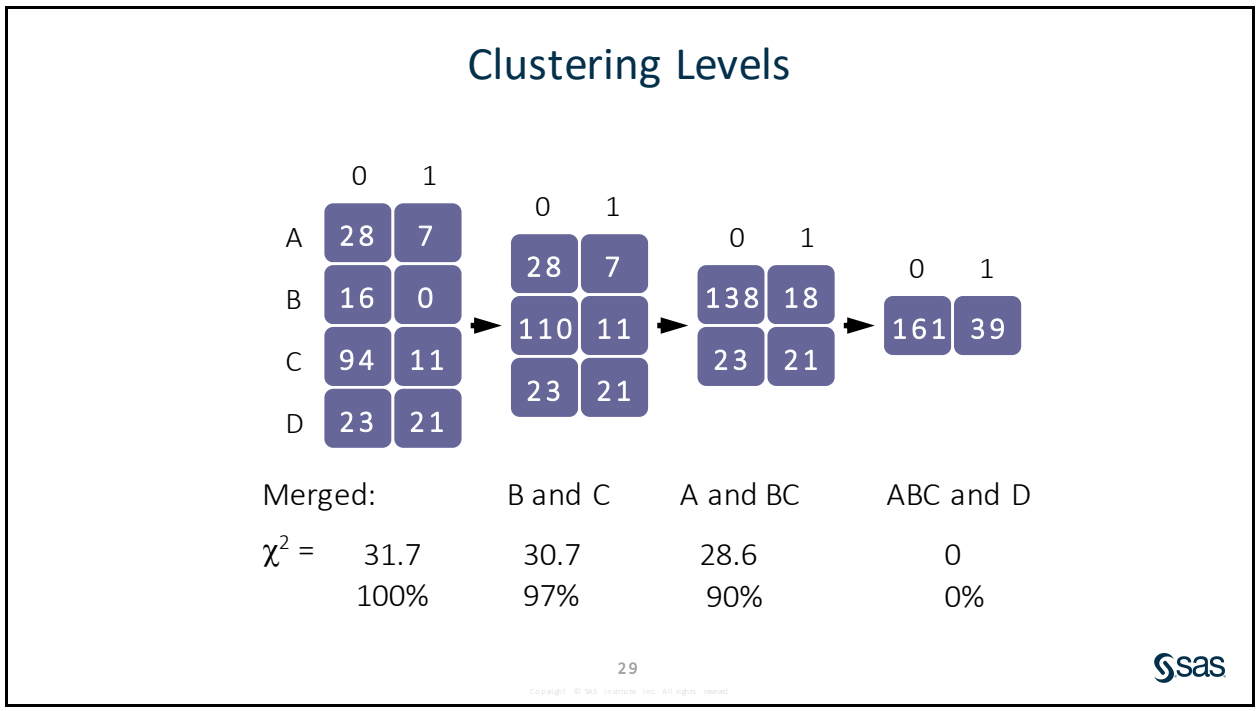
Level	Number of observations
A	1562
B	970
C	223
D	111
E	85
F	23
G	17
H	12
I	5

Combine to a single new level "OTHER" and then dummy-code six categories.

25

Another strategy for dealing with categorical inputs with many levels is *thresholding*. This method requires a minimum number of cases in a level before creating a dummy-code input (the threshold is 50 in the slide above). Any level that fails to meet this minimum threshold is relegated to a new level called *OTHER*. If you reduce the number of inputs that are available to a model, thresholding limits the model's ability to discover spurious input-target associations.

Note: A thresholding macro is shown in the appendix.



The best remedy for sparseness in the levels is to collapse the levels of a categorical input. Ideally, subject-matter considerations should be used to collapse levels. This is not always practical in predictive modeling.

A simple data-driven method for collapsing levels of contingency tables was developed by Greenacre (1988, 1993). The levels (rows) are hierarchically clustered based on the reduction in the chi-square test of association between the categorical variable and the target. At each step, the two levels that give the least reduction in the chi-square statistic are merged. The process is continued until the reduction in chi-square drops below some threshold (for example, 99%). This method quickly puts rare categories in with other categories that have similar marginal response rates. Although this method is simple and effective, there is a potential loss of information because only univariate associations are considered.

3.03 Multiple Choice Poll

Which of the following statements is ***true*** regarding Greenacre's method for collapsing levels of contingency tables?

a. The levels are hierarchically clustered where the levels that give the largest reduction in the chi-square test of association are merged.
b. Levels with similar marginal response rates are merged.
c. The method does not account for the sample size in each level.
d. The method is appropriate for any categorical input.

Clustering Levels of Categorical Inputs

Example: Use PROC MEANS to calculate the response rate and frequency in each of the levels of **branch**. Then use the CLUSTER procedure to cluster the levels of **branch** using Greenacre's method and to create a high resolution dendrogram. Use the DATA step to compute the log of the *p*-value for each collapsed contingency table of the branch levels and find the cluster solution with the lowest *p*-value. Use the TREE procedure to create a SAS data set with the final cluster solution.

The levels of a categorical input can be clustered using Greenacre's method (1988, 1993) in PROC CLUSTER. The procedure was designed for general clustering applications, but with some simple pre-processing of the data, it can be made to cluster levels of categorical variables.

The variable **Branch** has 19 levels. The first step is to create a data set that contains the proportion of the target event (**Ins**) and number of cases in each level. The NWAY option caused the output data set to have 19 observations, one for each of the 19 levels. The output data set also has a variable **prop** for the proportion of target events. It automatically creates the variable **_FREQ_**, which counts the number of cases in each level.

```
/* pmlr03d02.sas */

proc means data=work.train_imputed noprint nway;
   class branch;
   var ins;
   output out=work.level mean=prop;
run;

title1 "Proportion of Events by Level";
proc print data=work.level;
run;
```

Proportion of Events by Level

Obs	Branch	_TYPE_	_FREQ_	prop
1	B1	1	1930	0.36995
2	B10	1	182	0.41758
3	B11	1	160	0.41875
4	B12	1	368	0.36957
5	B13	1	369	0.40650
6	B14	1	712	0.19663
7	B15	1	1510	0.23179
8	B16	1	1040	0.28558
9	B17	1	544	0.34007
10	B18	1	370	0.36757
11	B19	1	191	0.38743
12	B2	1	3543	0.32882
13	B3	1	1905	0.38320
14	B4	1	3737	0.37329
15	B5	1	1819	0.38977
16	B6	1	938	0.36567
17	B7	1	948	0.32700

18	B8	1	878	0.38610
19	B9	1	368	0.36685

Using the FREQ statement and METHOD=WARD in PROC CLUSTER gives identical results to Greenacre's method. The OUTTREE= option creates an output data set that can be used by the TREE procedure to create a SAS data set with the final cluster solution. The PLOTS=DENDROGRAM requests a high resolution dendrogram with the VERTICAL option specifying a vertical dendrogram and the HEIGHT=RSQ option specifying that the squared multiple correlation be used as the method to draw the height of the dendrogram. The ID statement specifies a variable that identifies observations in the printed cluster history and in the OUTTREE= data set. The ODS OUTPUT statements create an output data set called **cluster** from the output object **clusterhistory** that is associated with the results of the clustering algorithm.

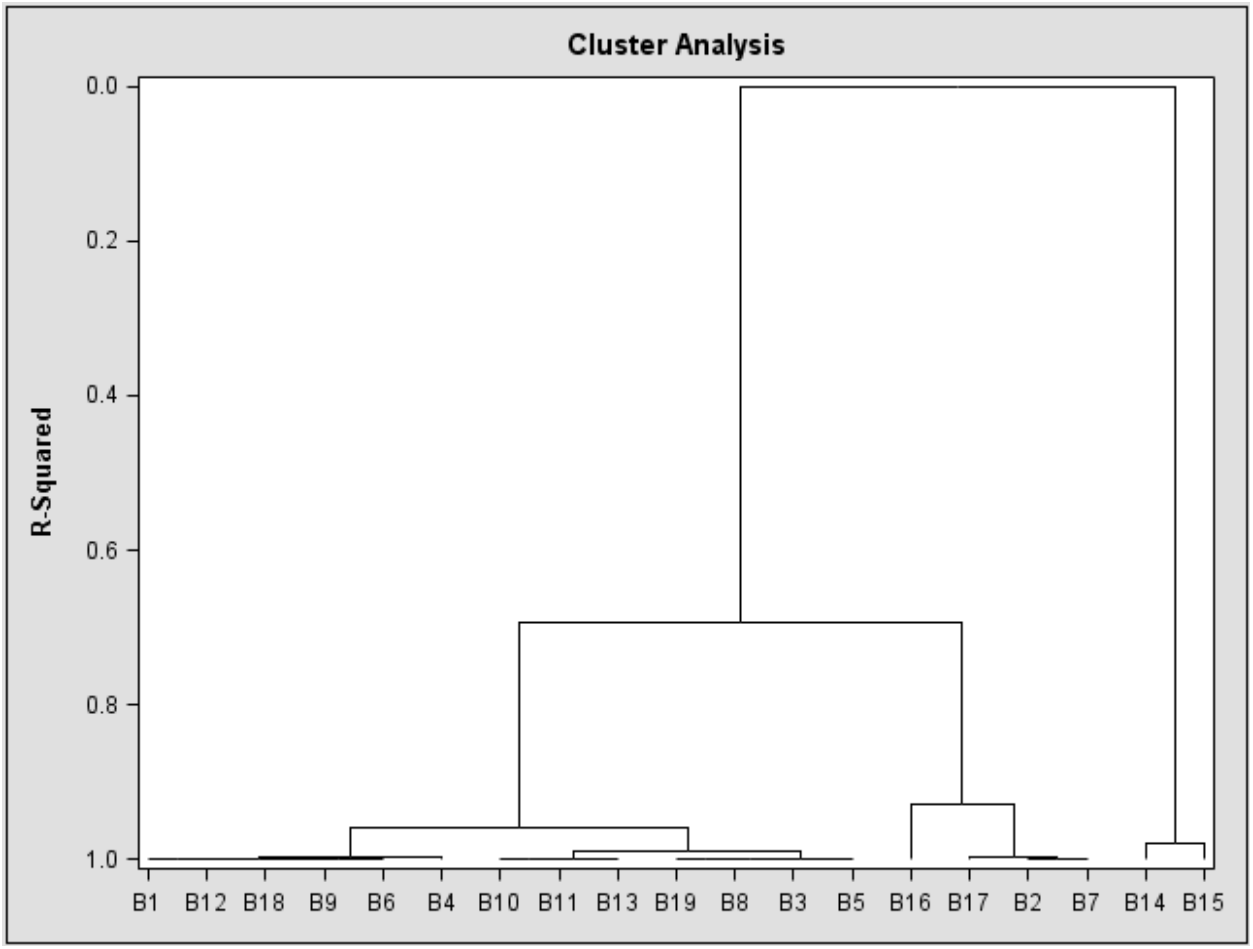
```
ods output clusterhistory=work.cluster;

proc cluster data=work.level method=ward outtree=fortree
        plots=(dendrogram(vertical height=rsq));
   freq _freq_;
   var prop;
   id branch;
run;
```

Partial Output

Cluster History

Number of Clusters	Clusters Joined		Freq	Semipartial R-Square	R-Square	Tie
18	B1	B12	2298	0.0000	1.00	
17	B18	B9	738	0.0000	1.00	
16	B10	B11	342	0.0000	1.00	
15	B19	B8	1069	0.0000	1.00	
14	CL17	B6	1676	0.0000	1.00	
13	B2	B7	4491	0.0000	1.00	
12	CL15	B3	2974	0.0001	1.00	
11	CL18	CL14	3974	0.0002	1.00	
10	CL16	B13	711	0.0004	.999	
9	CL12	B5	4793	0.0006	.999	
8	CL11	B4	7711	0.0008	.998	
7	B17	CL13	5035	0.0012	.997	
6	CL10	CL9	5504	0.0072	.989	
5	B14	B15	2222	0.0106	.979	
4	CL8	CL6	13215	0.0204	.958	
3	B16	CL7	6075	0.0297	.929	
2	CL4	CL3	19290	0.2350	.694	
1	CL2	CL5	21512	0.6937	.000	

The column labeled **RSquare** in the output is equivalent to the proportion of chi-squared in the 19×2 contingency table remaining after the levels are collapsed. At each step, the levels that give the smallest decrease in chi-squared are merged. The change in chi-squared is listed in the **Semipartial R-Square** column. The rows in the summary represent the results after the listed clusters were merged. The number of clusters is reduced from 18 to 1. When previously collapsed levels are merged, they are denoted using the CL as the prefix and the number of resulting clusters as the suffix. For example, at the fifth step, CL17 represents B18 and B9 that were merged at the second step creating 17 clusters.



The dendrogram shows that several branches can be combined with a minuscule reduction in chi-squared. The horizontal axis is also approximately ordered by the mean proportion of events in each cluster.

To calculate the optimum number of clusters, the chi-square statistic and the associated *p*-value needs to be computed for each collapsed contingency table. This information can be obtained by multiplying the chi-square statistic from the 19×2 contingency table with the proportion of chi-squared remaining after the levels are collapsed. The FREQ procedure is used to compute the chi-square statistic for the 19×2 contingency table.

```
proc freq data=work.train imputed noprint;
   tables branch*ins / chisq;
   output out=work.chi(keep=_pchi_) chisq;
run;
```

The DATA step computes the chi-square statistic for each collapsed contingency table. The **_n_** variable is used to put the overall chi-square value in each observation for the data set **cutoff**. The LOGSDF function computes the log of the probability that an observation from a specified distribution is greater than or equal to a specified value. The arguments for the function are the specified distribution in quotation marks, the numeric random variable, and the degrees of freedom. The log of the *p*-value is calculated in order to produce a more visually appealing graph.

```
data work.cutoff;
   if  n =1 then set work.chi;
   set work.cluster;
   chisquare= pchi *rsquared;
   degfree=numberofclusters-1;
   logpvalue=logsdf('CHISQ',chisquare,degfree);
run;
```

To plot the log of the *p*-value by the number of clusters, the SGPLOT procedure is used.

```
title1 "Plot of the Log of the P-Value by Number of Clusters";
proc sgplot data=work.cutoff;
   scatter y=logpvalue x=numberofclusters
           / markerattrs=(color=blue symbol=circlefilled);
   xaxis label="Number of Clusters";
   yaxis label="Log of P-Value" min=-120 max=-85;
run;
```

Selected SGPLOT procedure statements:

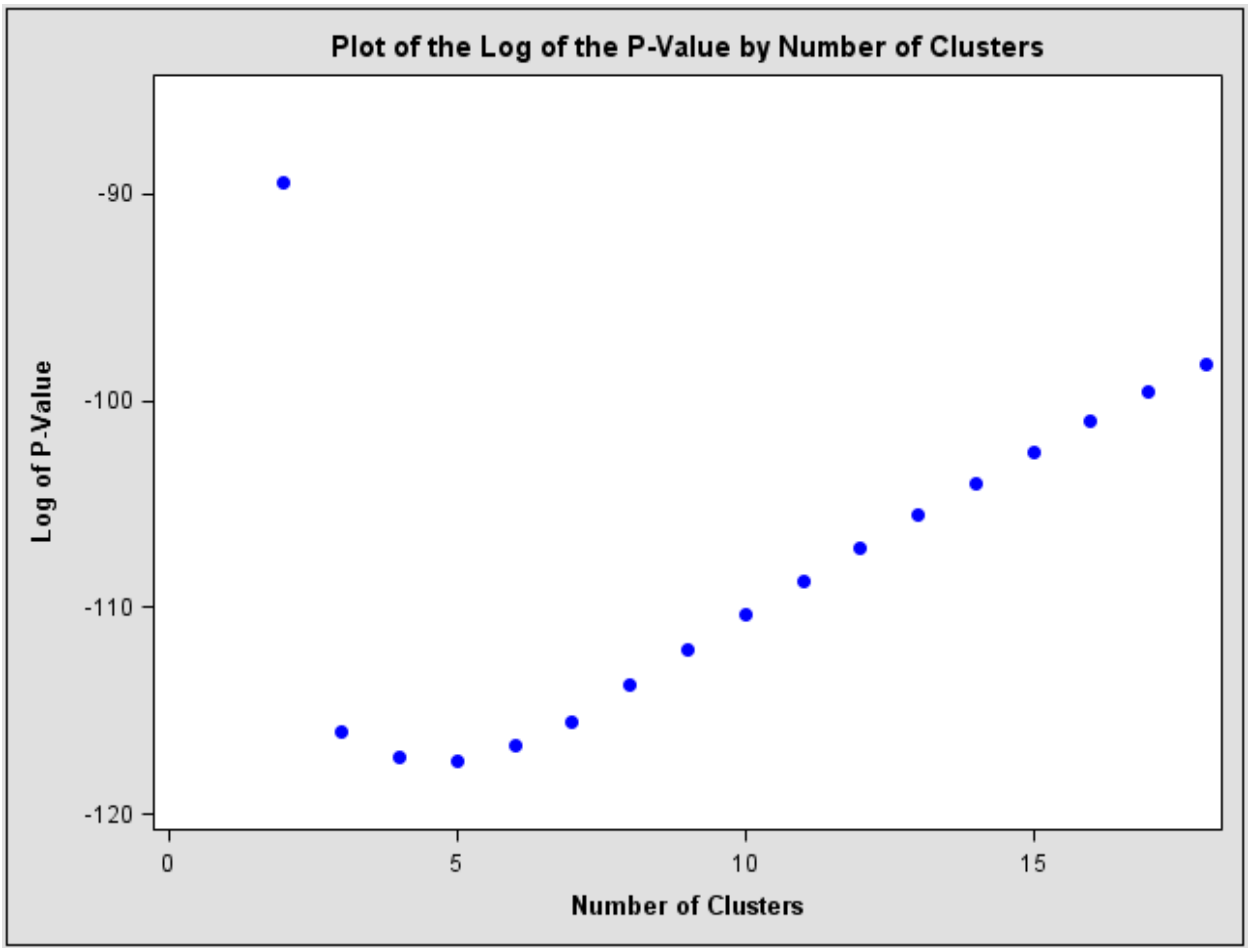
SCATTER	creates a scatter plot where the y= argument specifies the variable for the y-axis and the x= argument specifies the variable for the x-axis.
XAXIS	specifies options for the x-axis.
YAXIS	specifies options for the y-axis.

Selected SCATTER statement option:

MARKERATTRS=	specifies the appearance of the markers in the plot. You can specify the appearance by using a style element or by using suboptions. If you specify a style element, you can also specify suboptions to override specific appearance attributes.

Selected MARKETATTRS= suboptions:

COLOR=	specifies the color of the markers.
SYMBOL=	specifies the symbol for the markers.



The graph shows that the 5-cluster solution had the lowest *p*-value. To assign that value to the macro variable **NCL**, use the SQL procedure.

```
proc sql;
   select NumberOfClusters into :ncl
   from work.cutoff
   having logpvalue=min(logpvalue);
quit;
```

```
                                    Number
                                        of
                                  Clusters
                                  ________
                                         5
```

The TREE procedure produces a SAS data set with the 5 cluster solution. The input data set is the OUTTREE= data set produced in PROC CLUSTER. The **clus** data set from the OUT= option in PROC TREE shows which levels of **branch** are associated with each cluster (if the NCLUSTERS= option is correctly specified).

```
proc tree data=fortree nclusters=&ncl out=work.clus noprint;
   id branch;
run;
```

```
proc sort data=work.clus;
   by clusname;
run;

title1 "Levels of Branch by Cluster";
proc print data=work.clus;
   by clusname;
   id clusname;
run;
```

Levels of Branch by Cluster

CLUSNAME	Branch	CLUSTER
B16	B16	5
CL5	B14	4
	B15	4
CL6	B10	2
	B11	2
	B19	2
	B8	2
	B3	2
	B13	2
	B5	2
CL7	B2	3
	B7	3
	B17	3
CL8	B1	1
	B12	1
	B18	1
	B9	1
	B6	1
	B4	1

Rather than writing out, by hand, a series of rules like IF **branch** = 'B16' then branch_clus=5; ELSE IF… for many branches, you can use a DATA step to write the rules. An easy way to do this is to use a FILENAME statement to point to a file and then write the rules to that file using the PUT statement.

The FILENAME statement creates a *fileref*, **brclus**, which points to the physical file S:\workshop\branch_clus.sas. The DATA_NULL_ statement enables you to use the DATA step processing without being required to write out a data set. The FILE statement specifies where you want to put the output of the DATA step (much as the INFILE statement would enable you to specify the input source for a DATA step). The SET statement brings in the data set **clus**, which shows which levels of **branch** are associated with each cluster. The option END= creates an internal flag that you can use to identify the last record of the data set. You could write IF…THEN…ELSE syntax to do the branch assignment. The following example uses SELECT…WHEN…OTHERWISE to perform the same task. The last record captures anyone without a branch assignment and puts it in a category called 'U'.

```
filename brclus "&PMLRfolder\branch_clus.sas";

data _null_;
   file brclus;
   set work.clus end=last;
   if _n_ = 1 then put "select (branch);";
   put "  when ('" branch +(-1) "') branch_clus = '" cluster +(-1)
       "';";
   if last then do;
      put "  otherwise branch_clus = 'U';" / "end;";
   end;
run;
```

The code generated by the DATA Step is shown below. The negative one in the syntax is necessary to eliminate an extra space in the parentheses and in the quotation marks.

```
select (branch);
  when ('B16') branch_clus = '5';
  when ('B14') branch_clus = '4';
  when ('B15') branch_clus = '4';
  when ('B10') branch_clus = '2';
  when ('B11') branch_clus = '2';
  when ('B19') branch_clus = '2';
  when ('B8') branch_clus = '2';
  when ('B3') branch_clus = '2';
  when ('B13') branch_clus = '2';
  when ('B5') branch_clus = '2';
  when ('B2') branch_clus = '3';
  when ('B7') branch_clus = '3';
  when ('B17') branch_clus = '3';
  when ('B1') branch_clus = '1';
  when ('B12') branch_clus = '1';
  when ('B18') branch_clus = '1';
  when ('B9') branch_clus = '1';
  when ('B6') branch_clus = '1';
  when ('B4') branch_clus = '1';
  otherwise branch_clus = 'U';
end;
```

```
data work.train_imputed_greenacre;
   set work.train imputed;
   %include brclus / source2;
run;
```

End of Demonstration

Smoothed Weight of Evidence

- The weight of evidence (WOE) is a technique that replaces the values of a category with the log(odds) of the event.
- Smoothed weight of evidence (SWOE) is a technique that averages the observed log(odds) of the event in a particular group with the log(odds) in the overall population.

$$\ln\left(\frac{\#events + c\rho_1}{\#nonevents + c(1-\rho_1)}\right)$$

where c is a smoothing parameter.

33

Another technique to deal with categorical inputs is to replace the levels with a single column that represents the event rate for each category. If the values are replaced with the log(odds) of the event, then the technique is called *weight of evidence* (WOE). This typically overfits the training data, so a more general approach considers sampling variability in some way. One recommended technique is to use the smoothed weight of evidence, which is similar to averaging the observed log(odds) in a particular group with the log(odds) in the overall sample. A smoothing parameter called c is used, in which if the c value is larger, the smoothed WOE is closer to the sample WOE (that is, a large c smooths the data aggressively). A small c is more sensitive to the observed data.

Note: Some techniques use the column totals rather than the row totals to compute the weight of evidence.

Note: If the target variable is interval, a Bayesian-inspired estimate for the target mean is calculated in place of the smoothed weight of evidence. The estimate can be thought of as a weighted average of the overall target mean and the observed target mean in level i. It is given by the following formula:

$$smoothmean = (smooth * \bar{y} + n_i * \bar{y}_i) / (smooth + n_i)$$

where $\bar{y}$ is the overall target mean, $\bar{y}_i$ is the level i target mean, n_i is the number of cases in level i, and *smooth* is the value of the smoothing parameter.

Smoothed Weight of Evidence

	0	1	SWOE
A	28	7	-1.399
B	16	0	-2.021
C	94	11	-1.978
D	23	21	-0.499

where rho_1=0.195 and c=24

This slide shows the computed smoothed weight of evidence (SWOE) for an example shown earlier. Notice that the quasi-complete separation problem is eliminated. A simulation study (Georges 2014) showed that higher cardinality inputs favored the SWOE compared to dummy-coding, WOE, and thresholding.

Computing Smoothed Weight of Evidence

Example: Use PROC SQL to compute the proportion of events in the training data set. Then use PROC MEANS to calculate the response rate and frequency in each of the levels of **branch** and a DATA step to compute the smoothed weight of evidence for each **branch** level. Use a value of 24 for *c* and assign the overall logit to any observation with an undefined branch.

```
/* pmlr03d03.sas */

%global rho1;
proc sql noprint;
   select mean(ins) into :rho1
   from work.train_imputed;
run;

proc means data=work.train_imputed sum nway noprint;
   class branch;
   var ins;
   output out=work.counts sum=events;
run;
```

The code below writes a function that transforms a category into a numeric value. The PUT statement is used to write the function. The %INCLUDE function adds the smoothed weight of evidence to the training data set.

```
filename brswoe "&PMLRfolder\swoe_branch.sas";

data  null ;
   file brswoe
   set work.counts end=last;
   logit=log((events + &rho1*24)/(_FREQ_ - events + (1-&rho1)*24));
   if  n =1 then put "select (branch);" ;
   put "  when ('" branch +(-1) "') branch_swoe = " logit ";" ;
   if last then do;
   logit=log(&rho1/(1-&rho1));
   put "  otherwise branch_swoe = " logit ";" / "end;";
   end;
run;
```

A listing of the code generated by the DATA step is shown below.

```
select (branch);
  when ('B1') branch_swoe = -0.533682018 ;
  when ('B10') branch_swoe = -0.366920929 ;
  when ('B11') branch_swoe = -0.366824824 ;
  when ('B12') branch_swoe = -0.540183932 ;
  when ('B13') branch_swoe = -0.393681164 ;
  when ('B14') branch_swoe = -1.376871614 ;
  when ('B15') branch_swoe = -1.188201904 ;
  when ('B16') branch_swoe = -0.91025293 ;
  when ('B17') branch_swoe = -0.661782256 ;
  when ('B18') branch_swoe = -0.548226206 ;
  when ('B19') branch_swoe = -0.477468642 ;
  when ('B2') branch_swoe = -0.713003789 ;
  when ('B3') branch_swoe = -0.477918403 ;
  when ('B4') branch_swoe = -0.518845457 ;
  when ('B5') branch_swoe = -0.450637056 ;
  when ('B6') branch_swoe = -0.552908065 ;
  when ('B7') branch_swoe = -0.719594589 ;
  when ('B8') branch_swoe = -0.468178724 ;
  when ('B9') branch_swoe = -0.551166668 ;
  otherwise branch_swoe = -0.635055959 ;
end;
```

```
data work.train_imputed_swoe;
   set work.train imputed;
   %include brswoe / source2;
run;
```

End of Demonstration

Exercises

2. Clustering Categorical Input Levels

a. Include the program **pmlr03e01.sas** or use the **PMLR_UpToDate** macro. Use Greenacre's correspondence analysis to cluster levels of **cluster_code**. Use PROC MEANS to generate a data set with information about the average response rate and sample size for each level of **cluster_code**.

b. Use PROC CLUSTER to group those levels and create a horizontal dendrogram. Use the log of the *p*-value of the appropriate χ^2 test to determine which level of clustering is suitable.

c. Use PROC SGPLOT to plot the log of the *p*-value by the number of clusters. (The range of the Y axis should be -40 to 0.) Use PROC TREE to create a SAS data set of the final results.

d. Use the DATA step to create a file with the assignments for **cluster_code**. Define a new variable called **cluster_clus** and create a new data set called **pmlr.pva_train_imputed_clus** with the new assignments for **cluster_code**.

What is the optimum number of clusters for the **cluster_code** variable?

3. Smoothed Weight of Evidence

a. Include the program **pmlr03e01.sas** or use the **PMLR_UpToDate** macro. Compute the smoothed weight of evidence for **cluster_code**. Use PROC SQL to compute the proportion of events in the training data set.

b. Use PROC MEANS to calculate the response rate and frequency in each of the levels of **cluster_code** and a DATA step to compute the smoothed weight of evidence for each **cluster_code** level.

c. Use a value of *24* for *c* and assign the overall logit to any observation with an undefined **cluster_code**.

d. Define the new variable **cluster_swoe** and put the new assignments in a data set called **pmlr.pva_train_imputed_swoe**.

What is the value of the smoothed weight of evidence for **cluster_code 01**?

End of Exercises

3.04 Multiple Choice Poll

What is the optimum number of clusters for the **cluster_code** variable?

a. 3
b. 4
c. 5
d. 6

37

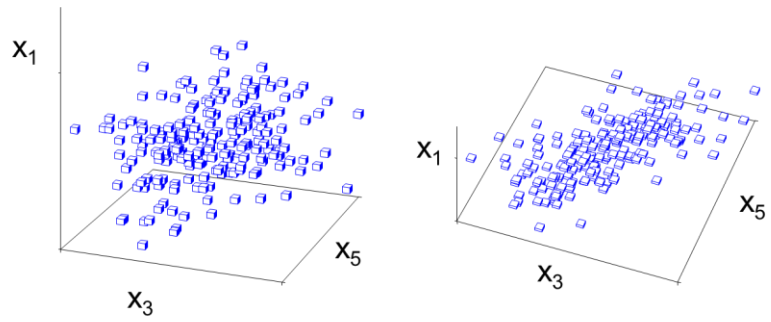
3.3 Variable Clustering

Objectives

- Describe the problem of redundant variables.
- Explain the concept of principal component analysis.
- Use the VARCLUS procedure to demonstrate variable clustering.

41

Redundancy



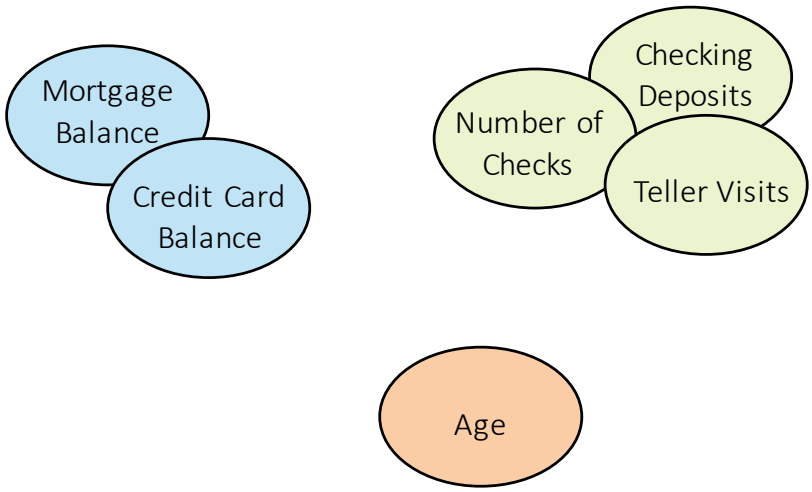
42

Including redundant inputs can degrade the analysis by any of the following conditions:

- destabilizing the parameter estimates
- increasing the risk of overfitting
- confounding interpretation
- increasing computation time
- increasing scoring effort
- increasing the cost of data collection and augmentation

Redundancy is an *unsupervised* concept. It does not involve the target variable. On the other hand, the *relevancy* of a variable considers the relationship between an input variable and the target variable. In high-dimensional data sets, identifying irrelevant inputs is more difficult than identifying redundant inputs. A good strategy is to first reduce redundancy and then tackle irrelevancy in a lower dimension space.

Variable Clustering



43

One approach to variable reduction is variable clustering. *Variable clustering* finds groups of variables that are as correlated as possible among themselves and as uncorrelated as possible with variables in other clusters. A common strategy is to choose one variable from each cluster based on subject-matter knowledge.

Variable Clustering Example

	X_1	X_2	X_3	X_4	X_5
X_1	1	−.11	−.03	−.69	−.04
X_2	−.11	1	−.14	.07	.04
X_3	−.03	−.14	1	.04	−.73
X_4	−.69	.07	.04	1	.02
X_5	−.04	.04	−.73	.02	1

Correlation Matrix

44

In the example, it seems that X_1 is correlated with X_4, and X_3 is correlated with X_5. The variable X_2 is correlated with no other variable. Therefore, there should be three variable clusters.

Principal Components

$$PC_{(1)} = W_{(1)1}X_1 + W_{(1)2}X_2 + \ldots + W_{(1)p}X_p$$
$$PC_{(2)} = W_{(2)1}X_1 + W_{(2)2}X_2 + \ldots + W_{(2)p}X_p$$
$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots \qquad\qquad \vdots$$
$$PC_{(p)} = W_{(p)1}X_1 + W_{(p)2}X_2 + \ldots + W_{(p)p}X_p$$

where the weights were chosen to maximize the quantity

$$\frac{\text{Variance of PC}}{\text{Total Variance}}$$

and the correlation $\text{corr}(PC_{(i)}, PC_{(j)})=0$ for each i is not equal to j.

45

Variable clustering is based on principal components. *Principal components* are weighted linear combinations of the predictor variables where the weights are chosen to account for the largest amount of variation in the data. *Total variation*, in this case, is the sum of the sample variances of the predictor variables. The principal components are numbered according to how much variation in the data is accounted for. (The first principal component accounts for the largest, the second principal component accounts for the second largest, and so on.) Each principal component accounts for a unique portion of the variation in the data. In other words, they are not correlated.

To obtain a unique solution for the principal components, a constraint is imposed where the sum of the squared weights must equal one for each principal component.

Principal Components

	X_1	X_2	X_3	X_4	X_5	PC_1	PC_2	PC_3	PC_4	PC_5	
X_1	1	−.11	−.03	−.69	−.04	1.8	0	0	0	0	PC_1
X_2	−.11	1	−.14	.07	.04	0	1.7	0	0	0	PC_2
X_3	−.03	−.14	1	.04	−.73	0	0	1.0	0	0	PC_3
X_4	−.69	.07	.04	1	.02	0	0	0	0.3	0	PC_4
X_5	−.04	.04	−.73	.02	1	0	0	0	0	0.2	PC_5

Correlation Matrix

Covariance Matrix

46

The correlation matrix is the covariance matrix of the standardized variables. Because each standardized variable has a variance equal to one, the total variability among the standardized variables is only the number of variables. The principal components are produced by an eigen-decomposition of the correlation matrix. The *eigenvalues* are the variances of the principal components. They sum to the number of variables. The first principal component corresponds to the first eigenvalue and explains the largest proportion of the variability. Each principal component explains a decreasing amount of the total variability. In the above example, the first three principal components explain 90% of the total variability.

Principal components analysis can also be used for reducing redundant dimensions. A set of *k* variables can be transformed into a set of *k* principal components. In practice, dimension reduction is achieved by retaining only the first few principal components if they explain a sufficient proportion of the total variation. The reduced set of principal components might then be used in place of the original variables in the analysis.

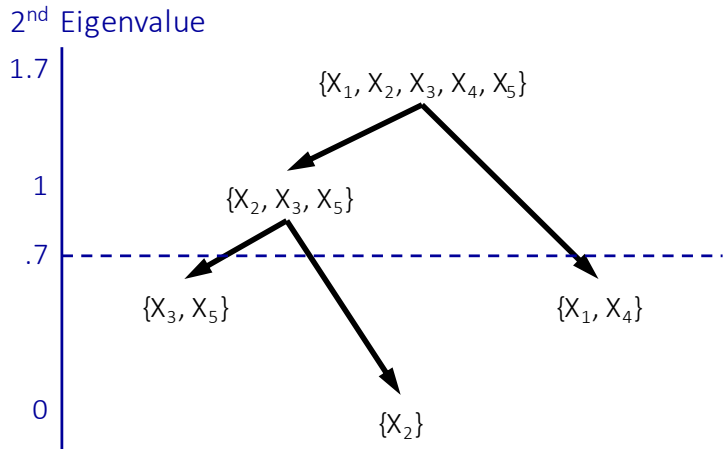
Principal Component Coefficients

	PC_1	PC_2	PC_3	PC_4	PC_5
X_1	−.25	−.65	.09	.69	.15
X_2	.21	.10	.97	.02	.11
X_3	−.65	.28	.04	−.11	.70
X_4	.23	.66	−.14	.70	.07
X_5	.65	−.23	−.19	−.10	.69

47

The chief advantage of variable clustering over principal components is the coefficients. The coefficients of the principal components (*eigenvectors*) are usually nonzero for all the original variables. Thus, even if only a few principal components were used, all the inputs must still be retained in the analysis.

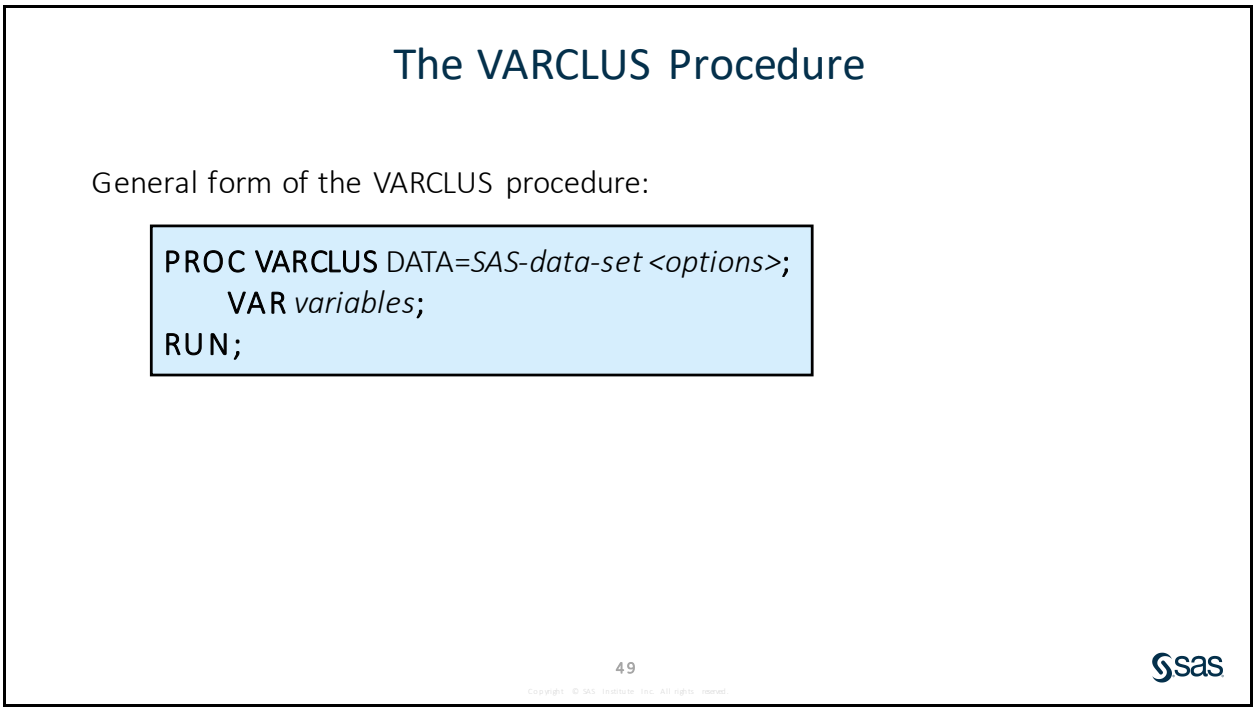
Divisive Clustering



48

Variable clustering finds groups of variables that are as correlated as possible among themselves and as uncorrelated as possible with variables in other clusters. The basic algorithm is binary and divisive. All variables start in one cluster. A principal components analysis is done on the variables in the cluster. If the second eigenvalue is greater than a specified threshold (in other words, more than one dominant dimension), then the cluster is split. The principal component scores are then rotated obliquely so that the variables can be split into two groups. This process is repeated for the two child clusters until the second eigenvalue drops below the threshold. (By default, PROC VARCLUS does a nonhierarchical version of this algorithm where variables can also be reassigned to other clusters.)

Larger thresholds for the second eigenvalue give fewer clusters and less of the variation is explained. Smaller thresholds give more clusters and more of the variation is explained. The value *1* is a common choice for the threshold because it represents the average size of the eigenvalues. To account for sampling variability, smaller values such as *.7* were suggested (Jackson 1991).



Selected PROC VARCLUS statement options:

MAXEIGEN=*n*	specifies the largest permissible value of the second eigenvalue in each cluster. The default is 1 (using the correlation matrix).
SHORT	suppresses printing of the cluster structure, scoring coefficient, and intercluster correlation matrices.

Selected VARCLUS procedure statement:

VAR	specifies the variables to be clustered. If you do not specify the VAR statement, all numeric variables are processed.

Cluster Representatives

$$1-R^2 \text{ ratio} = \frac{1-R^2_{\text{own cluster}}}{1-R^2_{\text{next closest}}}$$

Smaller is better:

$$\frac{1-\uparrow}{1-\downarrow} \Rightarrow \frac{\downarrow}{\uparrow} \Rightarrow \downarrow$$

50

As with principal component analysis, dimension reduction could be achieved by replacing the original variables with the cluster scores (components). A simple alternative is to select a representative variable from each cluster. An ideal representative has high correlation with its own cluster and has a low correlation with the other clusters. Consequently, variables with the lowest $1-R^2$ ratio (defined above) in each cluster are good representatives. Of course, subject-matter considerations might control the selection of other representatives.

Variable Clustering

Example: Cluster the numeric variables in the **work.train_imputed_swoe** data set. These include the original numeric inputs, the missing indicators, and the dummy variables for the collapsed **branch** (64 total). Use the HI option so that clusters at different levels maintain a hierarchical structure that prevents variables from transferring from one cluster to another after the split is made. Use the Output Delivery System to print the table of the R-square statistics from the last iteration of PROC VARCLUS. Also print the table that shows the proportion of variation explained by the clusters.

The ODS OUTPUT statement creates a SAS data set named **summary** from the output object **clusterquality**, and a SAS data set named **clusters** from each **RSquare** output object. Because there are several **RSquare** objects created (one for the two-cluster solution, one for the three-cluster solution, and so on, up to the final cluster solution), the **clusters** data set concatenates these objects. There is a column called **NumberOfClusters** that indicates to which cluster solution each observation in the clusters data set belongs.

```
/* pmlr03d04.sas */
ods select none;
ods output clusterquality=work.summary
           rsquare=work.clusters;

proc varclus data=work.train_imputed_swoe maxeigen=.7 hi;
   var &inputs branch swoe miacctag
       miphone mipos miposamt miinv
       miinvbal micc miccbal miccpurc
       miincome mihmown milores mihmval
       miage micrscor;
run;
ods select all;
```

The CALL SYMPUT routine creates the macro variable **nvar**. It contains the value of the number of clusters in the last iteration of the clustering algorithm. The COMPRESS function strips blanks from variables. Because the **clusters** data set contains the results of all the cluster solutions, the **nvar** macro variable is used to restrict focus to the final result of the VARCLUS algorithm. The **nvar** macro variable is useful later, because if you select one representative from each variable cluster, you have **nvar** inputs for future modeling consideration.

```
%global nvar;
data  null ;
   set work.summary;
   call symput('nvar',compress(NumberOfClusters));
run;

title1 "Variables by Cluster";
proc print data=work.clusters noobs label split='*';
   where NumberOfClusters=&nvar;
   var Cluster Variable RSquareRatio VariableLabel;
   label RSquareRatio="1 - RSquare*Ratio";
run;
```

Variables by Cluster

Cluster	Variable	1 - RSquare Ratio	Variable Label
Cluster 1	branch_swoe	0.4189	
	MIPhone	0.0042	
	MIPOS	0.0042	
	MIPOSAmt	0.0042	
	MIInv	0.0042	
	MIInvBal	0.0042	
	MICC	0.0042	
	MICCBal	0.0042	
	MICCPurc	0.0042	
Cluster 2	MIIncome	0.0074	
	MIHMOwn	0.0446	
	MILORes	0.0074	
	MIHMVal	0.0074	
	MIAge	0.0817	
Cluster 3	Dep	0.4122	Checking Deposits
	Checks	0.3536	Number of Checks
	Teller	0.5254	Teller Visits
Cluster 4	MTGBal	0.0344	Mortgage Balance
	CCBal	0.0322	Credit Card Balance
Cluster 5	MM	0.0926	Money Market
	MMBal	0.1068	Money Market Balance
	MMCred	0.4543	Money Market Credits
Cluster 6	Income	0.1662	Income
	HMVal	0.2450	Home Value
Cluster 7	ILS	0.0046	Installment Loan
	ILSBal	0.0046	Loan Balance
Cluster 8	POS	0.0841	Number Point of Sale
	POSAmt	0.0832	Amount Point of Sale
Cluster 9	NSF	0.2503	Number Insufficient Fund
	NSFAmt	0.2482	Amount NSF
Cluster 10	CD	0.2852	Certificate of Deposit
	CDBal	0.3141	CD Balance
Cluster 11	DDA	0.4846	Checking Account
	DirDep	0.3817	Direct Deposit
Cluster 12	LOC	0.2833	Line of Credit
	LOCBal	0.2951	Line of Credit Balance
Cluster 13	Age	0.0000	Age
Cluster 14	Inv	0.0000	Investment
Cluster 15	InArea	0.0000	Local Address
Cluster 16	AcctAge	0.0000	Age of Oldest Account
Cluster 17	Moved	0.0000	Recent Address Change
Cluster 18	CRScore	0.0000	Credit Score
Cluster 19	MICRScor	0.0000	
Cluster 20	IRABal	0.0000	IRA Balance
Cluster 21	MIAcctAg	0.0000	
Cluster 22	SavBal	0.0000	Saving Balance
Cluster 23	CashBk	0.0000	Number Cash Back
Cluster 24	DDABal	0.0000	Checking Balance
Cluster 25	SDB	0.0000	Safety Deposit Box
Cluster 26	InvBal	0.0000	Investment Balance
Cluster 27	CCPurc	0.0000	Credit Card Purchases
Cluster 28	ATMAmt	0.0000	ATM Withdrawal Amount
Cluster 29	Sav	0.0000	Saving Account

Cluster 30	CC	0.0000	Credit Card
Cluster 31	Phone	0.0000	Number Telephone Banking
Cluster 32	HMOwn	0.0000	Owns Home
Cluster 33	DepAmt	0.0000	Amount Deposited
Cluster 34	IRA	0.0000	Retirement Account
Cluster 35	MTG	0.0000	Mortgage
Cluster 36	ATM	0.0000	ATM
Cluster 37	LORes	0.0000	Length of Residence

The results show which variable is assigned to a specific cluster along with the 1-R-square ratio.

```
title1 "Variation Explained by Clusters";
proc print data=work.summary label;
run;
```

Number of Clusters	Total Variation Explained by Clusters	Proportion of Variation Explained by Clusters	Minimum Proportion Explained by a Cluster	Maximum Second Eigenvalue in a Cluster	Minimum R-squared for a Variable	Maximum 1-R**2 Ratio for a Variable
1	8.839653	0.1449	0.1449	5.021094	0.0000	
2	13.846715	0.2270	0.1956	3.457352	0.0000	1.0000
3	17.207611	0.2821	0.1373	2.625736	0.0000	1.4229
4	19.690396	0.3228	0.1373	2.314577	0.0001	1.4229
5	21.919904	0.3593	0.2239	2.059159	0.0001	1.3331
6	23.915604	0.3921	0.2239	1.965075	0.0003	1.2875
7	25.812779	0.4232	0.2239	1.607659	0.0003	1.2875
8	27.298020	0.4475	0.2239	1.476805	0.0003	1.3890
9	28.676857	0.4701	0.2239	1.410293	0.0003	1.4518
10	30.055834	0.4927	0.2211	1.383226	0.0003	1.4518
11	31.348901	0.5139	0.2211	1.262340	0.0003	1.0995
12	32.596585	0.5344	0.2211	1.261671	0.0003	1.0995
13	33.783565	0.5538	0.2211	1.245679	0.0003	1.1286
14	35.008806	0.5739	0.3093	1.201420	0.0003	1.1286
15	36.073987	0.5914	0.3259	1.008973	0.0003	1.1286
16	37.054640	0.6075	0.3616	1.002464	0.0003	1.1286
17	38.053599	0.6238	0.3616	1.001149	0.0003	1.1286
18	39.053168	0.6402	0.3616	1.000314	0.0003	1.1068
19	40.042180	0.6564	0.3616	0.999772	0.0003	1.1024
20	41.041900	0.6728	0.3616	0.999062	0.0016	1.1024
21	42.040962	0.6892	0.3616	0.996365	0.0371	1.1024
22	43.031772	0.7054	0.3616	0.986328	0.0371	1.1024
23	44.016425	0.7216	0.3616	0.979084	0.0813	1.1024
24	44.789888	0.7343	0.3616	0.970850	0.0813	1.1024
25	45.759249	0.7502	0.3616	0.966296	0.0813	1.1024
26	46.696723	0.7655	0.3616	0.956958	0.0813	1.1024
27	47.653680	0.7812	0.3616	0.947470	0.1320	1.1024
28	48.553991	0.7960	0.4489	0.921793	0.1320	1.1024
29	49.463897	0.8109	0.5234	0.882095	0.1320	1.1024
30	50.345236	0.8253	0.5234	0.860653	0.1896	1.1024
31	51.205517	0.8394	0.5234	0.843103	0.1896	1.1024
32	52.047720	0.8532	0.5234	0.841695	0.2686	0.8516
33	52.886839	0.8670	0.6108	0.778443	0.3604	0.6743
34	53.665282	0.8798	0.6283	0.770747	0.3604	0.6743
35	54.432136	0.8923	0.6283	0.743344	0.4973	0.5930
36	55.175480	0.9045	0.6326	0.734767	0.4973	0.5930
37	55.910247	0.9166	0.6442	0.693155	0.4973	0.5254

The **summary** data set contains the number of clusters for the final iteration. The table also has information about the proportion of variation that is explained by the clusters and the maximum second eigenvalue in a cluster. This information can be used to decide whether too few or too many clusters were formed.

To reduce the amount of text that needs to be entered, a new macro variable named **Reduced** is created. It contains the names of all the variables selected from PROC VARCLUS. As was mentioned above, subject-matter considerations should be involved in deciding which inputs become the cluster representatives. Consider the first cluster, which consists of the variables **branch_swoe**, **MIPhone**, **MIPOS**, **MIPOSAmt**, **MIInv**, **MIInvBal**, **MICC**, **MICCBal**, and **MICCPurc**. According to the 1-R-square ratio, any of the missing indicators would be the best cluster representative. However, closer investigation of the development data set might reveal that the missingness of those eight inputs has more to do with poor data quality than any other source of missing data. If these missing indicators actually imply poor data quality, then their usefulness as predictors with good generalizing power is suspect, especially if the pattern of missingness that occurs in the development data is unlikely to exist in the scoring population. With that subject-matter consideration, perhaps **branch_swoe** is the best cluster representative from cluster 1.

```
%global reduced;
%let reduced= branch swoe MIINCOME Dep CCBal MM Income ILS POS NSF CD
              DDA LOC Age Inv InArea AcctAge Moved CRScore MICRScor
              IRABal MIAcctAg SavBal CashBk DDABal SDB InvBal CCPurc
              ATMAmt Sav CC Phone HMOwn DepAmt IRA MTG ATM LORes;
```

End of Demonstration

Exercises

4. Variable Clustering

a. Include the program **pmlr03e03.sas** or use the **PMLR_UpToDate** macro. Use PROC VARCLUS to cluster all numeric variables in a hierarchical structure. (Use the macro variable **ex_inputs** with the missing indicator variables and the smoothed weight of evidence variable.)

b. Use MAXEIGEN=.70 as your stopping criterion.

c. Use the Output Delivery System to print the table of the R-square statistics from the last iteration of PROC VARCLUS.

d. Print the table that shows the proportion of variation explained by the clusters.

1) How many clusters were in your final solution?

2) What proportion of the variation was explained by the clusters?

End of Exercises

3.05 Multiple Choice Poll

How many clusters were in the final solution?

a. 28
b. 30
c. 32
d. 35

54

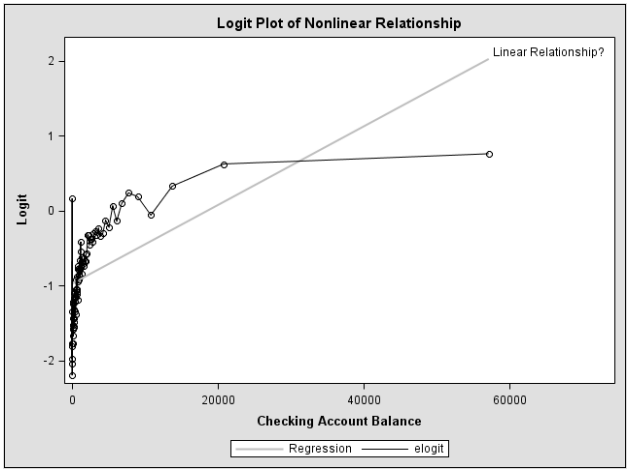
3.4 Variable Screening

Objectives

- Use the Spearman and Hoeffding correlation statistics to illustrate variable screening.
- Create empirical logit plots.
- Demonstrate a method to accommodate nonlinear relationships between the predictor variables and the target variable.

58

Detecting Nonlinear Relationships



59

As was stated earlier in the course, logistic regression was developed for inputs with effects that have a constant rate of change. When this linear relationship is violated, the predictive accuracy of the model might decrease. One way to detect nonlinear relationships is to plot the empirical logits by the input variable values. However, when there are many input variables, the creation of many plots might be cumbersome.

Rank of Spearman and Hoeffding Correlations

Spearman Rank	Hoeffding Rank	Association
High	High	Monotonic
Low	High	Non-monotonic
High	Low	Monotonic
Low	Low	Weak

60

Another way to detect nonlinear relationships is to compare the ranks of the Spearman correlation statistic with the ranks of the Hoeffding's D statistic. The Spearman correlation statistic in this example is a correlation of the ranks of the input variables with the binary target. The Spearman correlation statistic is used rather than the Pearson correlation statistic because Spearman is less sensitive to nonlinearities and outliers. However, when variables are not monotonically related to each other, the Spearman correlation statistic can miss important associations.

A general and robust similarity measure is Hoeffding's D. It can detect a wide variety of associations between two variables. Hoeffding's D statistic has values between –0.5 and 1, but if there are many ties, smaller values might result.

An added benefit of comparing the ranks of the Spearman correlation statistics with the ranks of the Hoeffding's D statistic is to eliminate clearly irrelevant variables. Even after variable clustering, some further variable reduction might be needed before you use the variable selection techniques in PROC LOGISTIC. Very liberal univariate screening might be helpful when the number of clusters created in PROC VARCLUS is still relatively large. Because some of the variable selection techniques use the full model, eliminating clearly irrelevant variables (for example, *p*-values greater than .50) stabilizes the full model and might improve the variable selection technique without much risk of eliminating important input variables.

Variable Screening

Example: Use the CORR procedure to examine the associations between the inputs chosen from PROC VARCLUS and the target variable. Use the SPEARMAN option to request the Spearman correlation statistic and the HOEFFDING option to request Hoeffding's D statistic. Then create a table that compares the rank order of the Spearman correlation statistic to the rank order of the Hoeffding's D statistic. Finally, create a scatter plot of the ranks of Spearman versus the ranks of Hoeffding's D with PROC SGPLOT. Add reference lines and data labels to the plot.

The output object for the table of Spearman correlation statistics is called SPEARMANCORR, and the output object for the table of Hoeffding's D statistics is called HOEFFDINGCORR.

```
/*  pmlr03d05.sas  */

ods select none;
ods output spearmancorr=work.spearman
           hoeffdingcorr=work.hoeffding;

proc corr data=work.train_imputed_swoe spearman hoeffding;
   var ins;
   with &reduced;
run;
```

The variable names of interest in the SAS data sets **Spearman** and **Hoeffding** are **variable** (the variable name), **ins** (the correlation statistic), and **pins** (the *p*-value). The two data sets are sorted by the variable names and merged by the variable names. The absolute values of the correlation coefficients are created for sorting purposes.

```
ods select all;

proc sort data=work.spearman;
    by variable;
run;

proc sort data=work.hoeffding;
    by variable;
run;

data work.correlations;
    merge work.spearman(rename=(ins=scorr pins=spvalue))
          work.hoeffding(rename=(ins=hcorr pins=hpvalue));
    by variable;
    scorr_abs=abs(scorr);
    hcorr_abs=abs(hcorr);
run;
```

PROC RANK is used to create the ranks of the correlation coefficients. The DESCENDING option reverses the direction of the ranks. Therefore, the highest rank receives a value of 1.

```
proc rank data=work.correlations out=work.correlations1 descending;
    var scorr abs hcorr abs;
    ranks ranksp rankho;
run;
```

The final data set is sorted by the rank of the Spearman correlation statistic. Then a table is generated and it shows the rank and associated statistics of the Spearman correlations and Hoeffding's D statistics.

```
proc sort data=work.correlations1;
   by ranksp;
run;

title1 "Rank of Spearman Correlations and Hoeffding Correlations";
proc print data=work.correlations1 label split='*';
   var variable ranksp rankho scorr spvalue hcorr hpvalue;
   label ranksp='Spearman rank*of variables'
         scorr='Spearman Correlation'
         spvalue='Spearman p-value'
         rankho='Hoeffding rank*of variables'
         hcorr='Hoeffding Correlation'
         hpvalue='Hoeffding p-value';
run;
```

Rank of Spearman Correlations and Hoeffding Correlations

Obs	Variable	Spearman rank of variables	Hoeffding rank of variables	Spearman Correlation	Spearman p-value	Hoeffding Correlation	Hoeffding p-value
1	SavBal	1	1	0.25090	<.0001	0.00981	<.0001
2	CD	2	7	0.20283	<.0001	0.00186	<.0001
3	DDA	3	5	-0.19512	<.0001	0.00237	<.0001
4	MM	4	12	0.15949	<.0001	0.00103	<.0001
5	Dep	5	2	-0.15414	<.0001	0.00362	<.0001
6	Sav	6	4	0.15154	<.0001	0.00238	<.0001
7	CC	7	6	0.14636	<.0001	0.00216	<.0001
8	ATM	8	8	-0.12290	<.0001	0.00147	<.0001
9	IRA	9	17	0.11230	<.0001	0.00020	0.0001
10	IRABal	10	18	0.11122	<.0001	0.00018	0.0002
11	Phone	11	14	-0.10133	<.0001	0.00056	<.0001
12	Inv	12	31	0.09852	<.0001	0.00004	0.0583
13	branch_swoe	13	10	0.09398	<.0001	0.00142	<.0001
14	CCPurc	14	16	0.08323	<.0001	0.00023	<.0001
15	POS	15	15	-0.07757	<.0001	0.00041	<.0001
16	SDB	16	19	0.07477	<.0001	0.00017	0.0004
17	CCBal	17	13	0.07023	<.0001	0.00061	<.0001
18	ATMAmt	18	9	-0.06973	<.0001	0.00142	<.0001
19	InvBal	19	32	0.06828	<.0001	-0.00004	1.0000
20	NSF	20	20	-0.06648	<.0001	0.00009	0.0079
21	InArea	21	37	-0.06166	<.0001	-0.00000	0.4178
22	DepAmt	22	11	-0.04875	<.0001	0.00133	<.0001
23	DDABal	23	3	0.04818	<.0001	0.00299	<.0001
24	CashBk	24	28	-0.04601	<.0001	-0.00005	1.0000
25	ILS	25	26	-0.01780	0.0090	-0.00006	1.0000
26	Age	26	36	0.01437	0.0350	-0.00001	0.5353
27	Income	27	34	0.00964	0.1576	-0.00003	1.0000

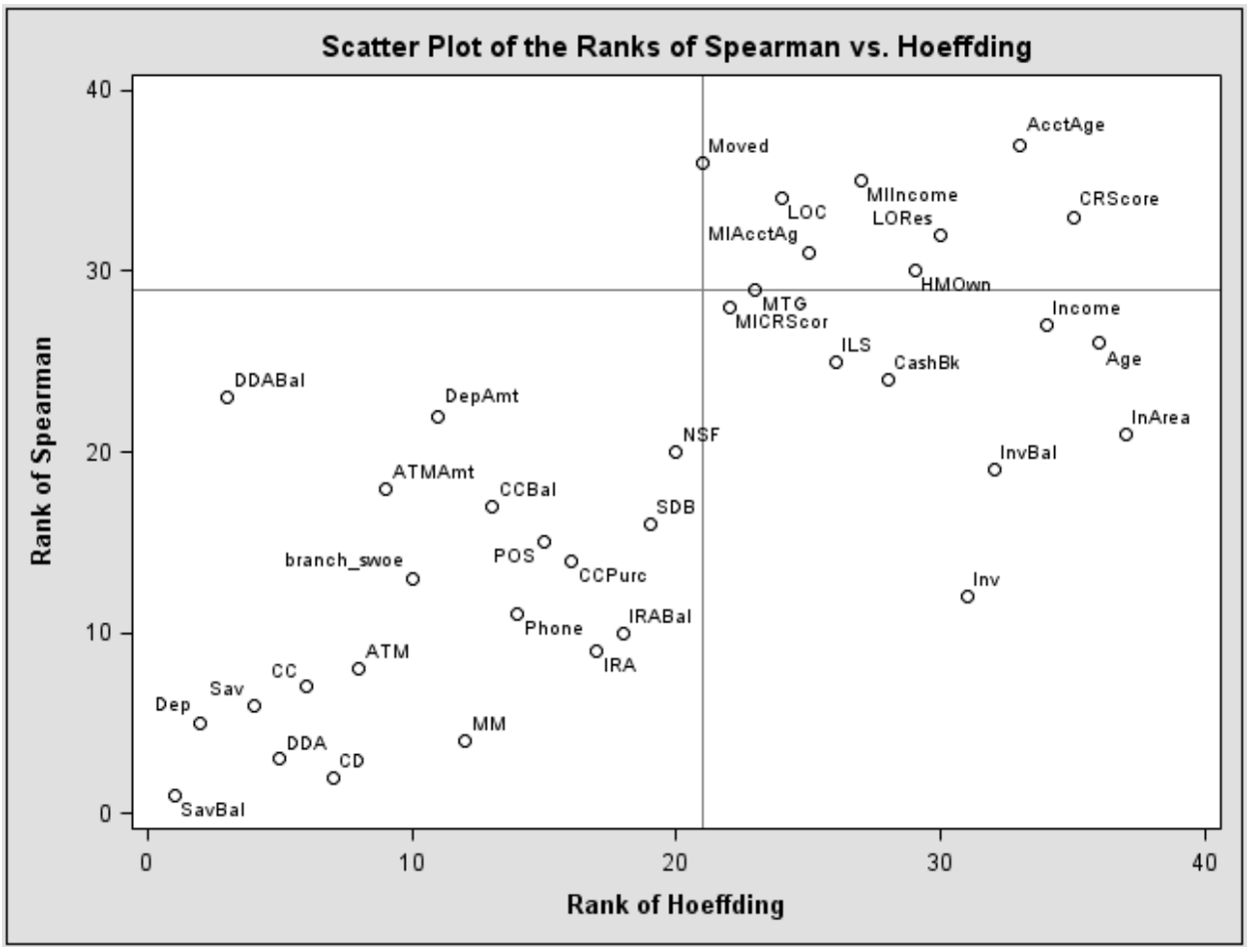
28	MICRScor	28	22	0.00891	0.1915	-0.00007	1.0000
29	MTG	29	23	-0.00428	0.5299	-0.00006	1.0000
30	HMOwn	30	29	-0.00411	0.5470	-0.00005	1.0000
31	MIAcctAg	31	25	0.00323	0.6353	-0.00006	1.0000
32	LORes	32	30	0.00270	0.6925	-0.00004	1.0000
33	CRScore	33	35	0.00152	0.8231	-0.00003	1.0000
34	LOC	34	24	0.00137	0.8405	-0.00006	1.0000
35	MIIncome	35	27	-0.00085	0.9006	-0.00006	1.0000
36	Moved	36	21	0.00069	0.9193	-0.00007	1.0000
37	AcctAge	37	33	-0.00009	0.9895	-0.00003	1.0000

If the Spearman rank is low but the Hoeffding's D rank is high, then the association is probably not monotonic. Empirical logit plots could be used to investigate this type of relationship.

A graphical representation of this table can aid decision making. PROC SGPLOT is used to create a scatter plot of Spearman ranks against Hoeffding ranks. To label the points with the value of a third variable, the DATALABEL= option is used. You can use the REFLINE statement to add reference lines to the appropriate axes.

To have reference lines on the plot, the SQL procedure is used to create macro variables that point to the smallest Spearman rank and Hoeffding rank associated with a *p*-value greater than 0.5. (This is only for presentation.)

```
%global vref href;
proc sql noprint;
   select min(ranksp) into :vref
   from (select ranksp
         from work.correlations1
         having spvalue > .5);

   select min(rankho) into :href
   from (select rankho
         from work.correlations1
         having hpvalue > .5);
quit;

title1 "Scatter Plot of the Ranks of Spearman vs. Hoeffding";
proc sgplot data=work.correlations1;
   refline &vref / axis=y;
   refline &href / axis=x;
   scatter y=ranksp x=rankho / datalabel=variable;
   yaxis label="Rank of Spearman";
   xaxis label="Rank of Hoeffding";
run;
```



In general, the upper right corner of the plot contains the names of variables that can reasonably be excluded from further analysis, due to their poor ranks on both metrics. The criterion to use when you eliminate variables is a subjective decision. Thus, nine variables are eliminated from the analysis.

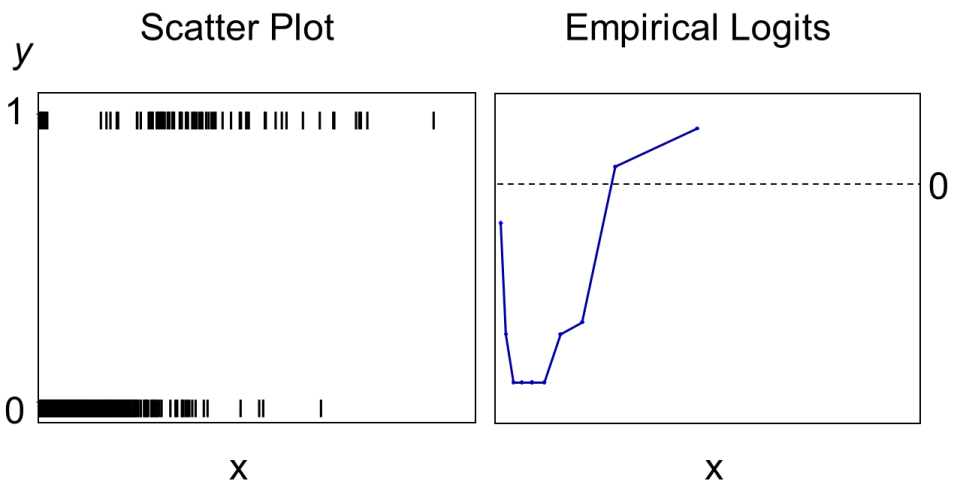
In addition to being a useful screening tool, this analysis might point toward further analyses. Low ranks for Spearman and high ranks for Hoeffding's D are found for the variables **DDABal**, **DepAmt**, and **ATMAmt**. Although these variables do not have a monotonic relationship with **Ins**, some other type of relationship is detected by Hoeffding's D statistic. Empirical logit plots should be used to examine these relationships.

The %LET statement creates a macro variable named **screened**. The macro variable has the names of the variables that remain after the univariate screening method.

```
%global screened;
%let screened = SavBal Dep DDA CD Sav CC ATM MM branch_swoe Phone IRA
                IRABal DDABal ATMAmt ILS POS NSF CCPurc SDB DepAmt
                CCBal Inv InArea Age CashBk MICRScor Income;
```

End of Demonstration

Univariate Smoothing



63

In regression analysis, it is standard practice to examine scatter plots of the target versus each input variable. When the target is binary, these plots are not very enlightening. A useful plot to detect nonlinear relationships is a plot of the empirical logits.

Empirical Logits

$$\ln\left(\frac{m_i + \frac{\sqrt{M_i}}{2}}{M_i - m_i + \frac{\sqrt{M_i}}{2}}\right)$$

where

m_i = number of events

M_i = number of cases

64

Univariate plots of binary data need to be smoothed to better reveal the relationship between a continuous input variable and the target. A simple, scalable, and robust smoothing method is to plot empirical logits for quantiles of the input variables. These logits use a minimax estimate of the proportion of events in each bin (Duffy and Santner 1989). This eliminates the problem that is caused by zero counts and reduces variability.

The number of bins determines the amount of smoothing (for example, the fewer bins, the more smoothing). One large bin would give a constant logit. For very large data sets and intervally scaled inputs, 100 bins often work well. If the standard logistic model were true, then the plots should be linear. Sample variability can cause apparent deviations, particularly when the bin size is too small. However, serious nonlinearities, such as nonmonotonicity, are usually easy to detect.

Empirical Logit Plots

Example: Create an empirical logit plot of checking account balance. Use PROC RANK with the GROUPS= option to bin the input variable. Then use PROC MEANS and a DATA step to generate the values for the logits, and use PROC SGPLOT to create the empirical logit plot.

To create a plot of the empirical logits versus a continuous input variable, the input variable first needs to be binned. In PROC RANK, the bins are an equal size (quantiles) except when the number of tied values exceeds the bin size. In that case, the bin is enlarged to contain all the tied values. The VAR statement in PROC RANK lists the variable in the DATA= data set to be grouped. The RANKS statement names the variable that represents the groups in the OUT= data set. If the RANKS statement is not used, the VAR variable is replaced by the groups.

```
/* pmlr03d06.sas */
%global var;
%let var=DDABal;

proc rank data=work.train_imputed_swoe groups=100 out=work.ranks;
   var &var;
   ranks bin;
run;

title1 "Checking Account Balance by Bin";
proc print data=work.ranks(obs=10);
   var &var bin;
run;
```

Checking Account Balance by Bin

Obs	DDABal	bin
1	1986.81	76
2	1594.84	71
3	1437.57	69
4	190.03	33
5	1772.13	73
6	375.62	42
7	324.94	40
8	13.85	21
9	9644.48	95
10	284.88	38

To compute the empirical logit, the number of target event cases (**Ins**=1) and total cases in each bin needs to be computed. The empirical logits are plotted against the mean of the input variable in each bin. This needs to be computed as well. Both tasks can be done in the MEANS procedure with the CLASS statement. The appropriate statistics (SUM and MEAN) need to be specified in the OUTPUT statement.

```
proc means data=work.ranks noprint nway;
   class bin;
   var ins &var;
   output out=work.bins sum(ins)=ins mean(&var)=&var;
run;
```

```
title1 "Number of Observations, Events, and Average Checking Account "
       "Balance by Bin";
proc print data=work.bins(obs=10);
run;
```

Number of Observations, Events, and Average Checking Account Balance by Bin

Obs	bin	_TYPE_	_FREQ_	ins	DDABal
1	0	1	135	12	-32.2597
2	9	1	3994	2161	0.0000
3	19	1	173	12	2.8347
4	20	1	215	19	8.8542
5	21	1	215	26	17.2156
6	22	1	215	40	28.3862
7	23	1	216	25	41.2978
8	24	1	215	26	53.9779
9	25	1	215	32	68.1660
10	26	1	215	45	83.2786

The variable **Ins** contains the number of events, and the automatic variable **_FREQ_** contains the bin size.

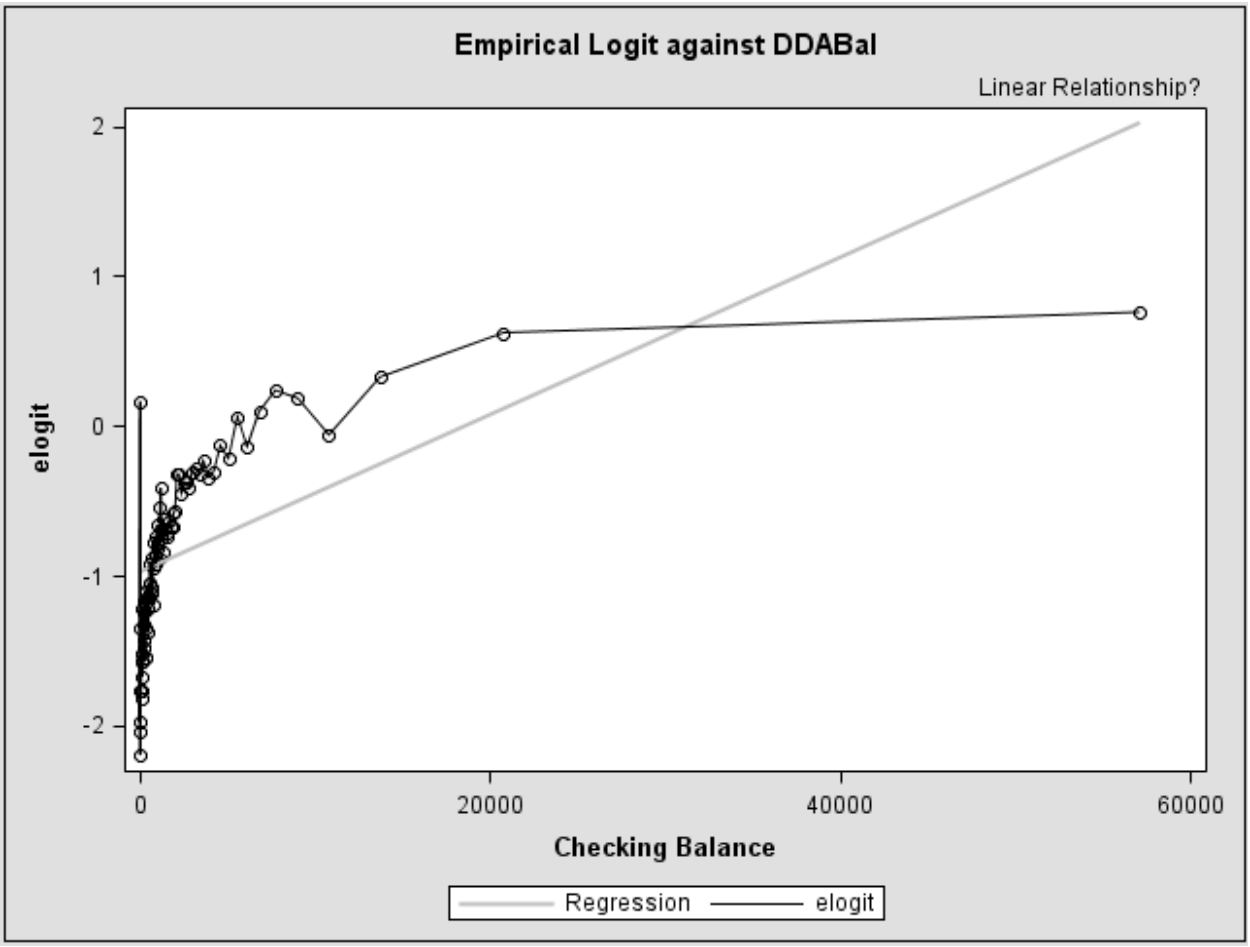
```
data work.bins;
   set work.bins;
   elogit=log((ins+(sqrt(_FREQ_)/2))/
          (_FREQ_-ins+(sqrt(_FREQ_)/2)));
run;
```

PROC SGPLOT enables compatible plots to be placed on top of one another. For example, the SERIES statement connects unmarked points, by default. The REG statement draws a linear regression line through marked data points by default. Although this regression line is not necessarily the same line that results from a logistic regression model on the original (unbinned) data, it might offer a visual cue that suggests whether there is a linear trend in the variable under consideration.

```
title1 "Empirical Logit against &var";
proc sgplot data=work.bins;
   reg y=elogit x=&var /
       curvelabel="Linear Relationship?"
       curvelabelloc=outside
       lineattrs=(color=ligr);
   series y=elogit x=&var;
run;
```

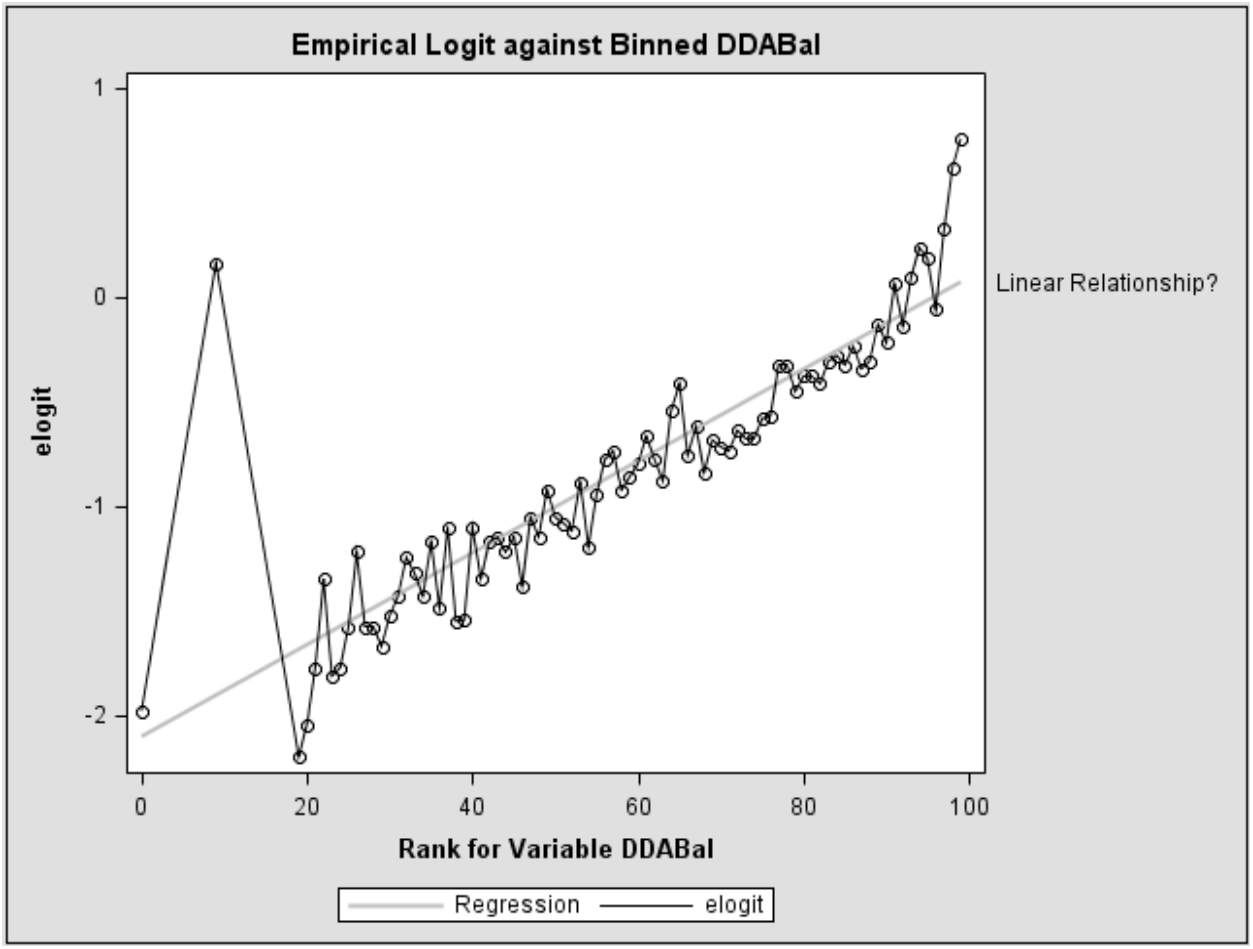
Selected REG statement options:

CURVELABEL	adds a label for the regression curve.
CURVELABELLOC	specifies whether the curve label is placed inside the plot axes (INSIDE) or outside of the plot axes (OUTSIDE).



The pattern made by plotting logit against checking account balance has two striking features. There is a spike in the logits at the $0 balance level. In addition to that spike, the trend is monotonic, but certainly not linear.

```
title1 "Empirical Logit against Binned &var";
proc sgplot data=work.bins;
   reg y=elogit x=bin /
       curvelabel="Linear Relationship?"
       curvelabelloc=outside
       lineattrs=(color=ligr);
   series y=elogit x=bin;
run;
```



The pattern made by plotting logit against the rank of checking account balance seems to accommodate the nonlinearity of the last plot. However, the spike indicates that a portion of the population does not behave as their balances might suggest. The trend in checking account balance is clear, and this is probably a very good input. How, though, can one account for the nonlinear relationship between response behavior and balance amount?

End of Demonstration

Exercises

5. Variable Screening and Logit Plots

a. Include the program **pmlr03e03.sas** or use the **PMLR_UpToDate** macro. Use the Spearman and Hoeffding correlation coefficients to assess the inputs with the least evidence of a relationship with the target.

b. Use the **ex_reduced** macro variable in PROC CORR and use the LENGTH statement in the DATA step to specify a length of *32* for the character variable called **variable**.

c. Create a table with the Spearman rank of inputs and the Hoeffding rank of inputs and use PROC SGPLOT to create a scatter plot of the ranks of Spearman versus the ranks of Hoeffding.

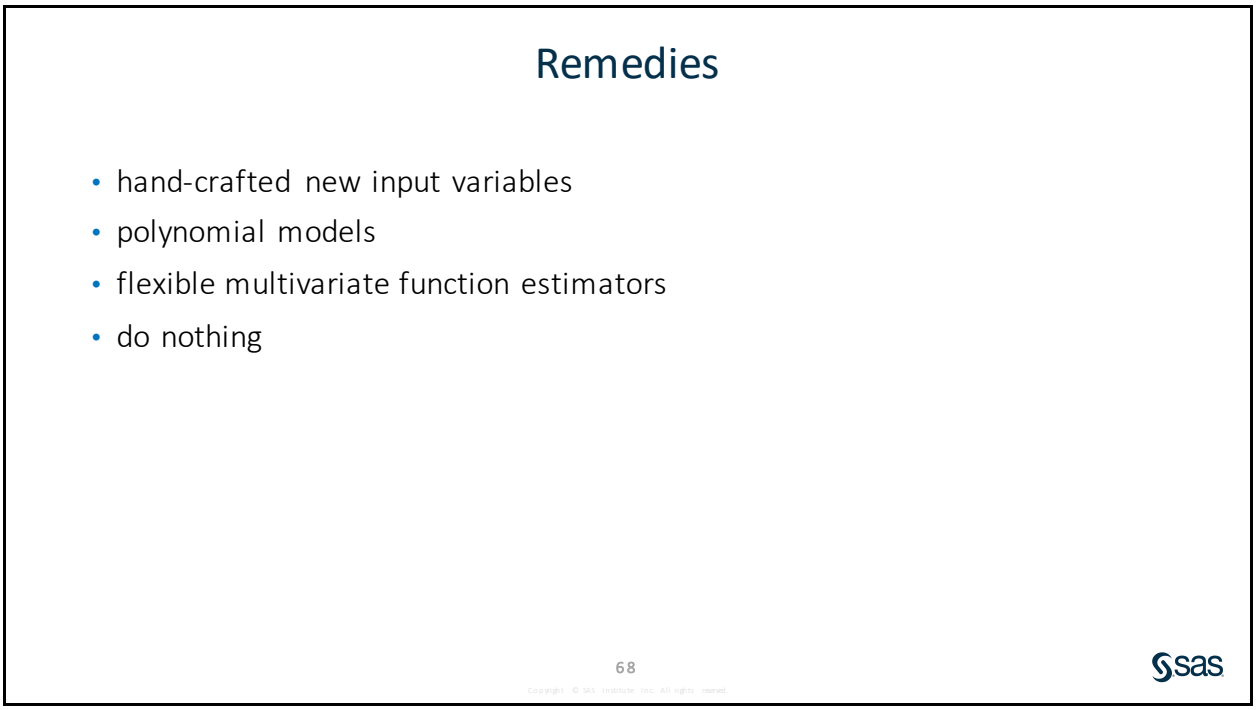
1) Which input variable shows evidence of a nonlinear relationship with the target?

2) Which input variables can be eliminated due to irrelevancy?

d. Create an empirical logit plot of **LAST_GIFT_AMT** and the bins of **LAST_GIFT_AMT**. Use 20 groups in PROC RANK.

What is the relationship between **LAST_GIFT_AMT** and **target_b**?

End of Exercises



- Skilled and patient data analysts can accommodate nonlinearities in a logistic regression model by transforming or discretizing the input variables. This can become impractical with high-dimensional data and increases the risk of overfitting.
- Polynomial terms might improve model fit but hamper model interpretation and generalization. Quadratic terms and two-factor interactions can be useful additions to a modeler's toolkit but are no panacea. Higher-order polynomials are not reliable smoothers.
- Methods such as classification trees, generalized additive models, projection pursuit, multivariate adaptive regression splines, radial basis function networks, and multilayer perceptrons are flexible alternatives to logistic regression (Hastie and Tibshirani 1990, Ripley 1996).
- Standard logistic regression is often robust enough to produce accurate predictions even when there are substantial nonlinear relations between the target and the input variables. Also, more flexible approaches, which might include building in interaction terms, sometimes do not show enough improvement to warrant the extra effort.

Accommodating Nonlinearities

Example: Re-impute checking account balance when the customer does not have a checking account. Then use PROC SGPLOT to create the empirical logit plots.

To use the checking account balance, you might consider taking a logarithmic transformation, a square root transformation, or some other transformation to try to linearize the relationship between the logit and the account balances. This linearization requires two steps. First, the spike at $0 must be evaluated. Second, you can transform the balances to some scale that reflects the behavior exhibited in the data.

It seems suspicious that a large portion of the population has exactly $0 in their checking accounts. Investigation (of the data set or of the people who constructed the data set) shows that most of the individuals with exactly $0 balances do not have checking accounts. Their balances were set to $0 as part of the data pre-processing. This rule seems reasonable from a logical imputation standpoint. How can someone with no checking account have a checking balance? However, it is clear from the logit plots that those individuals behave like people with much more than $0 in their checking accounts.

PROC MEANS with the CLASS statement yields results for each level of the **DDA** variable.
The results of interest are the mean, minimum, and maximum for the **DDABal** and **Ins** variables.

```
/* pmlr03d07.sas */

title1 "Checking Account Balance and INS by Checking Account";
proc means data=work.train_imputed_swoe mean min max;
   class dda;
   var ddabal ins;
run;
```

From the output, it is clear that the individuals without checking accounts had $0 balances imputed for them. Those individuals respond at a higher rate than individuals with checking accounts. If this seems unreasonable, consider that the individuals without checking accounts presumably do their everyday banking with a different bank, and treat this bank as an investment institution.

```
                    Checking Account Balance and INS by Checking Account

    Checking
     Account    N Obs    Variable    Label                        Mean         Median         Minimum
-------------------------------------------------------------------------------------------------------
           0     3968    DDABal      Checking Balance                0              0               0
                         Ins                                 0.5415827      1.0000000               0

           1    17544    DDABal      Checking Balance          2713.45    890.5900000    -774.8300000
                         Ins                                 0.3022116              0               0
-------------------------------------------------------------------------------------------------------


                Checking
                 Account    N Obs    Variable    Label                       Maximum
                ---------------------------------------------------------------------
                       0     3968    DDABal      Checking Balance                  0
                                     Ins                                   1.0000000

                       1    17544    DDABal      Checking Balance         278093.83
                                     Ins                                   1.0000000
                ---------------------------------------------------------------------
```

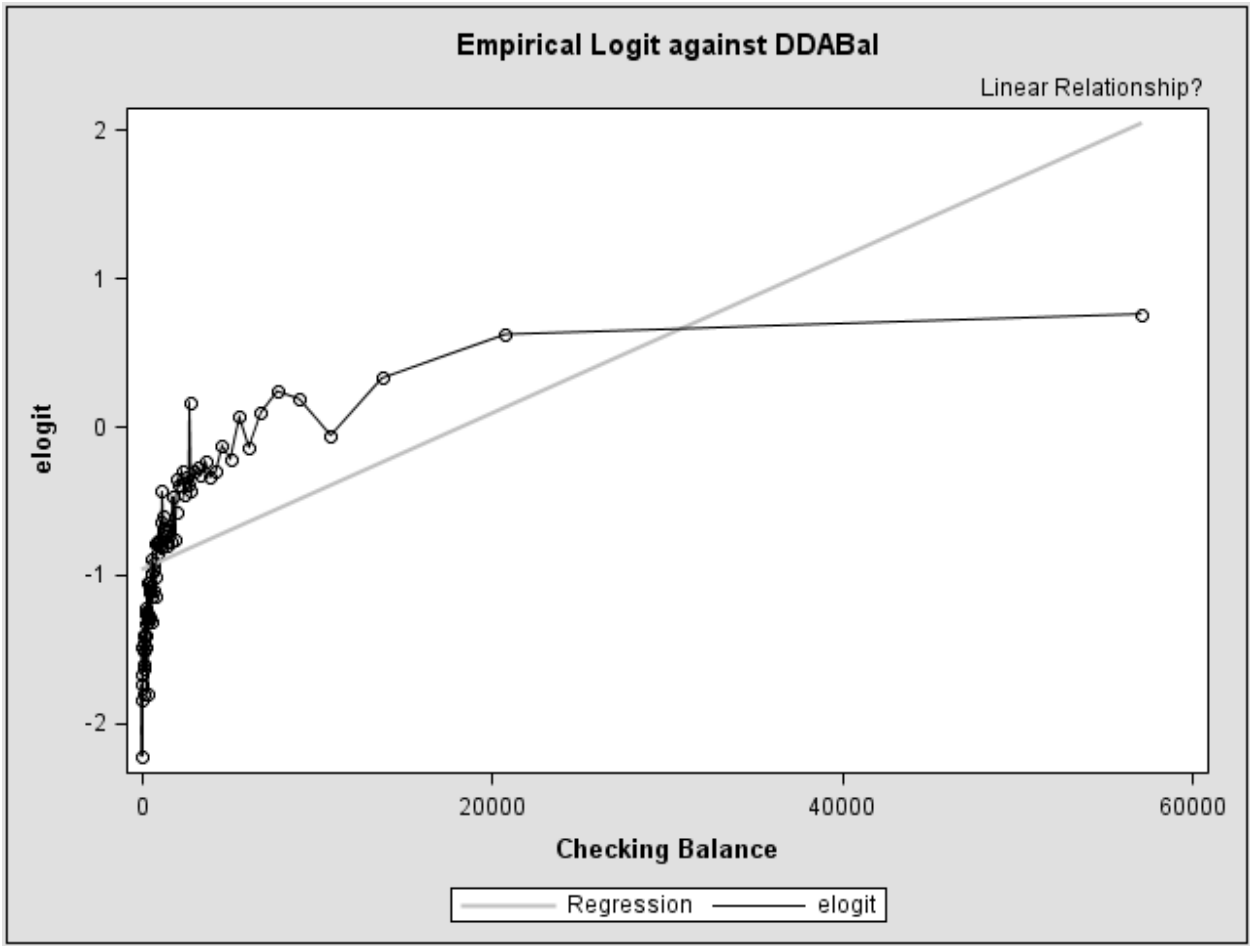
There are several ways to account for this discrepancy between the **DDABal** value and the empirically observed response behavior. One of the most straightforward is to re-impute. The code below creates a macro variable mean that has the value of the checking account balances of those customers who have checking accounts. That mean is substituted for the $0 balances[1] of those who do not have checking accounts. The SELECT INTO statement in PROC SQL creates the macro variable, and the DATA step fills in the $0 balances with the balance of customers who have their accounts at this bank.

```
%global mean;
proc sql noprint;
   select mean(ddabal) into :mean
   from work.train_imputed_swoe where dda;
quit;

data train_imputed_swoe_dda;
   set work.train imputed swoe;
   if not dda then ddabal=&mean;
run;
```

To evaluate the effectiveness of this re-imputation, look again at the empirical logit plots.

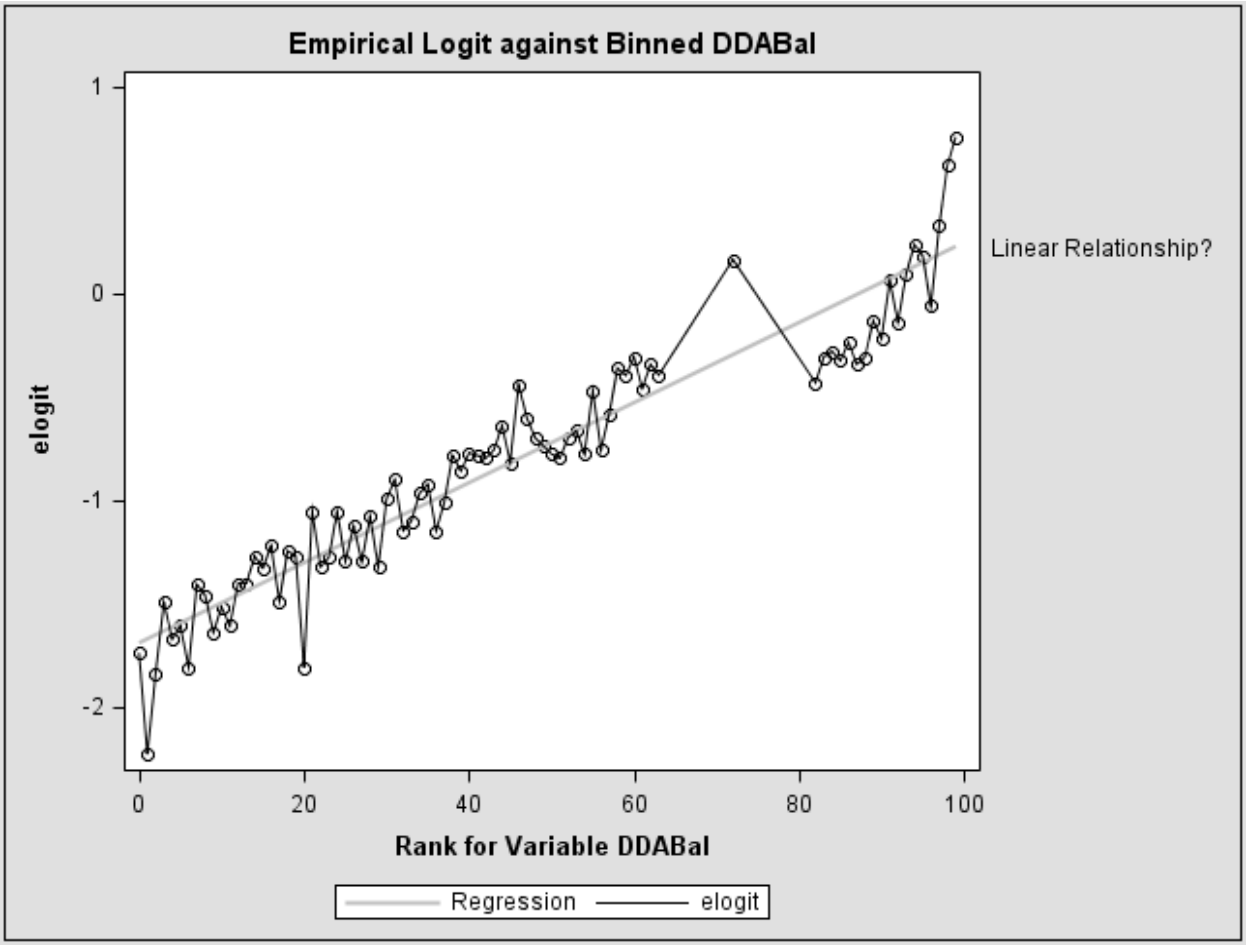
```
%global var;
%let var=DDABal;

proc rank data=work.train_imputed_swoe_dda groups=100 out=work.ranks;
   var &var;
   ranks bin;
run;

proc means data=work.ranks noprint nway;
   class bin;
   var ins &var;
   output out=work.bins sum(ins)=ins mean(&var)=&var;
run;

data work.bins;
   set work.bins;
   elogit=log((ins+(sqrt(_FREQ_ )/2))/
          ( _FREQ_ -ins+(sqrt(_FREQ_ )/2)));
run;

title1 "Empirical Logit against &var";
proc sgplot data=work.bins;
   reg y=elogit x=&var /
       curvelabel="Linear Relationship?"
       curvelabelloc=outside
       lineattrs=(color=ligr);
   series y=elogit x=&var;
run;
```

[1] Arguably, most, if not all, of these customers have checking accounts somewhere, and therefore do have balances.



The empirical logit plot of checking account balances still does not show a linear relationship even when the mean is substituted for the $0 balances for the customers who do not have checking accounts.

```
title1 "Empirical Logit against Binned &var";
proc sgplot data=work.bins;
   reg y=elogit x=bin /
       curvelabel="Linear Relationship?"
       curvelabelloc=outside
       lineattrs=(color=ligr);
   series y=elogit x=bin;
run;
```



The bins do a very good job of capturing how response behavior changes in a linear way with respect to checking account balances.

Note: The question of which transformation does the best job of linearizing the relationship is an important one for modelers. You might try several of your favorite transformations. The limitations are usually only the analyst's imagination and the software on the server that houses the database.

Some analysts might reserve the term *binning* for the practice of creating a categorical predictor from a continuous input. Therefore, it might be more appropriate to label this a *rank-group* or *percentile* transformation.

Regardless of the input's original distribution, the percentiles have a uniform distribution. This uniformly distributed version of **DDABal** has a linear association with the logit. Even better, unlike the log transformation, the percentile transformation affords easy interpretability with odds ratios. A one-unit change in the percentiles results in the corresponding change in the odds of the response.

Example: Create a new input, **B_DDABal**, which represents the binned checking account balances. Because you need the bin assignment rules to create the **B_DDABal** input, if it eventually is part of your model, generate DATA step code to perform the binning.

The RANK procedure assigns observations to 100 groups, according to **DDABal**. PROC MEANS creates a data set that contains the maximum **DDABal** value in each bin. This information can be used to assign a new data set its own bins, without running PROC RANK.

```
proc rank data=work.train_imputed_swoe_dda groups=100 out=work.ranks;
   var ddabal;
   ranks bin;
run;

proc means data=work.ranks noprint nway;
   class bin;
   var ddabal;
   output out=work.endpts max=max;
run;

title1 "Checking Account Balance Endpoints";
proc print data=work.endpts(obs=10);
run;
```

Checking Account Balance Endpoints

Obs	bin	_TYPE_	_FREQ_	max
1	0	1	215	1.75
2	1	1	215	8.45
3	2	1	214	16.44
4	3	1	216	27.54
5	4	1	215	41.08
6	5	1	215	53.01
7	6	1	215	66.42
8	7	1	216	82.96
9	8	1	215	96.55
10	9	1	215	111.59

Rather than writing a series of rules by hand (for example, IF **DDABal**<=1.72 THEN BIN=0; ELSE IF…) for many bins, you can use a DATA step to write the rules. An easy way to do this is to use a FILENAME statement to point to a file and then write the rules to that file with the PUT statement.

The FILENAME statement creates a *fileref*, **Rank**, which points to the S:\workshop\rank.sas physical file. The DATA _NULL_ statement enables you to use the DATA step processing without the requirement to write a data set. The FILE statement specifies where you want to put the output of the DATA step (similar to the INFILE statement that enables you to specify the input source for a DATA step). The SET statement brings in the data set **endpts**, which has the maximum balance in each bin. The END= option creates an internal flag that you can use to identify the last record of the data set. You could write IF-THEN/ELSE syntax to do the bin assignment. The following example uses SELECT-WHEN/OTHERWISE to perform the same task. The last record captures everyone with a balance larger than the maximum in the penultimate bin.

```
filename rank "&PMLRfolder\rank.sas";

data _null_;
   file rank;
   set work.endpts end=last;
   if _n_=1 then put "select;";
   if not last then do;
      put "  when (ddabal <= " max ") B_DDABal=" bin ";";
      end;
   else if last then do;
      put "  otherwise B_DDABal=" bin ";" / "end;";
   end;
run;
```

Partial Listing of **rank.sas**

```
select;
  when (ddabal <= 1.75 ) B_DDABal =0 ;
  when (ddabal <= 8.45 ) B_DDABal =1 ;
  when (ddabal <= 16.44 ) B_DDABal =2 ;
  when (ddabal <= 27.54 ) B_DDABal =3 ;
  when (ddabal <= 41.08 ) B_DDABal =4 ;
  when (ddabal <= 53.01 ) B_DDABal =5 ;
  when (ddabal <= 66.42 ) B_DDABal =6 ;
  when (ddabal <= 82.96 ) B_DDABal =7 ;
  when (ddabal <= 96.55 ) B_DDABal =8 ;
  when (ddabal <= 111.59 ) B_DDABal =9 ;
  when (ddabal <= 126.69 ) B_DDABal =10 ;
```

To use this code, you can use the %INCLUDE statement that is embedded in a DATA step. For example, the following code puts **B_DDABal** on the **imputed** data set. The SOURCE option requests that the code be included in the log as well.

```
data work.train_imputed_swoe_bins;
   set work.train imputed swoe_dda;
   %include rank / source;
run;
```

To see that the recoding worked, evaluate the minimum and maximum checking account balances in each of the bins. PROC MEANS with a CLASS statement can do this.

```
title1 "Minimum and Maximum Checking Account Balance by Bin";
proc means data=train_imputed_swoe_bins min max;
   class B_DDABal;
   var DDABal;
run;
```

Partial Output

Minimum and Maximum Checking Account Balance by Bin

The MEANS Procedure

Analysis Variable : DDABal Checking Balance

B_DDABal	N Obs	Minimum	Maximum
0	215	-774.8300000	1.7500000
1	215	1.8900000	8.4500000
2	214	8.4600000	16.4400000
3	216	16.4600000	27.5400000
4	215	27.6200000	41.0800000
5	215	41.1900000	53.0100000
6	215	53.0200000	66.4200000
7	216	66.5000000	82.9600000
8	215	83.0600000	96.5500000
9	215	96.6200000	111.5900000
10	215	111.6300000	126.6900000

Because the binned balance input is a replacement for the **DDABal** column, exchange **B_DDABal** for **DDABal** in the **screened** macro variable.

```
%global screened;
%let screened = SavBal Dep DDA CD Sav CC ATM MM branch swoe Phone IRA
                IRABal B_DDABal ATMAmt ILS POS NSF CCPurc SDB DepAmt
                CCBal Inv InArea Age CashBk MICRScor Income;
```

Note: Binning is not the only option. Other popular techniques for hand-crafting inputs include the transformations mentioned above as well as splines. Notice that **B_DDABal** is not a categorical input. The point of the linearization exercise is to take advantage of the linear relationship between logits and bins, not to add 100 dummy variables to the list of inputs.

If there were any question about data quality, the binning code must be much more robust.

End of Demonstration

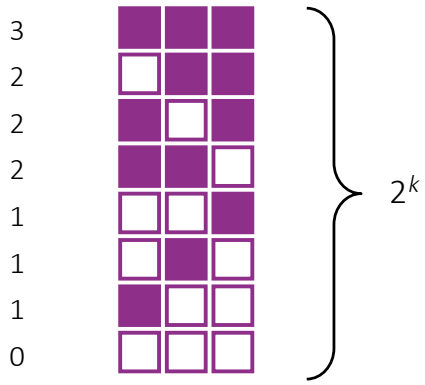
3.5 Subset Selection

Objectives

- Describe the subset selection methods in the LOGISTIC procedure.
- Demonstrate a method that detects interactions.
- Find the subset of variables that minimizes the Schwarz Bayesian information criterion.

72

All Subsets



73

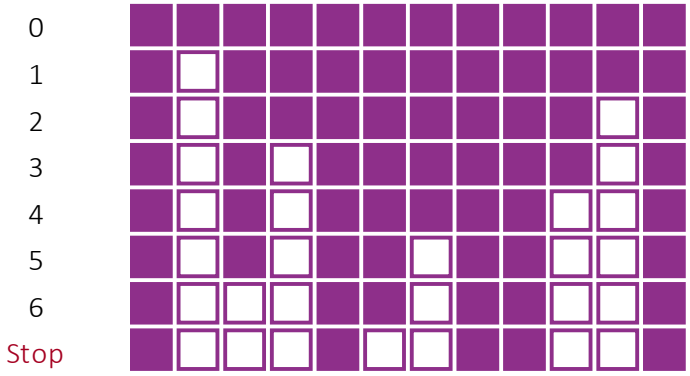
Variable selection methods in regression are concerned with finding subsets of the inputs that are jointly important in predicting the target. The most thorough search considers all possible subsets. This can be prohibitively expensive when the number of inputs, k, is large, as there are 2^k possible subsets to consider.

Stepwise variable selection is an often criticized and yet heavily used subset selection method. Stepwise selection first searches the 1-input models and selects the best. It then searches the 2-input models that contain the input selected in the first step and selects the best. The model is incrementally built in this fashion until no further improvement is made.

There is also a backward portion of the algorithm where, at each step, the variables in the current model can be removed if they become unimportant. The usual criterion that is used for entry and removal from the model is the *p*-value from a significance test in which the coefficient is zero, although other criteria can also be used. In subset selection, the *p*-value is merely a tuning parameter that measures the relative strength of the candidate variables.

Stepwise selection was devised to provide a computationally efficient alternative to examining all subsets. It is not guaranteed to find the best subset and it can be shown to perform badly in many situations (Harrell 1997).

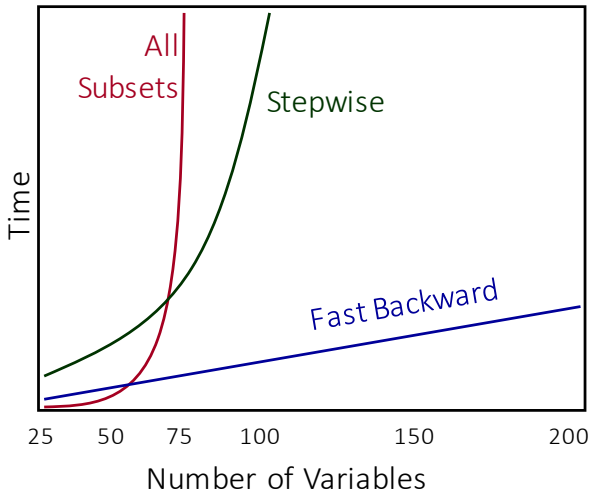
Backward Elimination



75

Backward variable selection starts with all the candidate variables in the model simultaneously. At each step, the least important input variable is removed (as determined by the *p*-value). Backward elimination is less inclined to exclude important inputs or include spurious inputs than forward (stepwise) methods (Mantel 1970; Harrell 1997). However, it is considered more computationally expensive than stepwise because more steps are usually required and they involve larger models.

Scalability in PROC LOGISTIC



76

Most of the literature about the different subset selection methods consider linear rather than logistic regression. The conventional wisdom regarding computation time is that **stepwise < backward < all subsets**.

However, logistic regression (as implemented by PROC LOGISTIC) gives a different story. For up to ≈60 inputs, the results are reversed: **all subsets** < **fast backward** < **stepwise**.

For any number of inputs, backward elimination (with the FAST option) is more efficient than stepwise. (The above simulation was conducted with 50,000 cases and 200 intercorrelated inputs. Sixteen of the inputs were important. Six were strongly so.)

Logistic regression requires an iterative optimization algorithm. Each model fit is much more expensive than with linear regression. Each step in the stepwise algorithm requires iterative optimization.

All-subsets selection is executed in PROC LOGISTIC with the SELECTION=SCORE option (SAS Institute Inc. 2014). This method requires that only one model be fit (the full model). Then, the results are manipulated to calculate a score test for each possible combination of input variables. It also uses a branch-and-bound method for efficiently searching the many combinations. This method is the fastest until the number of possible combinations becomes unmanageable, at which point the performance acutely deteriorates. If redundant inputs are eliminated first (using variable clustering), then all-subsets selection can be a practical method for predictive modeling.

When combined with the FAST option, backward variable selection requires only a single logistic regression (SAS Institute Inc. 2014). PROC LOGISTIC uses the method of Lawless and Singhal (1978) to manipulate the full model fit to approximate fitting reduced models. The FAST option is extremely efficient because the model is not refitted for every variable removed. Fast backward elimination had the best overall performance, that is, a linear increase in time as the number of inputs increased. Notice that ordinary backward (without the FAST option) would be slower than stepwise.

Note: The FAST option provides only approximations to the regression coefficients. If you compare models selected by backward elimination with and without the FAST option, you see different regression coefficients.

3.06 Multiple Choice Poll

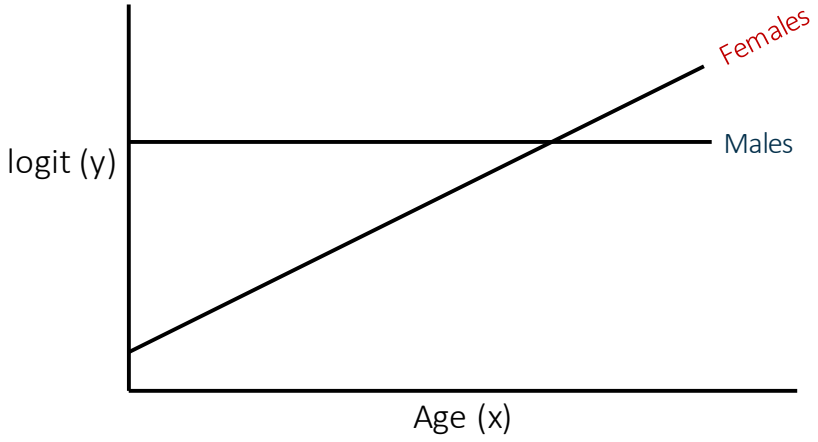
Which of the following statements is ***true*** regarding best subsets selection?

a. The models are ranked by the likelihood ratio chi-square.
b. The method uses a forward selection to find the best model.
c. The method is relatively efficient for a small number of variables (for example, fewer than 50).
d. The FAST option increases the efficiency of the best subsets selection.

77

sas

Detecting Interactions

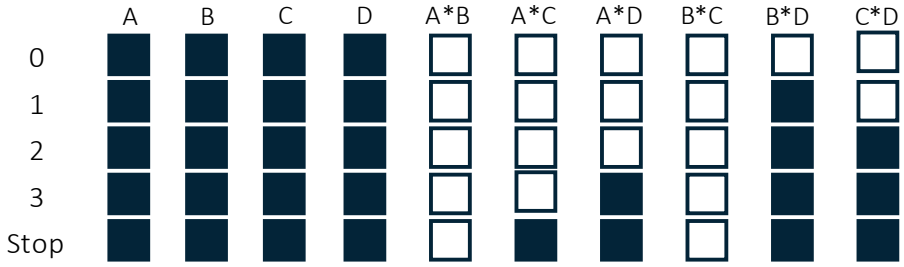


79

sas

Interaction occurs when the relationship between an input variable and the target differs by the level of another input variable. For example, the graph above shows that the relationship between **gender** and the target differs by the value of **age**. Therefore, the **age*gender** interaction should be evaluated in the model. Furthermore, any estimate of the odds ratio for **gender** should be made with respect to a specific **age**.

Interaction Detection – Forward Selection



80

sas

The forward selection method might be useful in detecting interactions. You start with the main effects only model (using the INCLUDE= option) and then let enable the forward selection method to search for any significant interactions. The significance level should be relatively low because you want to include only relatively strong interactions in your final model.

BIC-based Significance Level

$$BIC = -2\ln(likelihood) + (\#\ parameters) * \ln(n)$$

$$BIC^{p+1} < BIC^{p}$$

$$-2\ln(L_{p+1}) + (p+1) * \ln(n) < -2\ln(L_p) + p * \ln(n)$$

$$\ln(n) < 2(\ln(L_{p+1}) - \ln(L_p))$$

Likelihood Ratio Test

$$"\ p - value" < 1 - F_{\chi^2}(\ln(n))$$

81

sas

In subset selection, the significance level controls model complexity. Larger values enable more predictors in the model. The slide shows a derivation of a significance level based on the Bayesian information criterion (BIC). This criterion, also known as the Schwarz Bayesian criterion (SBC), is a measure of fit penalized for model complexity. Selecting the model with the smallest BIC favors a tight fit to the training data (large likelihood) and a small number of parameters.

A BIC-based significance level ensures the reduction of the (approximate) BIC. Reduction in the BIC is ensured if the likelihood ratio test statistic exceeds a cutoff. In the second to last step in the above slide, the right side of the equation is the likelihood ratio test statistic for the null hypothesis that the one additional parameter is zero. The score and Wald chi-square test statistics that are used in subset selection are asymptotically equivalent to the likelihood ratio statistic. They approximately follow a chi-square distribution with one degree of freedom. Therefore, an approximate BIC-based significance level is to accept a predictor for entry or retention, if the *p*-value is less than the tail area of a chi-square PDF with one degree of freedom to the right of ln(*n*). This area is computed with the PROBCHI function. This derivation is based on Potts (2014).

Interaction Detection and Automatic Subset Selection

Example: Compute a BIC-based significance level using the sample size for n. Then use the forward selection method to detect important 2 factor interactions. Use the INCLUDE option to include all the screened variables and **res**.

```
/* pmlr03d08.sas  */

title "P-Value for Entry and Retention";

%global sl;
proc sql;
   select 1-probchi(log(sum(ins ge 0)),1) into :sl
   from work.train_imputed_swoe_bins;
quit;
```

```
                    P-Value for Entry and Retention
                                  ________
                                  0.001586
```

The significance level is much lower than the common 0.05 level.

```
title1 "Interaction Detection using Forward Selection";
proc logistic data=work.train imputed_swoe_bins;
   class res (param=ref ref='S');
   model ins(event='1')= &screened res
            SavBal|Dep|DDA|CD|Sav|CC|ATM|MM|branch_swoe|Phone|IRA|
            IRABal|B DDABal|ATMAmt|ILS|POS|NSF|CCPurc|SDB|DepAmt|
            CCBal|Inv|InArea|Age|CashBk|MICRScor|Income|res @2 /
            include=28 clodds=pl selection=forward slentry=&sl;
run;
```

Selected MODEL statement options:

SELECTION= specifies the method used to select the variables in the model. FORWARD requests forward selection.

SLENTRY= specifies the significance level for entry into the model.

INCLUDE=*n* includes the first *n* predictor variables in the MODEL statement in every model.

The bar notation with @2 constructs a model with all the main effects and the two-factor interactions. If you increased it to @3, then you construct a model with all the main effects, the two-factor interactions, and the three-factor interactions.

Partial Output

Summary of Forward Selection

Step	Effect Entered	DF	Number In	Score Chi-Square	Pr > ChiSq	Variable Label
1	SavBal*B_DDABal	1	29	342.3565	<.0001	
2	SavBal*DDA	1	30	75.6237	<.0001	
3	MM*B_DDABal	1	31	61.2113	<.0001	
4	branch_swoe*ATMAmt	1	32	55.0544	<.0001	
5	Sav*B_DDABal	1	33	46.3236	<.0001	
6	ATMAmt*DepAmt	1	34	36.9443	<.0001	
7	SavBal*SDB	1	35	28.9771	<.0001	
8	SavBal*ATMAmt	1	36	24.4441	<.0001	
9	B_DDABal*ATMAmt	1	37	28.2743	<.0001	
10	SavBal*IRA	1	38	18.1867	<.0001	
11	SavBal*MM	1	39	18.2860	<.0001	
12	SavBal*CC	1	40	17.6232	<.0001	
13	Sav*NSF	1	41	14.3527	0.0002	
14	DDA*ATMAmt	1	42	14.8869	0.0001	
15	Dep*ATM	1	43	14.5252	0.0001	
16	IRA*B_DDABal	1	44	13.9868	0.0002	
17	CD*MM	1	45	12.9265	0.0003	
18	MM*IRABal	1	46	11.9803	0.0005	
19	CD*Sav	1	47	10.8910	0.0010	
20	B_DDABal*CashBk	1	48	10.3711	0.0013	
21	Sav*CC	1	49	10.1994	0.0014	

Twenty-one interactions were selected using the BIC-based significance level.

Type 3 Analysis of Effects

Effect	DF	Wald Chi-Square	Pr > ChiSq
SavBal	1	299.0089	<.0001
Dep	1	3.2378	0.0720
DDA	1	7.9296	0.0049
CD	1	219.7287	<.0001
Sav	1	121.8246	<.0001
CC	1	6.1556	0.0131
ATM	1	0.2451	0.6205
MM	1	142.9057	<.0001
branch_swoe	1	129.4792	<.0001
Phone	1	3.3792	0.0660
IRA	1	29.0627	<.0001
IRABal	1	6.3858	0.0115
B_DDABal	1	724.9335	<.0001
ATMAmt	1	44.4148	<.0001
ILS	1	10.1028	0.0015
POS	1	0.0017	0.9673
NSF	1	48.3251	<.0001
CCPurc	1	0.0717	0.7889
SDB	1	6.7272	0.0095
DepAmt	1	1.8362	0.1754
CCBal	1	4.0781	0.0434
Inv	1	19.8636	<.0001
InArea	1	0.2743	0.6005

Age	1	0.2609	0.6095
CashBk	1	9.7760	0.0018
MICRScor	1	0.1271	0.7215
Income	1	0.8792	0.3484
Res	2	0.3174	0.8532
SavBal*DDA	1	25.4583	<.0001
CD*Sav	1	11.0089	0.0009
SavBal*CC	1	22.8138	<.0001
Sav*CC	1	10.1952	0.0014
Dep*ATM	1	14.8454	0.0001
SavBal*MM	1	17.9087	<.0001
CD*MM	1	17.1153	<.0001
SavBal*IRA	1	14.6885	0.0001
MM*IRABal	1	8.8578	0.0029
SavBal*B_DDABal	1	192.3418	<.0001
Sav*B_DDABal	1	61.5090	<.0001
MM*B_DDABal	1	49.2303	<.0001
IRA*B_DDABal	1	13.7755	0.0002
SavBal*ATMAmt	1	5.5194	0.0188
DDA*ATMAmt	1	16.7930	<.0001
branch_swoe*ATMAmt	1	51.6642	<.0001
B_DDABal*ATMAmt	1	25.4494	<.0001
Sav*NSF	1	12.7568	0.0004
SavBal*SDB	1	11.8839	0.0006
ATMAmt*DepAmt	1	22.9479	<.0001
B_DDABal*CashBk	1	9.8889	0.0017

Several main effects do not seem important because they have high **p**-values.

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-2.0116	0.1406	204.7488	<.0001
SavBal	1	0.000167	9.646E-6	299.0089	<.0001
Dep	1	-0.0457	0.0254	3.2378	0.0720
DDA	1	-0.1794	0.0637	7.9296	0.0049
CD	1	1.1361	0.0766	219.7287	<.0001
Sav	1	1.0100	0.0915	121.8246	<.0001
CC	1	0.1248	0.0503	6.1556	0.0131
ATM	1	0.0314	0.0633	0.2451	0.6205
MM	1	1.7606	0.1473	142.9057	<.0001
branch_swoe	1	0.9275	0.0815	129.4792	<.0001
Phone	1	-0.0345	0.0188	3.3792	0.0660
IRA	1	1.0636	0.1973	29.0627	<.0001
IRABal	1	0.000012	4.85E-6	6.3858	0.0115
B_DDABal	1	0.0282	0.00105	724.9335	<.0001
ATMAmt	1	0.000199	0.000030	44.4148	<.0001
ILS	1	-0.2470	0.0777	10.1028	0.0015
POS	1	0.000278	0.00678	0.0017	0.9673
NSF	1	0.6470	0.0931	48.3251	<.0001
CCPurc	1	-0.0108	0.0404	0.0717	0.7889
SDB	1	0.1381	0.0533	6.7272	0.0095
DepAmt	1	4.286E-6	3.163E-6	1.8362	0.1754
CCBal	1	-5.2E-7	2.576E-7	4.0781	0.0434
Inv	1	0.4398	0.0987	19.8636	<.0001
InArea	1	-0.0435	0.0831	0.2743	0.6005

Age		1	-0.00067	0.00132	0.2609	0.6095
CashBk		1	-0.9778	0.3127	9.7760	0.0018
MICRScor		1	-0.0387	0.1087	0.1271	0.7215
Income		1	0.000607	0.000647	0.8792	0.3484
Res	R	1	0.0235	0.0428	0.3001	0.5838
Res	U	1	0.0139	0.0377	0.1358	0.7125
SavBal*DDA		1	0.000023	4.555E-6	25.4583	<.0001
CD*Sav		1	-0.3162	0.0953	11.0089	0.0009
SavBal*CC		1	-0.00002	4.43E-6	22.8138	<.0001
Sav*CC		1	0.2240	0.0702	10.1952	0.0014
Dep*ATM		1	-0.1063	0.0276	14.8454	0.0001
SavBal*MM		1	-0.00003	6.697E-6	17.9087	<.0001
CD*MM		1	-0.4979	0.1204	17.1153	<.0001
SavBal*IRA		1	-0.00002	4.759E-6	14.6885	0.0001
MM*IRABal		1	-0.00002	5.724E-6	8.8578	0.0029
SavBal*B_DDABal		1	-1.67E-6	1.201E-7	192.3418	<.0001
Sav*B_DDABal		1	-0.0106	0.00135	61.5090	<.0001
MM*B_DDABal		1	-0.0150	0.00213	49.2303	<.0001
IRA*B_DDABal		1	-0.0108	0.00291	13.7755	0.0002
SavBal*ATMAmt		1	1.663E-9	7.08E-10	5.5194	0.0188
DDA*ATMAmt		1	0.000077	0.000019	16.7930	<.0001
branch_swoe*ATMAmt		1	0.000164	0.000023	51.6642	<.0001
B_DDABal*ATMAmt		1	-1.22E-6	2.418E-7	25.4494	<.0001
Sav*NSF		1	-0.4561	0.1277	12.7568	0.0004
SavBal*SDB		1	-0.00001	3.758E-6	11.8839	0.0006
ATMAmt*DepAmt		1	-1.73E-9	3.6E-10	22.9479	<.0001
B_DDABal*CashBk		1	0.0184	0.00584	9.8889	0.0017

Association of Predicted Probabilities and Observed Responses

Percent Concordant	78.9	Somers' D	0.578
Percent Discordant	21.1	Gamma	0.578
Percent Tied	0.0	Tau-a	0.262
Pairs	104768511	c	0.789

Odds Ratio Estimates and Profile-Likelihood Confidence Intervals

Effect		Unit	Estimate	95% Confidence Limits	
Phone		1.0000	0.966	0.931	1.002
ILS		1.0000	0.781	0.670	0.909
POS		1.0000	1.000	0.987	1.014
CCPurc		1.0000	0.989	0.914	1.071
CCBal		1.0000	1.000	1.000	1.000
Inv		1.0000	1.552	1.281	1.886
InArea		1.0000	0.957	0.813	1.127
Age		1.0000	0.999	0.997	1.002
MICRScor		1.0000	0.962	0.776	1.189
Income		1.0000	1.001	0.999	1.002
Res	R vs S	1.0000	1.024	0.941	1.113
Res	U vs S	1.0000	1.014	0.942	1.092

Only the variables not involved in an interaction have odds ratio estimates.

Example: Use the backward elimination method in PROC LOGISTIC to find a subset of inputs. Use the screened variable, **res**, and the 21 interactions detected in the forward selection method. Specify the FAST option, the HIERARCHY=SINGLE option, the BIC-based significance level, and the profile likelihood confidence intervals.

```
title1 "Backward Selection for Variable Annuity Data Set";
proc logistic data=work.train imputed_swoe_bins;
   class res (param=ref ref='S');
   model ins(event='1')= &screened res SavBal*B DDABal MM*B_DDABal
                branch_swoe*ATMAmt B_DDABal*Sav SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
                SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt
                Dep*ATM IRA*B_DDABal CD*MM MM*IRABal CD*Sav
                B DDABal*CashBk Sav*CC / clodds=pl selection=backward
                slstay=&sl hier=single fast;
run;
```

Selected MODEL statement options:

FAST uses a computational algorithm to compute a first-order approximation to the remaining slope estimates for each subsequent elimination of a variable from the model. Variables are removed from the model based on these approximate estimates.

SLSTAY= specifies the significance level of the Wald chi-square for an effect to stay in the model in a backward elimination step.

HIER= specifies how model hierarchy is to be applied. For HIER=SINGLE, only one effect can enter or leave the model at one time, subject to model hierarchy.

Partial Output

Summary of Backward Elimination

Step	Effect Removed	DF	Number In	Wald Chi-Square	Pr > ChiSq	Variable Label
1	POS	1	48	0.0017	0.9673	Number Point of Sale
1	Res	2	47	0.3178	0.8531	Area Classification
1	CCPurc	1	46	0.0713	0.7895	Credit Card Purchases
1	MICRScor	1	45	0.1243	0.7244	
1	Age	1	44	0.2614	0.6091	Age
1	InArea	1	43	0.2819	0.5954	Local Address
1	Income	1	42	1.0045	0.3162	Income
1	Phone	1	41	3.4393	0.0637	Number Telephone Banking
1	CCBal	1	40	4.2769	0.0386	Credit Card Balance
1	SavBal*ATMAmt	1	39	5.5083	0.0189	
1	MM*IRABal	1	38	8.5373	0.0035	
1	IRABal	1	37	0.0682	0.7939	IRA Balance
1	B_DDABal*CashBk	1	36	9.9528	0.0016	
1	CashBk	1	35	0.8386	0.3598	Number Cash Back

Type 3 Analysis of Effects

Effect	DF	Wald Chi-Square	Pr > ChiSq
SavBal	1	308.5136	<.0001
Dep	1	4.1510	0.0416
DDA	1	8.8417	0.0029
CD	1	224.8969	<.0001
Sav	1	121.4798	<.0001
CC	1	5.1746	0.0229
ATM	1	0.2447	0.6208
MM	1	144.7969	<.0001
branch_swoe	1	136.5762	<.0001
IRA	1	32.6535	<.0001
B_DDABal	1	755.9515	<.0001
ATMAmt	1	43.7181	<.0001
ILS	1	11.3597	0.0008
NSF	1	42.7157	<.0001
SDB	1	6.8696	0.0088
DepAmt	1	1.2025	0.2728
Inv	1	18.3357	<.0001
SavBal*B_DDABal	1	189.7509	<.0001
MM*B_DDABal	1	52.2924	<.0001
branch_swoe*ATMAmt	1	43.5764	<.0001
Sav*B_DDABal	1	63.0526	<.0001
SavBal*SDB	1	13.6189	0.0002
SavBal*DDA	1	31.1188	<.0001
ATMAmt*DepAmt	1	38.0936	<.0001
B_DDABal*ATMAmt	1	21.1726	<.0001
SavBal*IRA	1	10.0760	0.0015
SavBal*MM	1	17.0591	<.0001
SavBal*CC	1	27.6861	<.0001
Sav*NSF	1	11.8468	0.0006
DDA*ATMAmt	1	20.5372	<.0001
Dep*ATM	1	15.6629	<.0001
IRA*B_DDABal	1	14.0585	0.0002
CD*MM	1	17.5899	<.0001
CD*Sav	1	11.5000	0.0007
Sav*CC	1	12.4844	0.0004

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-2.0456	0.0987	429.4626	<.0001
SavBal	1	0.000169	9.641E-6	308.5136	<.0001
Dep	1	-0.0510	0.0250	4.1510	0.0416
DDA	1	-0.1852	0.0623	8.8417	0.0029
CD	1	1.1461	0.0764	224.8969	<.0001
Sav	1	1.0052	0.0912	121.4798	<.0001
CC	1	0.1110	0.0488	5.1746	0.0229
ATM	1	0.0305	0.0618	0.2447	0.6208
MM	1	1.7655	0.1467	144.7969	<.0001
branch_swoe	1	0.9386	0.0803	136.5762	<.0001
IRA	1	1.1179	0.1956	32.6535	<.0001

B_DDABal	1	0.0284	0.00103	755.9515	<.0001
ATMAmt	1	0.000178	0.000027	43.7181	<.0001
ILS	1	-0.2602	0.0772	11.3597	0.0008
NSF	1	0.6022	0.0921	42.7157	<.0001
SDB	1	0.1394	0.0532	6.8696	0.0088
DepAmt	1	3.281E-6	2.992E-6	1.2025	0.2728
Inv	1	0.4212	0.0984	18.3357	<.0001
SavBal*B_DDABal	1	-1.67E-6	1.213E-7	189.7509	<.0001
MM*B_DDABal	1	-0.0154	0.00213	52.2924	<.0001
branch_swoe*ATMAmt	1	0.000138	0.000021	43.5764	<.0001
Sav*B_DDABal	1	-0.0107	0.00135	63.0526	<.0001
SavBal*SDB	1	-0.00001	3.785E-6	13.6189	0.0002
SavBal*DDA	1	0.000025	4.524E-6	31.1188	<.0001
ATMAmt*DepAmt	1	-1.44E-9	2.34E-10	38.0936	<.0001
B_DDABal*ATMAmt	1	-9.96E-7	2.164E-7	21.1726	<.0001
SavBal*IRA	1	-0.00002	4.828E-6	10.0760	0.0015
SavBal*MM	1	-0.00003	6.731E-6	17.0591	<.0001
SavBal*CC	1	-0.00002	4.551E-6	27.6861	<.0001
Sav*NSF	1	-0.4380	0.1273	11.8468	0.0006
DDA*ATMAmt	1	0.000071	0.000016	20.5372	<.0001
Dep*ATM	1	-0.1071	0.0271	15.6629	<.0001
IRA*B_DDABal	1	-0.0109	0.00289	14.0585	0.0002
CD*MM	1	-0.5042	0.1202	17.5899	<.0001
CD*Sav	1	-0.3225	0.0951	11.5000	0.0007
Sav*CC	1	0.2478	0.0701	12.4844	0.0004

The results show that the backward elimination method reduced the number of variables from 49 to 35 (17 main effects and 18 interactions). The variable **DepAmt** was not eliminated even though the *p*-value was high because **DepAmt** was involved in an interaction.

Association of Predicted Probabilities and Observed Responses

Percent Concordant	78.8	Somers' D	0.576
Percent Discordant	21.2	Gamma	0.576
Percent Tied	0.0	Tau-a	0.261
Pairs	104768511	c	0.788

Odds Ratio Estimates and Profile-Likelihood Confidence Intervals

Effect	Unit	Estimate	95% Confidence Limits	
ILS	1.0000	0.771	0.662	0.896
Inv	1.0000	1.524	1.258	1.850

Only variables not involved in an interaction are shown in the Odds Ratio Estimates and Profile-Likelihood Confidence Intervals table.

Example: Fit a logistic regression model on the inputs selected by the backward elimination method and only display the odds ratios for **B_DDABAL** for several values of **SavBal**.

```
title1 "Candidate Model for Variable Annuity Data Set";
ods select OddsRatiosPL;
proc logistic data=work.train imputed swoe bins;
   model ins(event='1')= SavBal Dep DDA CD Sav CC ATM MM branch_swoe
         IRA B DDABal ATMAmt ILS NSF SDB DepAmt Inv SavBal*B DDABal
         MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
         SavBal*DDA AtmAmt*DepAmt B_DDABAL*ATMAmt SavBal*IRA
```

```
          SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
          CD*MM CD*Sav Sav*CC / clodds=pl;
   oddsratio B_DDABAL / at(savbal=0, 1211, 52299) cl=pl;
run;
```

```
                Odds Ratio Estimates and Profile-Likelihood Confidence Intervals

     Odds Ratio                                                                 Estimate

     B_DDABal at SavBal=0 Sav=0.4684 MM=0.1137 IRA=0.0529 ATMAmt=1252.6            1.020
     B_DDABal at SavBal=1211 Sav=0.4684 MM=0.1137 IRA=0.0529 ATMAmt=1252.6         1.018
     B_DDABal at SavBal=52299 Sav=0.4684 MM=0.1137 IRA=0.0529 ATMAmt=1252.6        0.935

                Odds Ratio Estimates and Profile-Likelihood Confidence Intervals

Odds Ratio                                                          95% Confidence Limits

B_DDABal at SavBal=0 Sav=0.4684 MM=0.1137 IRA=0.0529 ATMAmt=1252.6      1.019        1.021
B_DDABal at SavBal=1211 Sav=0.4684 MM=0.1137 IRA=0.0529 ATMAmt=1252.6   1.017        1.019
B_DDABal at SavBal=52299 Sav=0.4684 MM=0.1137 IRA=0.0529 ATMAmt=1252.6  0.923        0.946
```

The effect of **B_DDABAL** is changed by the values of **SavBal**. The effect is positive for small values of **SavBal**, but it is negative for large values of **SavBal**. The other variables involved in the interaction with **B_DDABAL** are set at their means.

Example: Use the **interact** macro to create an interaction plot of **B_DDABAL** and **SavBal**.

```
%macro interact(data=,target=,event=,inputs=,var1=,var2=,mean_inputs=);
```

The CODE statement is used to create scoring code.

```
proc logistic data=&data noprint;
   model &target(event="&event")=&inputs;
   code file="&PMLRfolder\interaction.txt";
run;
```

PROC UNIVARIATE and the DATA step are used to create macro variables for the percentiles 5, 25, 50, 75, and 95 of **SavBal** and **B_DDABAL**.

```
proc univariate data=&data noprint;
   var &var1 &var2;
   output out=work.percentiles pctlpts=5 25 50 75 95 pctlpre=&var1._p
          &var2._p;
run;

data _null_;
   set work.percentiles;
   call symput("&var1._p5",&var1._p5);
   call symput("&var1. p25",&var1. p25);
   call symput("&var1._p50",&var1._p50);
   call symput("&var1. p75",&var1. p75);
   call symput("&var1._p95",&var1._p95);
   call symput("&var2._p5",&var2._p5);
   call symput("&var2. p25",&var2. p25);
   call symput("&var2._p50",&var2._p50);
   call symput("&var2._p75",&var2._p75);
```

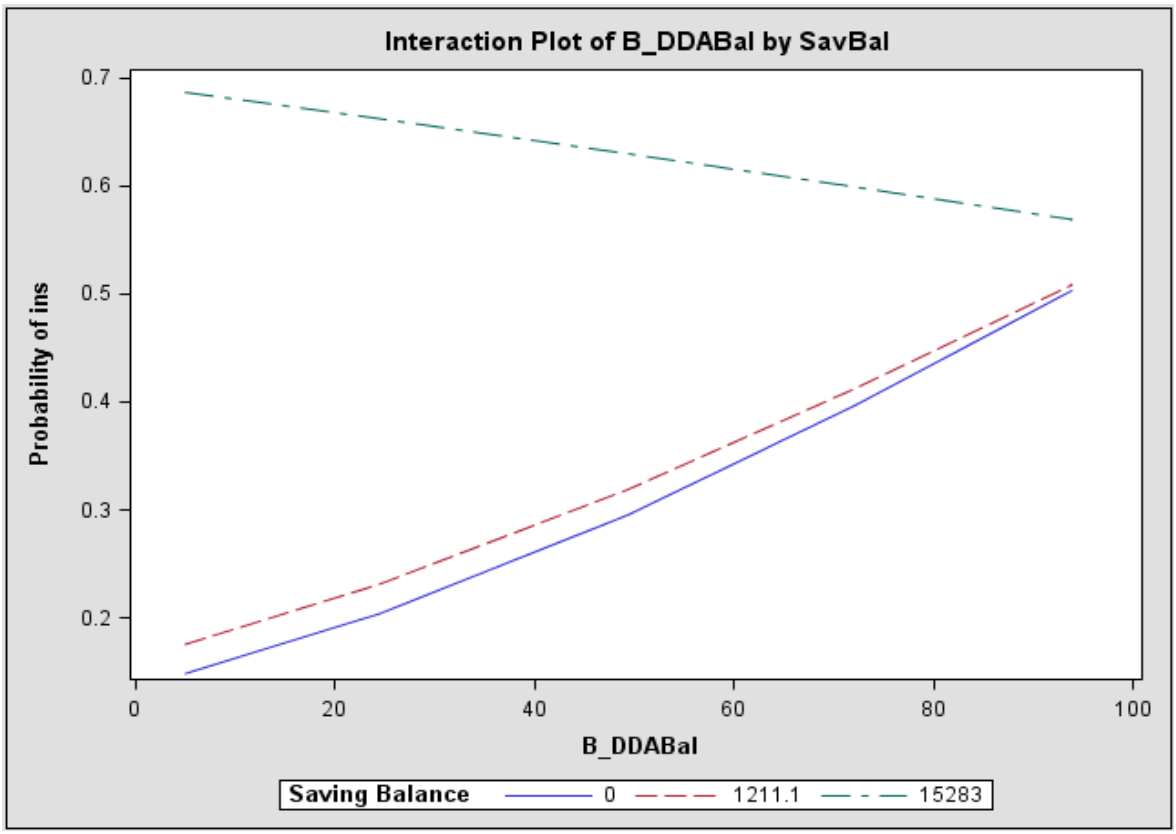
```
   call symput("&var2._p95",&var2._p95);
run;
```

PROC MEANS is used to compute the means of the variables that are not involved in the interaction. This step is necessary because only observations with values for all the variables are scored. The DATA step is used to create the plotting points and the scoring code is used to compute the predicted probabilities.

```
proc means data=&data noprint;
   var &mean inputs;
   output out=work.plot mean=;
run;

data work.plot(drop=_type_ _freq_);
   set work.plot;
   do &var2 = &&&var2._p5,&&&var2._p25,&&&var2._p50,&&&var2._p75,
              &&&var2. p95;
      do &var1 = &&&var1._p5,&&&var1._p25,&&&var1._p50,&&&var1._p75,
                 &&&var1. p95;
      %include "&PMLRfolder\interaction.txt";
      output;
      end;
   end;
run;

title1 "Interaction Plot of &var2 by &var1";
proc sgplot data=work.plot;
   series y=p_&target&event x=&var2 / group=&var1;
   yaxis label="Probability of &target";
run;

%mend interact;
```

The arguments for the macro are the data set name, the target variable name, the event category, the inputs in the logistic model, the two variables involved in the interaction, and the variables not involved in the interaction.

```
%interact(data=train imputed swoe bins,target=ins,event=1,inputs=
          SavBal Dep DDA CD Sav CC ATM MM branch_swoe IRA B_DDABal
          ATMAmt ILS NSF SDB DepAmt Inv SavBal*B_DDABal MM*B_DDABal
          branch swoe*ATMAmt Sav*B DDABal SavBal*SDB SavBal*DDA
          AtmAmt*DepAmt B_DDABAL*ATMAmt SavBal*IRA SavBal*MM SavBal*CC
          Sav*NSF DDA*ATMAmt Dep*ATM IRA*B DDABal CD*MM CD*Sav Sav*CC,
          var1=SavBal,var2=B_DDABal,mean_inputs=SavBal Dep DDA CD Sav
          CC ATM MM branch_swoe IRA B_DDABal ATMAmt ILS NSF SDB DepAmt
          Inv);
```



The interaction plot shows the positive effects of **B_DDABAL** at low values of **SavBal** and the negative effects of **B_DDABAL** at high values of **SavBal**.

Example: Use the best subsets selection method (SELECTION=SCORE) to find a subset of inputs. Use the BEST= option to limit the amount of output. Because the best subsets method does not support class variables, create dummy variables for **Res** in a DATA step. Include the screened variables, the dummy codes for **Res**, and the 21 interactions detected in the forward selection.

```
data work.train_imputed_swoe_bins;
   set work.train_imputed_swoe_bins;
   resr=(res='R');
   resu=(res='U');
run;

title1 "Models Selected by Best Subsets Selection";
proc logistic data=train_imputed_swoe_bins;
   model ins(event='1')=&screened resr resu SavBal*B_DDABal
         MM*B DDABal branch swoe*ATMAmt B DDABal*Sav SavBal*SDB
         SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
         SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
         IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
         / selection=score best=1;
run;
```

Selected MODEL statement option:

BEST=*n* specifies that *n* models with the highest score chi-square statistics are to be displayed for each model size. It is used exclusively with the SCORE model selection method.

Partial Output

Regression Models Selected by Score Criterion

Number of Variables	Score Chi-Square	Variables Included in Model
1	1872.2915	B_DDABal
2	2490.9205	CD B_DDABal
3	2935.4526	SavBal CD B_DDABal
4	3240.0673	SavBal CD B_DDABal SavBal*B_DDABal
5	3501.6203	SavBal CD branch_swoe B_DDABal SavBal*B_DDABal
6	3731.8960	SavBal CD Sav MM branch_swoe B_DDABal
7	3952.2174	SavBal CD Sav MM branch_swoe B_DDABal SavBal*B_DDABal
8	4039.4342	SavBal Dep CD Sav MM branch_swoe B_DDABal SavBal*B_DDABal
9	4086.6501	SavBal Dep CD Sav MM branch_swoe IRA B_DDABal SavBal*B_DDABal
10	4129.3415	SavBal CD Sav MM branch_swoe IRA B_DDABal SavBal*B_DDABal DDA*ATMAmt Dep*ATM
11	4158.9269	SavBal CD Sav MM branch_swoe IRA B_DDABal SavBal*B_DDABal MM*B_DDABal DDA*ATMAmt Dep*ATM
12	4190.0309	SavBal CD Sav MM branch_swoe IRA B_DDABal ATMAmt SavBal*B_DDABal SavBal*SDB B_DDABal*ATMAmt Dep*ATM
13	4221.5081	SavBal CD Sav MM branch_swoe IRA B_DDABal ATMAmt SavBal*B_DDABal branch_swoe*ATMAmt SavBal*SDB B_DDABal*ATMAmt Dep*ATM
14	4252.2092	SavBal CD Sav MM branch_swoe IRA B_DDABal ATMAmt SavBal*B_DDABal branch_swoe*ATMAmt B_DDABal*ATMAmt SavBal*CC Dep*ATM Sav*CC
15	4276.0193	SavBal CD Sav MM branch_swoe IRA B_DDABal ATMAmt Inv SavBal*B_DDABal branch_swoe*ATMAmt B_DDABal*ATMAmt SavBal*CC Dep*ATM Sav*CC
16	4299.3353	SavBal CD Sav MM branch_swoe IRA B_DDABal ATMAmt Inv SavBal*B_DDABal branch_swoe*ATMAmt B_DDABal*ATMAmt SavBal*MM SavBal*CC Dep*ATM Sav*CC
17	4321.5464	SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt SavBal*DDA B_DDABal*ATMAmt SavBal*CC Dep*ATM Sav*CC
18	4342.9401	SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt Inv SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt SavBal*DDA B_DDABal*ATMAmt SavBal*CC Dep*ATM Sav*CC
19	4360.4921	SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt NSF Inv SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt SavBal*DDA B_DDABal*ATMAmt SavBal*CC Dep*ATM Sav*CC

```
20   4376.2334  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt SavBal*DDA
                B_DDABal*ATMAmt SavBal*MM SavBal*CC Dep*ATM Sav*CC

21   4389.9895  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt SavBal*DDA
                B_DDABal*ATMAmt SavBal*MM SavBal*CC Dep*ATM IRA*B_DDABal Sav*CC

22   4405.7177  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt SavBal*DDA ATMAmt*DepAmt
                B_DDABal*ATMAmt SavBal*ATMAmt SavBal*CC DDA*ATMAmt Dep*ATM Sav*CC

23   4421.9221  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt SavBal*DDA ATMAmt*DepAmt
                B_DDABal*ATMAmt SavBal*ATMAmt SavBal*MM SavBal*CC DDA*ATMAmt Dep*ATM
                Sav*CC

24   4435.1337  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt SavBal*SDB SavBal*DDA
                ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*MM SavBal*CC
                DDA*ATMAmt Dep*ATM Sav*CC

25   4448.4026  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*MM
                SavBal*CC DDA*ATMAmt Dep*ATM Sav*CC

26   4460.2073  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*MM
                SavBal*CC DDA*ATMAmt Dep*ATM IRA*B_DDABal Sav*CC

27   4471.0618  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt ILS NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*MM
                SavBal*CC DDA*ATMAmt Dep*ATM IRA*B_DDABal Sav*CC

28   4480.0708  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt ILS NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*MM
                SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal Sav*CC

29   4488.3877  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt ILS NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*MM
                SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM Sav*CC

30   4494.9329  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt ILS NSF Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
                SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM Sav*CC

31   4501.5413  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt ILS NSF SDB Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*MM
                SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM CD*Sav Sav*CC
```

```
32   4507.5847  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt ILS NSF SDB Inv
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
                SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM CD*Sav
                Sav*CC

33   4512.5798  SavBal DDA CD Sav MM branch_swoe IRA B_DDABal ATMAmt ILS NSF SDB CCBal
                Inv SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal
                SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
                SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
                CD*MM CD*Sav Sav*CC

34   4516.8991  SavBal DDA CD Sav MM branch_swoe Phone IRA B_DDABal ATMAmt ILS NSF SDB
                CCBal Inv SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal
                SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
                SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
                CD*MM CD*Sav Sav*CC

35   4520.7835  SavBal DDA CD Sav MM branch_swoe IRA IRABal B_DDABal ATMAmt ILS NSF SDB
                CCBal Inv SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal
                SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
                SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
                CD*MM MM*IRABal CD*Sav Sav*CC

36   4525.1160  SavBal DDA CD Sav MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt ILS
                NSF SDB CCBal Inv SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt
                Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt
                SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
                IRA*B_DDABal CD*MM MM*IRABal CD*Sav Sav*CC

37   4528.2905  SavBal DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt ILS
                NSF SDB CCBal Inv SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt
                Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt
                SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
                IRA*B_DDABal CD*MM MM*IRABal CD*Sav Sav*CC

38   4530.4633  SavBal DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt ILS
                NSF SDB DepAmt CCBal Inv SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt
                Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt
                SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
                IRA*B_DDABal CD*MM MM*IRABal CD*Sav Sav*CC

39   4533.4881  SavBal DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt ILS
                NSF SDB CCBal Inv CashBk SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt
                Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt
                SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
                IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC

40   4535.6452  SavBal DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt ILS
                NSF SDB DepAmt CCBal Inv CashBk SavBal*B_DDABal MM*B_DDABal
                branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt
                B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF
                DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk
                Sav*CC
```

```
41   4538.2246  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                ILS NSF SDB DepAmt CCBal Inv CashBk SavBal*B_DDABal MM*B_DDABal
                branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt
                B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF
                DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk
                Sav*CC

42   4538.8480  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                ILS NSF SDB DepAmt CCBal Inv CashBk Income SavBal*B_DDABal MM*B_DDABal
                branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt
                B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF
                DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk
                Sav*CC

43   4539.2520  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                ILS NSF SDB DepAmt CCBal Inv InArea CashBk Income SavBal*B_DDABal
                MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB SavBal*DDA
                ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA SavBal*MM
                SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM MM*IRABal CD*Sav
                B_DDABal*CashBk Sav*CC

44   4539.3062  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                ILS NSF SDB DepAmt CCBal Inv InArea CashBk Income resr SavBal*B_DDABal
                MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB SavBal*DDA
                ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA SavBal*MM
                SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM MM*IRABal CD*Sav
                B_DDABal*CashBk Sav*CC

45   4539.4348  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                ILS NSF SDB DepAmt CCBal Inv InArea CashBk Income resr resu
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
                SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM
                MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC

46   4539.4668  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                ILS NSF SDB DepAmt CCBal Inv InArea Age CashBk Income resr resu
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
                SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM
                MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC

47   4539.4936  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                ILS NSF SDB DepAmt CCBal Inv InArea Age CashBk MICRScor Income resr resu
                SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB
                SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
                SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal CD*MM
                MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC

48   4539.5150  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                ILS POS NSF SDB DepAmt CCBal Inv InArea Age CashBk MICRScor Income resr
                resu SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal
                SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
                SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
                CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
```

```
      49   4539.5155  SavBal Dep DDA CD Sav CC MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
                      ILS POS NSF CCPurc SDB DepAmt CCBal Inv InArea Age CashBk MICRScor
                      Income resr resu SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt
                      Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt
                      SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
                      IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC

      50   4539.5156  SavBal Dep DDA CD Sav CC ATM MM branch_swoe Phone IRA IRABal B_DDABal
                      ATMAmt ILS POS NSF CCPurc SDB DepAmt CCBal Inv InArea Age CashBk
                      MICRScor Income resr resu SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt
                      Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt
                      SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
                      IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC
```

In best subsets selection, model hierarchy is not maintained. For example, model 10 is not hierarchically well formulated. Furthermore, the model with the best fit to the data is not readily apparent. The score test statistic should not be used because it increases with model size. The Bayesian information criterion (BIC) is often used for model selection. The BIC is essentially the –2 log likelihood plus a penalty term that increases as the model becomes bigger. The penalty term for BIC is $(k+1)\cdot\ln(n)$, where k is the number of variables in the model and n is the sample size. Smaller values of BIC are preferable. The BIC is the same as the Schwarz criterion (SC) except when the event or trials syntax is used.

When the SELECTION=SCORE option is used, output data sets are not available. However, the Output Delivery System can be used to create an output data set with the variables selected in the all subsets selection. The SCORE statement can then be used with the FITSTAT option to generate model fit statistics for each model that is selected in the all subsets selection. Because a series of logistic regression models must be scored, the scoring process is in a macro DO loop.

Example: Use the **fitstat** macro to generate fit statistics for the models generated by the best subsets selection.

The first step in the macro is to create an output data set with the results of the best subsets selection.

```
%macro fitstat(data=,target=,event=,inputs=,best=,priorevent=);

ods select none;
ods output bestsubsets=work.score;

proc logistic data=&data namelen=50;
   model &target(event="&event")=&inputs / selection=score best=&best;
run;
```

With an SQL SELECT INTO function, the names and number of variables selected from the best subsets selection are transferred to macro variables. The automatic macro variable **sqlobs** gives the number of models produced overall.

```
proc sql noprint;
   select variablesinmodel into :inputs1 -
   from work.score;
   select NumberOfVariables into :ic1 -
   from work.score;
quit;

%let lastindx = &SQLOBS;
```

The next step is a DO loop that scores each model that is selected from the best subsets selection and generates model fit statistics using PROC LOGISTIC. The FITSTAT option in the SCORE statement displays the fit statistics for the data that you are scoring. The macro variable **im** is set equal to the variable names in the **model_indx** model. The macro variable **ic** is set equal to the number of variables in the **model_indx** model. Because a data set is created with the predicted information for each model that is scored by PROC LOGISTIC, PROC DATASETS is used to delete these unnecessary data sets. Only the data sets with the fit statistics for each model scored are kept.

```
%do model_indx=1 %to &lastindx;

%let im=&&inputs&model indx;
%let ic=&&ic&model_indx;

ods output scorefitstat=work.stat&ic;
proc logistic data=&data namelen=50;
  model &target(event="&event")=&im;
  score data=&data out=work.scored fitstat
        priorevent=&priorevent;
run;

proc datasets
   library=work
   nodetails
   nolist;
   delete scored;
run;
quit;

%end;
```

The data sets with the model fit statistics are concatenated in the next step.

```
data work.modelfit;
   set work.stat1 - work.stat&lastindx;
   model = _n_;
run;

%mend fitstat;
```

The arguments for the macro are the data set name, the target variable name, the event category, the inputs in the logistic model, the number models with the highest score chi-square statistics to be displayed for each model size, and the prior event probability.

```
%fitstat(data=train_imputed_swoe_bins,target=ins,event=1,
         inputs=&screened resr resu SavBal*B_DDABal MM*B_DDABal
         branch swoe*ATMAmt B DDABal*Sav SavBal*SDB SavBal*DDA
         ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA
         SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM IRA*B_DDABal
         CD*MM MM*IRABal CD*Sav B_DDABal*CashBk Sav*CC, best=1,
         priorevent=0.02);
```

The models are sorted by the Bayesian Information criterion.

```
proc sort data = work.modelfit;
   by bic;
run;

title1 "Fit Statistics from Models selected from Best-Subsets";
proc print data=work.modelfit;
   var model auc aic bic misclass adjrsquare brierscore;
run;
```

Fit Statistics from Models selected from Best-Subsets

Obs	model	AUC	AIC	BIC	MisClass	Adj RSquare	Brier Score
1	35	0.788365	50279.66	50566.81	0.3398	0.354277	0.30715
2	30	0.787561	50321.71	50568.98	0.3400	0.352545	0.307317
3	33	0.788091	50299.66	50570.86	0.3399	0.353479	0.307291
4	36	0.788464	50275.9	50571.03	0.3398	0.354468	0.307156
5	28	0.787377	50340.06	50571.38	0.3399	0.3518	0.307333
6	27	0.787237	50348.13	50571.47	0.3396	0.351464	0.307294
7	37	0.788533	50271.74	50574.84	0.3397	0.354673	0.307166
8	34	0.788171	50295.93	50575.11	0.3399	0.353669	0.307297
9	32	0.788012	50313.5	50576.72	0.3399	0.352952	0.307405
10	26	0.787156	50364.17	50579.53	0.3397	0.350862	0.307379
11	39	0.788885	50261.01	50580.06	0.3398	0.355162	0.307148
12	38	0.78862	50270.52	50581.6	0.3398	0.35478	0.307158
13	29	0.78739	50342.95	50582.24	0.3401	0.35177	0.307446
14	25	0.786873	50377.32	50584.7	0.3397	0.350356	0.307381
15	41	0.78915	50251	50586.01	0.3397	0.355627	0.307095
16	40	0.788968	50259.86	50586.89	0.3398	0.355267	0.307139
17	31	0.787825	50334.21	50589.45	0.3401	0.352195	0.307527
18	24	0.785995	50391.72	50591.13	0.3396	0.349807	0.307081
19	42	0.789184	50251.19	50594.17	0.3397	0.355687	0.307086
20	23	0.785837	50409.67	50601.1	0.3398	0.34914	0.307214
21	43	0.789204	50252.82	50603.78	0.3397	0.3557	0.307084
22	44	0.78921	50254.79	50613.73	0.3397	0.355701	0.307088
23	45	0.789198	50256.66	50623.57	0.3398	0.355705	0.307087
24	22	0.785666	50442.77	50626.22	0.3399	0.347964	0.3074
25	46	0.789209	50258.16	50633.05	0.3397	0.355721	0.307084
26	47	0.789211	50259.91	50642.78	0.3397	0.35573	0.307083
27	48	0.789211	50261.94	50652.78	0.3397	0.355729	0.307083
28	49	0.789211	50264.01	50662.83	0.3398	0.355727	0.307084
29	50	0.789214	50265.52	50672.32	0.3398	0.355743	0.307084
30	21	0.785887	50504.3	50679.78	0.3406	0.34583	0.307937
31	20	0.785632	50518.06	50685.56	0.3406	0.3453	0.307924
32	19	0.785453	50552.23	50711.76	0.3407	0.344082	0.308113
33	18	0.784705	50603.71	50755.26	0.3409	0.342277	0.30829
34	16	0.782463	50637.91	50773.51	0.3400	0.340984	0.307593
35	17	0.78402	50644.78	50788.35	0.3409	0.34082	0.308477
36	15	0.781906	50687.86	50815.48	0.3401	0.339224	0.307684
37	14	0.781262	50732.62	50852.26	0.3400	0.337635	0.307886
38	13	0.780791	50792.49	50904.16	0.3403	0.335527	0.308246
39	11	0.780299	50940.52	51036.24	0.3413	0.330323	0.309675
40	12	0.779831	50933.85	51037.54	0.3413	0.330621	0.309303

41	10	0.779263	50971.56	51059.3	0.3413	0.329188	0.309454
42	9	0.778262	51084.22	51163.98	0.3415	0.325232	0.309961
43	8	0.776789	51170.55	51242.34	0.3413	0.322171	0.310272
44	7	0.774855	51293.03	51356.84	0.3412	0.317835	0.310613
45	6	0.763777	51860.46	51916.29	0.3419	0.297678	0.312052
46	5	0.764306	51870.2	51918.06	0.3406	0.297257	0.311568
47	4	0.755897	52361.16	52401.04	0.3406	0.279361	0.31313
48	3	0.741249	53024.22	53056.13	0.3404	0.254558	0.314466
49	2	0.703875	54890.87	54914.8	0.3464	0.180621	0.32416
50	1	0.678648	55982.66	55998.61	0.3464	0.134267	0.32753

Besides the BIC, other displayed statistics include the area under the ROC curve (AUC), Akaike's information criterion (AIC), the misclassification rate (the number of false negatives and false positives over the total using a cutoff of 0.50), the maximum-rescaled R-square statistic (adjusted R-square), and the Brier score. The Brier score measures the average squared deviation between predicted probabilities for a set of events and their outcomes, so a lower score represents higher accuracy. The adjusted R-square statistics, which can achieve a maximum value of 1, are most useful for comparing competing models that are not necessarily nested. Larger values indicate better models.

The results reveal that the 35-input model has the smallest value of **bic**. The following SQL code creates a macro variable **selected** that contains the names of the inputs in that model.

```
%global selected;
proc sql;
   select VariablesInModel into :selected
   from work.score
   where numberofvariables=35;
quit;
```

```
Variables Included in Model
_____________________________________________________________________________________

SavBal DDA CD Sav MM branch_swoe IRA IRABal B_DDABal ATMAmt ILS NSF SDB CCBal Inv
SavBal*B_DDABal MM*B_DDABal branch_swoe*ATMAmt Sav*B_DDABal SavBal*SDB SavBal*DDA ATMAmt*DepAmt
B_DDABal*ATMAmt SavBal*ATMAmt SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
IRA*B_DDABal CD*MM MM*IRABal CD*Sav Sav*CC
```

The model with the lowest BIC is not hierarchically well-formulated. (**DepAmt**, **Dep**, **ATM**, and **CC** must be added.) However, model hierarchy is important in inferential statistics, but it is not as important an issue in predictive modeling.

These automatic selection routines raise several questions. For techniques like stepwise selection and backward elimination, what are good stopping rules? For best subsets, what number of inputs yields the best model?

The answers to these questions are in the purposes of the models. The goal of most predictive modeling is generalization. Hence, the best model is the model that generalizes to new cases the best. How do you measure the generalizing ability of a model? What are some statistics that summarize a model's performance? (The next chapter has suggestions for comparing and selecting models.)

End of Demonstration

Exercises

6. Subset Selection Methods

a. Include the program **pmlr03e03.sas** or use the **PMLR_UpToDate** macro. Compute a BIC-based significance level and use the sample size for *n*. Then use the forward selection method to detect important two-factor interactions. Use the INCLUDE option to include all 26 screened variables and use the macro variable **&ex_screened**. Using the NAMELEN= option in the PROC LOGISTIC statement, set the maximum length of effect names to 50.

Which interactions were detected?

b. Use the backward elimination method in PROC LOGISTIC to find a subset of inputs. Use the screened variables and the interactions that are detected in the forward selection method. Specify the FAST option, the HIERARCHY=SINGLE option, the BIC-based significance level, and the profile likelihood confidence intervals.

How many inputs were in the final model?

c. Generate fit statistics for the models that were generated in the all subsets selection. Use the **fitstat** macro to score each of the models.

What is the *c* statistic for the model selected by best subsets selection with the lowest BIC?

End of Exercises

3.07 Multiple Choice Poll

What is the *c* statistic for the model selected by best subsets selection with the lowest BIC?

a. .610

b. .621

c. .628

d. .632

3.6 Chapter Summary

Preparing the data for predictive modeling can be laborious. First, missing values need to be replaced with reasonable values. Missing indicator variables are also needed if missingness is related to the target. If there are nominal input variables with numerous levels, the levels should be collapsed to reduce the likelihood of quasi-complete separation and to reduce the redundancy among the levels. Furthermore, if there are numerous input variables, variable clustering should be performed to reduce the redundancy among the variables. In addition, there are several selection methods in the LOGISTIC procedure to select a subset of variables.

To assist in identifying nonlinear associations, the Hoeffding's D statistic can be used. A variable with a low rank in the Spearman correlation statistic, but with a high rank in the Hoeffding's D statistic, might indicate that the association with the target is nonlinear.

General form of the STDIZE procedure:

```
PROC STDIZE DATA=SAS-data-set <options>;
     VAR variables;
RUN;
```

General form of the CLUSTER procedure:

```
PROC CLUSTER DATA=SAS-data-set <options>;
     FREQ variable;
     VAR variable;
     ID variable;
RUN;
```

General form of the VARCLUS procedure:

```
PROC VARCLUS DATA=SAS-data-set <options>;
     VAR variables;
RUN;
```

3.7 Solutions

Solutions to Exercises

1. Missing Value Imputation

a. Include the program **pmlr01s01.sas** or use the **PMLR_UpToDate** macro. Using the **pmlr.pva_train** data set, create missing value indicators for the inputs **DONOR_AGE**, **INCOME_GROUP**, and **WEALTH_RATING**. Call the new data set **pmlr.pva_train_mi**.

b. Submit the program below to group the variables **RECENT_RESPONSE_PROP** and **RECENT_AVG_GIFT_AMT** into three groups.

```
proc rank data=pmlr.pva train mi  out=pva train rank  groups=3;
   var recent response prop recent_avg_gift_amt;
   ranks grp_resp grp_amt;
run;
```

c. Sort **pva_train_rank** by **GRP_RESP** and **GRP_AMT** and call the output data set **pva_train_rank_sort**.

d. Use PROC STDIZE with a BY statement to impute missing values for each BY group and write the completed data set to output. Name the output data set **pva_train_imputed**.

e. Use the MEANS procedure to determine the values that the missing values were replaced with.

Note: An electronic copy of the solution program is in **pmlr03s01.sas**.

```
data pmlr.pva train mi(drop=i);
   set pmlr.pva train;
   /* name the missing indicator variables */
   array mi{*} mi DONOR AGE mi INCOME_GROUP
               mi WEALTH RATING;
   /* select variables with missing values */
   array x{*} DONOR AGE INCOME_GROUP WEALTH_RATING;
   do i=1 to dim(mi);
      mi{i}=(x{i}=.);
      nummiss+mi{i};
   end;
run;

proc rank data=pmlr.pva_train_mi out=work.pva_train_rank groups=3;
   var recent response prop recent_avg_gift_amt;
   ranks grp_resp grp_amt;
run;

proc sort data=work.pva_train_rank out=work.pva_train_rank_sort;
   by grp_resp grp_amt;
run;

proc stdize data=work.pva_train_rank_sort method=median
            reponly out=pmlr.pva_train_imputed;
   by grp_resp grp_amt;
   var DONOR_AGE INCOME_GROUP WEALTH_RATING;
run;
```

```
options nolabel;
proc means data=pmlr.pva train_imputed median;
   class grp_resp grp_amt;
   var DONOR_AGE INCOME_GROUP WEALTH_RATING;
run;
options label;
```

grp_resp	grp_amt	N Obs	Variable	Median
0	0	487	DONOR_AGE	65.0000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	5.0000000
	1	1147	DONOR_AGE	58.0000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	5.0000000
	2	1612	DONOR_AGE	58.0000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	6.0000000
1	0	671	DONOR_AGE	65.0000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	4.5000000
	1	1270	DONOR_AGE	59.0000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	5.0000000
	2	1202	DONOR_AGE	57.0000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	5.0000000
2	0	2155	DONOR_AGE	63.0000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	5.0000000
	1	733	DONOR_AGE	61.0000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	6.0000000
	2	410	DONOR_AGE	58.5000000
			INCOME_GROUP	4.0000000
			WEALTH_RATING	6.0000000

For the cell **GRP_RESP**=0 and **GRP_AMT**=0, what replaced the missing value for **DONOR_AGE**?
The missing value for DONOR_AGE was replaced with a value of 65.

2. Clustering Categorical Input Levels

a. Include the program **pmlr03e01.sas** or use the **PMLR_UpToDate** macro. Use Greenacre's correspondence analysis to cluster levels of **cluster_code**. Use PROC MEANS to generate a data set with information about the average response rate and sample size for each level of **cluster_code**.

b. Use PROC CLUSTER to group those levels and to create a horizontal dendrogram. Use the log of the *p*-value of the appropriate χ^2 test to determine which level of clustering is suitable.

c. Use PROC SGPLOT to plot the log of the *p*-value by the number of clusters (The range of the Y axis should be -40 to 0.) Use PROC TREE to create a data set of the final results.

d. Use the DATA step to create a file with the assignments for **cluster_code**. Define a new variable called **cluster_clus** and create a new data set called **pmlr.pva_train_imputed_clus** with the new assignments for **cluster_code**.

Note: An electronic copy of the solution program is in **pmlr03s02.sas**.

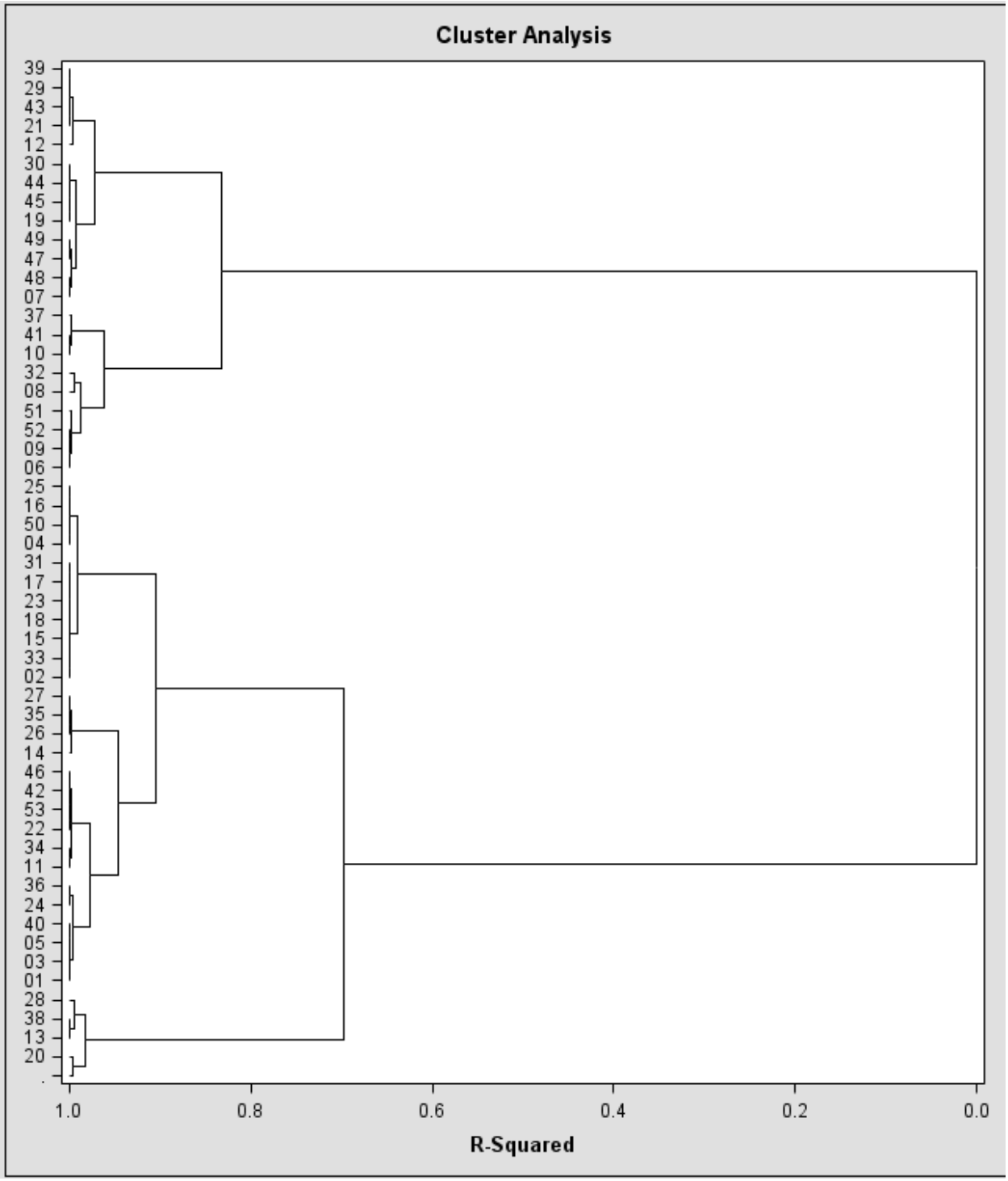
```
proc means data=pmlr.pva_train_imputed noprint nway;
   class CLUSTER_CODE;
   var target_b;
   output out=work.level mean=prop;
run;

ods output clusterhistory=work.cluster;
proc cluster data=work.level method=ward outtree=work.fortree
      plots=(dendrogram(horizontal height=rsq));
   freq _freq_;
   var prop;
   id CLUSTER_CODE;
run;
```

Partial Output

Cluster History

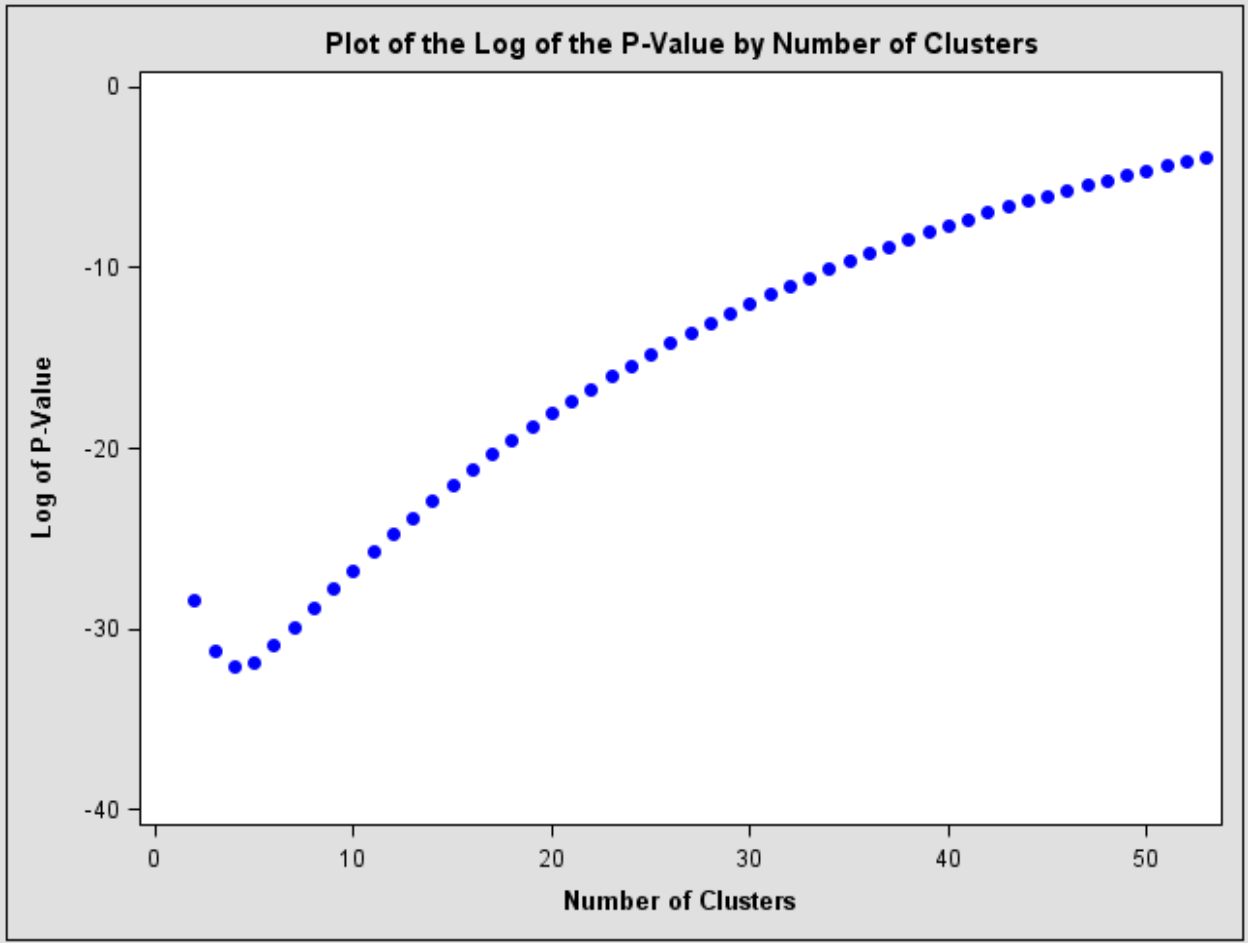
Number of Clusters	Clusters Joined		Freq	Semipartial R-Square	R-Square	Tie
53	16	25	325	0.0000	1.00	
52	19	45	291	0.0000	1.00	
51	03	05	254	0.0000	1.00	
50	18	23	455	0.0000	1.00	
49	CL52	44	473	0.0000	1.00	
48	47	49	432	0.0000	1.00	
47	22	53	265	0.0000	1.00	
46	15	CL50	565	0.0000	1.00	
45	26	35	472	0.0000	1.00	
44	07	48	192	0.0000	1.00	
43	09	52	107	0.0000	1.00	
42	17	31	296	0.0000	1.00	
41	02	33	247	0.0000	1.00	
40	11	34	374	0.0000	1.00	
39	CL47	42	397	0.0000	1.00	
38	04	50	126	0.0000	1.00	
37	21	43	407	0.0000	1.00	
36	29	39	344	0.0000	1.00	
35	CL51	40	662	0.0000	1.00	
34	06	CL43	179	0.0000	1.00	
33	10	41	422	0.0000	1.00	
32	CL46	CL42	861	0.0000	1.00	
31	CL45	27	793	0.0000	1.00	
30	01	CL35	763	0.0000	1.00	

29	13	38	395	0.0000	1.00
28	CL39	46	580	0.0000	1.00
27	CL38	CL53	451	0.0001	1.00
26	CL49	30	735	0.0001	1.00
25	24	36	769	0.0001	.999
24	CL37	CL36	751	0.0001	.999
23	CL41	CL32	1108	0.0002	.999
22	CL34	51	413	0.0002	.999
21	CL44	CL48	624	0.0002	.999
20	CL33	37	521	0.0003	.999
19	CL40	CL28	954	0.0003	.998
18	14	CL31	1045	0.0005	.998
17	.	20	407	0.0006	.997
16	CL30	CL25	1532	0.0006	.997
15	12	CL24	1062	0.0007	.996
14	CL29	28	574	0.0008	.995
13	08	32	261	0.0009	.994
12	CL21	CL26	1359	0.0014	.993
11	CL23	CL27	1559	0.0021	.991
10	CL22	CL13	674	0.0032	.987
9	CL17	CL14	981	0.0043	.983
8	CL16	CL19	2486	0.0049	.978
7	CL12	CL15	2421	0.0054	.973
6	CL10	CL20	1195	0.0107	.962
5	CL8	CL18	3531	0.0159	.946
4	CL5	CL11	5090	0.0409	.905
3	CL6	CL7	3616	0.0722	.833
2	CL9	CL4	6071	0.1358	.697
1	CL2	CL3	9687	0.6973	.000




```
proc freq data=pmlr.pva_train_imputed noprint;
   tables CLUSTER_CODE*TARGET_B / chisq;
   output out=work.chi(keep=_pchi_) chisq;
run;
```

```
data work.cutoff;
   if  n  = 1 then set work.chi;
   set work.cluster;
   chisquare= pchi *rsquared;
   degfree=numberofclusters-1;
   logpvalue=logsdf('CHISQ',chisquare,degfree);
run;

title1 "Plot of the Log of the P-Value by Number of Clusters";
proc sgplot data=work.cutoff;
   scatter y=logpvalue x=numberofclusters
           / markerattrs=(color=blue symbol=circlefilled);
   xaxis label="Number of Clusters";
   yaxis label="Log of P-Value" min=-40 max=0;
run;
```




```
%global ex_ncl;
proc sql;
   select NumberOfClusters into :ex_ncl
   from work.cutoff
   having logpvalue=min(logpvalue);
quit;
```

Number of Clusters
4

```
proc tree data=work.fortree nclusters=&ex_ncl out=work.clus noprint;
   id cluster_code;
run;

proc sort data=work.clus;
   by clusname;
run;

title1 "Cluster Assignments";
proc print data=work.clus;
   by clusname;
   id clusname;
run;
```

Cluster Assignments

CLUSNAME	CLUSTER_CODE	CLUSTER
CL4	16	1
	25	1
	03	1
	05	1
	18	1
	23	1
	22	1
	53	1
	15	1
	26	1
	35	1
	17	1
	31	1
	02	1
	33	1
	11	1
	34	1
	42	1
	04	1
	50	1
	40	1
	27	1
	01	1
	46	1
	24	1
	36	1
	14	1

CL6	09	3
	52	3
	06	3
	10	3
	41	3
	51	3
	37	3
	08	3
	32	3
CL7	19	2
	45	2
	44	2
	47	2
	49	2
	07	2
	48	2
	21	2
	43	2
	29	2
	39	2
	30	2
	12	2
CL9	13	4
	38	4
	.	4
	20	4
	28	4

What is the optimum number of clusters for the **cluster_code** variable?

The optimum number of clusters for the CLUSTER_CODE variable is 4.

```
filename clcode "&PMLRfolder\cluster_code.sas";

data  null ;
   file clcode;
   set work.clus end=last;
   if _n_ = 1 then put "select (compress(cluster_code));";
   put "  when ('" cluster_code +(-1) "') cluster_clus = '" cluster
         +(-1) "';";
   if last then do;
      put "  otherwise cluster_clus = 'U';"/ "end;";
   end;
run;

data pmlr.pva_train_imputed_clus;
   set pmlr.pva train_imputed;
   %include clcode;
run;
```

3. Smoothed Weight of Evidence

a. Include the program **pmlr03e01.sas** or use the **PMLR_UpToDate** macro. Compute the smoothed weight of evidence for **cluster_code**. Use PROC SQL to compute the proportion of events in the training data set.

b. Use PROC MEANS to calculate the response rate and frequency in each of the levels of **cluster_code** and a DATA step to compute the smoothed weight of evidence for each **cluster_code** level.

c. Use a value of *24* for *c* and assign the overall logit to any observation with an undefined **cluster_code**.

d. Define the new variable **cluster_swoe** and put the new assignments in a data set called **pmlr.pva_train_imputed_swoe**.

Note: An electronic copy of the solution program is in **pmlr03s03.sas**.

```
%global rho1_ex;
proc sql noprint;
   select mean(target_b) into :rho1_ex
   from pmlr.pva_train_imputed;
run;

proc means data=pmlr.pva_train_imputed sum nway noprint;
   class cluster_code;
   var target_b;
   output out=work.counts sum=events;
run;

filename clswoe "&PMLRfolder\swoe_cluster.sas";

data _null_;
   file clswoe;
   set work.counts end=last;
     logit=log((events + &rho1_ex*24)/(_FREQ_ - events + (1-
           &rho1_ex)*24));
   if _n_=1 then put "select (cluster_code);" ;
   put "  when ('" cluster_code +(-1) "') cluster_swoe = " logit
";" ;
   if last then do;
      logit=log(&rho1_ex/(1-&rho1_ex));
      put "  otherwise cluster_swoe = " logit ";" / "end;";
   end;
run;

data pmlr.pva_train_imputed_swoe;
   set pmlr.pva_train_imputed;
   %include clswoe;
run;

title;
proc print data=pmlr.pva_train_imputed_swoe(obs=1);
   where cluster_code = "01";
   var cluster_code cluster_swoe;
run;
```

```
                                    CLUSTER_     cluster_
                              Obs     CODE         swoe

                              267      01        -0.98447
```

What is the value of the smoothed weight of evidence for **cluster_code 01**?
The smoothed weight of evidence for cluster_code 01 is -0.98447.

4. Variable Clustering

a. Include the program **pmlr03e03.sas** or use the **PMLR_UpToDate** macro. Use PROC VARCLUS to cluster all numeric variables in a hierarchical structure. (Use the macro variable **ex_inputs** with the missing indicator variables and the smoothed weight of evidence variable.)

b. Use MAXEIGEN=.70 as your stopping criterion.

c. Use the Output Delivery System to print out the table of the R-square statistics from the last iteration of PROC VARCLUS.

d. Print the table that shows the proportion of variation explained by the clusters.

Note: An electronic copy of the solution program is in **pmlr03s04.sas**.

```
ods select none;
ods output clusterquality=work.summary
           rsquare=work.clusters;

proc varclus data=pmlr.pva train imputed swoe hi maxeigen=0.70;
   var &ex_inputs mi_DONOR_AGE mi_INCOME_GROUP
       mi_WEALTH_RATING cluster_swoe;
run;

ods select all;

data  null ;
   set work.summary;
   call symput('nvar',compress(NumberOfClusters));
run;

title1 "Variables by Cluster";
proc print data=work.clusters noobs label split='*';
   where NumberOfClusters=&nvar;
   var Cluster Variable RSquareRatio;
   label RSquareRatio="1 - RSquare*Ratio";
run;

title1 "Variation Explained by Clusters";
proc print data=summary label;
run;
```

```
                                Variables by Cluster

                                                          1 - RSquare
                  Cluster       Variable                    Ratio
```

Cluster 1	MONTHS_SINCE_ORIGIN	0.1694
	LIFETIME_CARD_PROM	0.0964
	LIFETIME_PROM	0.1097
	LIFETIME_GIFT_AMOUNT	0.6593
	LIFETIME_GIFT_COUNT	0.4943
	MONTHS_SINCE_FIRST_GIFT	0.1536
	mi_WEALTH_RATING	0.5208
Cluster 2	RECENT_AVG_GIFT_AMT	0.4247
	RECENT_AVG_CARD_GIFT_AMT	0.6359
	LIFETIME_GIFT_RANGE	0.3966
	LIFETIME_MAX_GIFT_AMT	0.1463
	LAST_GIFT_AMT	0.4065
Cluster 3	MEDIAN_HOME_VALUE	0.2932
	MEDIAN_HOUSEHOLD_INCOME	0.2241
	PER_CAPITA_INCOME	0.1872
	nses1	0.5398
Cluster 4	FREQUENCY_STATUS_97NK	0.3979
	RECENT_RESPONSE_PROP	0.2056
	RECENT_CARD_RESPONSE_PROP	0.3633
	RECENT_RESPONSE_COUNT	0.2453
	RECENT_CARD_RESPONSE_COUNT	0.2223
Cluster 5	CARD_PROM_12	0.3716
	NUMBER_PROM_12	0.2871
	MONTHS_SINCE_LAST_GIFT	0.5233
Cluster 6	nses_	0.0000
	nurb_	0.0000
Cluster 7	mi_DONOR_AGE	0.3349
	mi_INCOME_GROUP	0.4338
Cluster 8	PCT_MALE_VETERANS	0.0000
Cluster 9	PCT_VIETNAM_VETERANS	0.3168
	PCT_WWII_VETERANS	0.3527
Cluster 10	LIFETIME_AVG_GIFT_AMT	0.3681
	LIFETIME_MIN_GIFT_AMT	0.1462
Cluster 11	nses3	0.2840
	cluster_swoe	0.3086
Cluster 12	PEP_STAR	0.3905
	STATUS_ES	0.3201
Cluster 13	PCT_MALE_MILITARY	0.0000
Cluster 14	nurbu	0.0000
Cluster 15	nurbt	0.0000
Cluster 16	home01	0.0000
Cluster 17	nurbr	0.0000
Cluster 18	DONOR_AGE	0.0000
Cluster 19	STATUS_FL	0.0000
Cluster 20	MOR_HIT_RATE	0.0000
Cluster 21	nses4	0.0000
Cluster 22	INCOME_GROUP	0.0000
Cluster 23	RECENT_STAR_STATUS	0.0000
Cluster 24	IN_HOUSE	0.0000
Cluster 25	WEALTH_RATING	0.0000
Cluster 26	PUBLISHED_PHONE	0.0000
Cluster 27	PCT_OWNER_OCCUPIED	0.0000
Cluster 28	nurbs	0.0000

Variation Explained by Clusters

Obs	Number of Clusters	Total Variation Explained by Clusters	Proportion of Variation Explained by Clusters	Minimum Proportion Explained by a Cluster	Maximum Second Eigenvalue in a Cluster	Minimum R-squared for a Variable	Maximum 1-R**2 Ratio for a Variable
1	1	7.932328	0.1497	0.1497	5.826522	0.0000	_
2	2	12.645612	0.2386	0.2177	4.131285	0.0000	1.0000
3	3	16.730624	0.3157	0.2603	3.075172	0.0007	1.0041
4	4	19.665398	0.3710	0.2603	2.436935	0.0005	1.0278
5	5	21.840212	0.4121	0.3119	1.949434	0.0005	1.0616
6	6	23.775930	0.4486	0.3119	1.795110	0.0005	1.1896
7	7	25.538296	0.4819	0.3119	1.486987	0.0005	1.3666
8	8	26.833836	0.5063	0.3576	1.416451	0.0005	1.2155
9	9	28.070094	0.5296	0.3576	1.303067	0.0005	1.2155
10	10	28.813747	0.5437	0.3576	1.089054	0.0020	1.2155
11	11	29.827100	0.5628	0.3576	1.076913	0.0020	1.2155
12	12	30.722383	0.5797	0.3576	1.017719	0.0024	1.2155
13	13	31.720428	0.5985	0.3576	1.003693	0.0045	1.2155
14	14	32.715792	0.6173	0.3576	0.998028	0.0045	1.2182
15	15	33.713820	0.6361	0.3576	0.990889	0.0590	1.2182
16	16	34.664584	0.6540	0.4465	0.987679	0.0590	0.9570
17	17	35.628268	0.6722	0.4465	0.958577	0.0590	0.9570
18	18	36.579098	0.6902	0.4465	0.923508	0.1678	0.9477
19	19	37.415611	0.7060	0.4465	0.917929	0.1736	0.8674
20	20	38.286771	0.7224	0.4604	0.888606	0.1736	0.8674
21	21	39.168115	0.7390	0.5729	0.855329	0.1736	0.8674
22	22	39.964918	0.7541	0.5833	0.853670	0.1736	0.8674
23	23	40.818171	0.7702	0.5833	0.787615	0.3566	0.7397
24	24	41.488014	0.7828	0.6105	0.780909	0.3566	0.7397
25	25	42.211853	0.7965	0.6105	0.778992	0.5030	0.6593
26	26	42.990846	0.8111	0.6312	0.737511	0.5030	0.6593
27	27	43.728356	0.8251	0.6380	0.724100	0.5030	0.6593
28	28	44.452456	0.8387	0.6676	0.687113	0.5030	0.6593

1) How many clusters were in your final solution?
The number of clusters in the final solution is 28.

2) What proportion of the variation was explained by the clusters?
The proportion of the variation that was explained by the clusters is 0.8387.

5. Variable Screening and Logit Plots

a. Include the program **pmlr03e03.sas** or use the **PMLR_UpToDate** macro. Use the Spearman and Hoeffding correlation coefficients to assess the inputs with the least evidence of a relationship with the target.

b. Use the **ex_reduced** macro variable in PROC CORR and use the LENGTH statement in the DATA step to specify a length of *32* for the character variable called **variable**.

c. Create a table with the Spearman rank of inputs and the Hoeffding rank of inputs and use PROC SGPLOT to create a scatter plot of the ranks of Spearman versus the ranks of Hoeffding.

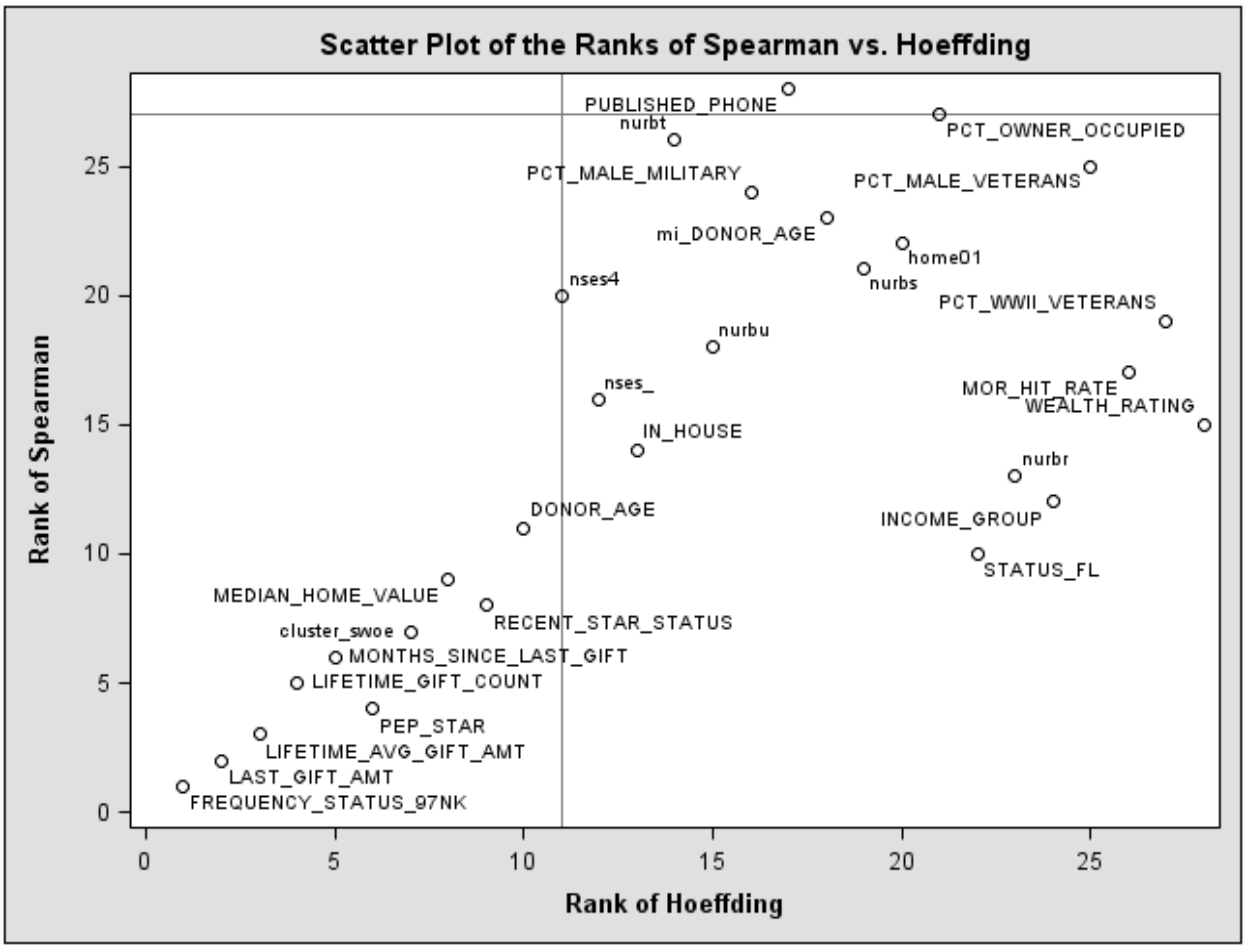
Note: An electronic copy of the solution program is in **pmlr03s05.sas**.

```
ods select none;
ods output spearmancorr=work.spearman
           hoeffdingcorr=work.hoeffding;

proc corr data=pmlr.pva_train_imputed_swoe spearman hoeffding;
   var target_b;
   with &ex_reduced;
run;

ods select all;

proc sort data=work.spearman;
    by variable;
run;

proc sort data=work.hoeffding;
    by variable;
run;

data work.correlations;
    length variable $ 32;
    merge work.spearman(rename=(target_b=scorr ptarget_b=spvalue))
          work.hoeffding(rename=(target_b=hcorr ptarget_b=hpvalue));
    by variable;
    scorr_abs=abs(scorr);
    hcorr_abs=abs(hcorr);
run;

proc rank data=work.correlations out=work.correlations1 descending;
    var scorr abs hcorr abs;
    ranks ranksp rankho;
run;

proc sort data=work.correlations1;
   by ranksp;
run;

title1 "Rank of Spearman Correlations and Hoeffding Correlations";
proc print data=work.correlations1 label split='*';
   var variable ranksp rankho scorr spvalue hcorr hpvalue;
   label ranksp = 'Spearman rank*of variables'
         scorr = 'Spearman Correlation'
         spvalue = 'Spearman p-value'
         rankho = 'Hoeffding rank*of variables'
         hcorr = 'Hoeffding Correlation'
         hpvalue = 'Hoeffding p-value';
run;
```

Rank of Spearman Correlations and Hoeffding Correlations

Obs	variable	Spearman rank of variables	Hoeffding rank of variables	Spearman Correlation	Spearman p-value	Hoeffding Correlation	Hoeffding p-value
1	FREQUENCY_STATUS_97NK	1	1	0.13777	<.0001	0.00213	<.0001
2	LAST_GIFT_AMT	2	2	-0.12345	<.0001	0.00197	<.0001
3	LIFETIME_AVG_GIFT_AMT	3	3	-0.11888	<.0001	0.00188	<.0001
4	PEP_STAR	4	6	0.11235	<.0001	0.00099	<.0001
5	LIFETIME_GIFT_COUNT	5	4	0.10943	<.0001	0.00156	<.0001
6	MONTHS_SINCE_LAST_GIFT	6	5	-0.09190	<.0001	0.00103	<.0001
7	cluster_swoe	7	7	0.08150	<.0001	0.00080	<.0001
8	RECENT_STAR_STATUS	8	9	0.07289	<.0001	0.00035	0.0006
9	MEDIAN_HOME_VALUE	9	8	0.06225	<.0001	0.00044	0.0001
10	STATUS_FL	10	22	-0.04935	<.0001	-0.00008	1.0000
11	DONOR_AGE	11	10	0.04266	<.0001	0.00022	0.0055
12	INCOME_GROUP	12	24	0.03859	0.0001	0.00008	0.0674
13	nurbr	13	23	-0.02987	0.0033	-0.00008	1.0000
14	IN_HOUSE	14	13	0.02963	0.0035	-0.00013	1.0000
15	WEALTH_RATING	15	28	0.02757	0.0066	-0.00001	0.4862
16	nses_	16	12	0.02538	0.0125	-0.00015	1.0000
17	MOR_HIT_RATE	17	26	0.02354	0.0205	-0.00006	0.9920
18	nurbu	18	15	-0.02238	0.0277	-0.00012	1.0000
19	PCT_WWII_VETERANS	19	27	0.01851	0.0685	-0.00004	0.8780
20	nses4	20	11	-0.01773	0.0810	-0.00015	1.0000
21	nurbs	21	19	0.01701	0.0942	-0.00011	1.0000
22	home01	22	20	0.01311	0.1970	-0.00010	1.0000
23	mi_DONOR_AGE	23	18	-0.01289	0.2048	-0.00012	1.0000
24	PCT_MALE_MILITARY	24	16	0.01256	0.2165	-0.00012	1.0000
25	PCT_MALE_VETERANS	25	25	0.01171	0.2493	-0.00008	1.0000
26	nurbt	26	14	0.01044	0.3043	-0.00013	1.0000
27	PCT_OWNER_OCCUPIED	27	21	0.00652	0.5211	-0.00010	1.0000
28	PUBLISHED_PHONE	28	17	-0.00268	0.7919	-0.00012	1.0000

```
%global vref href;
proc sql noprint;
   select min(ranksp) into :vref
   from (select ranksp
         from work.correlations1
         having spvalue > .5);

   select min(rankho) into :href
   from (select rankho
         from work.correlations1
         having hpvalue > .5);
quit;

title1 "Scatter Plot of the Ranks of Spearman vs. Hoeffding";
proc sgplot data=work.correlations1;
   refline &vref / axis=y;
   refline &href / axis=x;
   scatter y=ranksp x=rankho / datalabel=variable;
   yaxis label="Rank of Spearman";
   xaxis label="Rank of Hoeffding";
run;
```



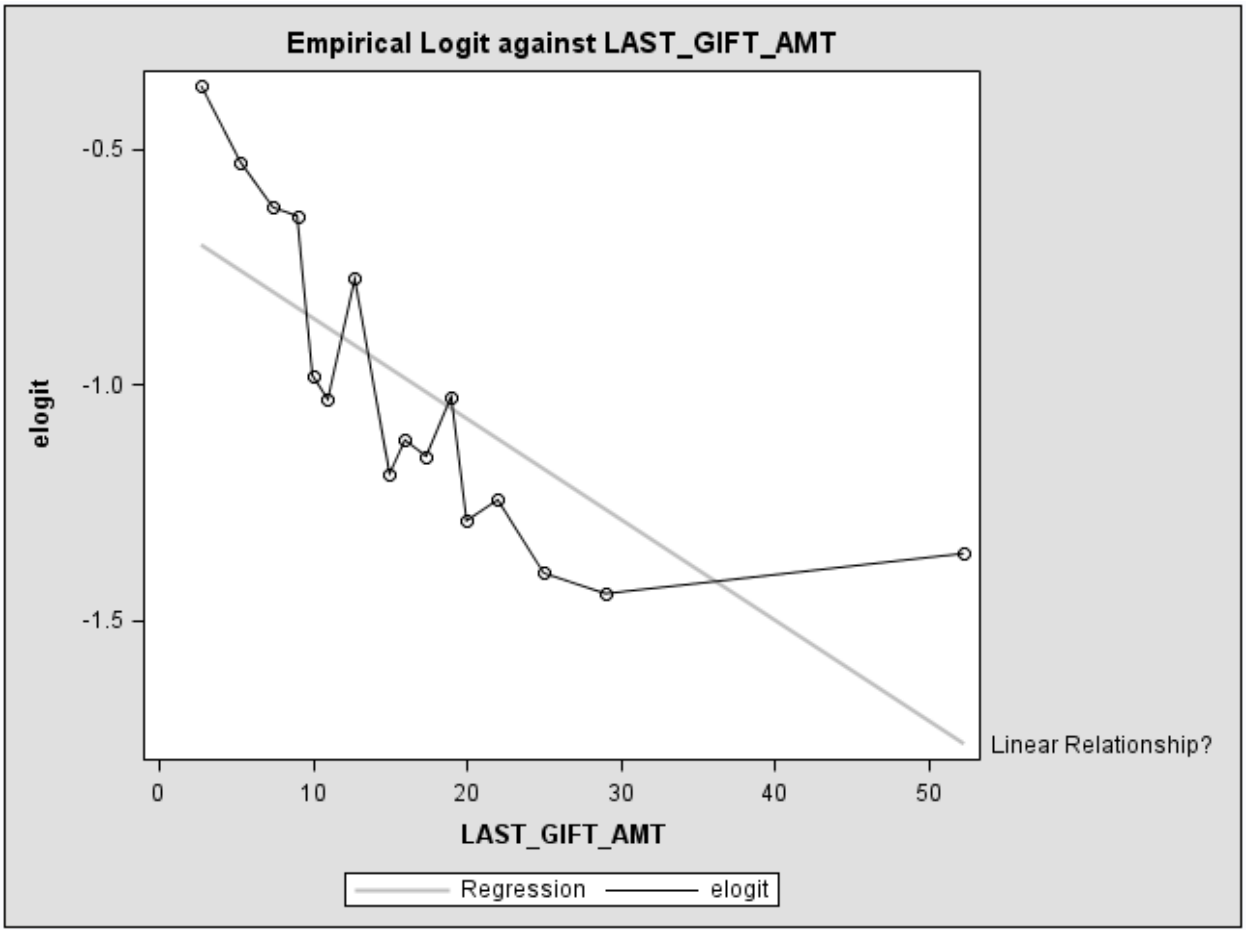
1) Which input variable shows evidence of a nonlinear relationship with the target?
None of the input variables shows evidence of a nonlinear relationship with the target.

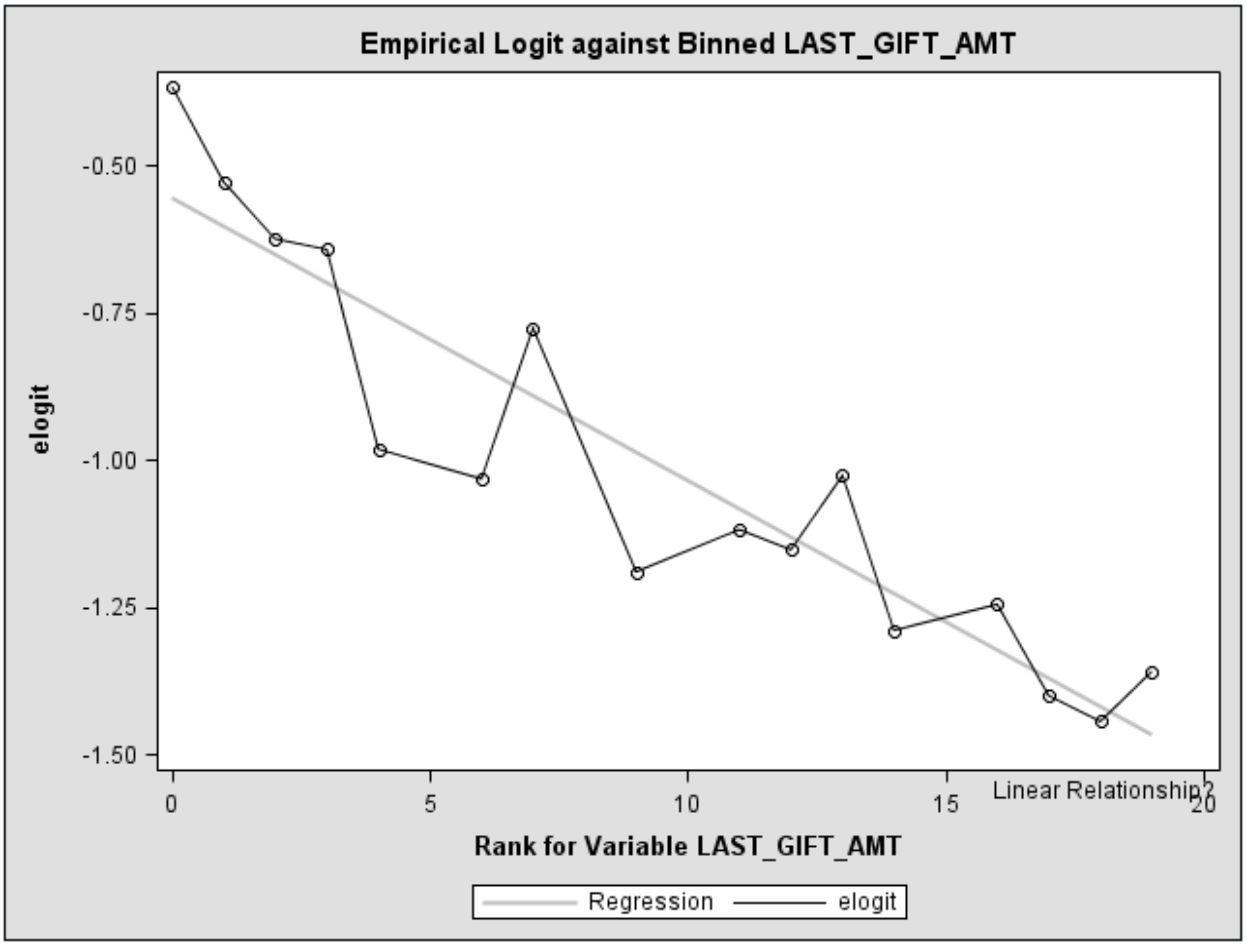
2) Which input variables can be eliminated due to irrelevancy?
The input variables PCT_OWNER_OCCUPIED and PUBLISHED_PHONE can be eliminated due to irrelevancy.

d. Create an empirical logit plot of **LAST_GIFT_AMT** and the bins of **LAST_GIFT_AMT**. Use 20 groups in PROC RANK.

```
%global var;
%let var = LAST GIFT AMT;
proc rank data=pmlr.pva_train_imputed_swoe groups=20 out=work.ranks;
   var &var;
   ranks bin;
run;

proc means data=work.ranks noprint nway;
   class bin;
   var target_b &var;
   output out=work.bins sum(target_b)=target_b
          mean(&var)=&var;
run;
```

```
data work.bins;
   set work.bins;
   elogit=log((target_b+(sqrt(_FREQ_ )/2))/
          ( _FREQ_ -target_b+(sqrt(_FREQ_ )/2)));
run;

title1 "Empirical Logit against &var";
proc sgplot data =work.bins;
   reg y=elogit x=&var /
       curvelabel="Linear Relationship?"
       curvelabelloc=outside
       lineattrs=(color=ligr);
   series y=elogit x=&var;
run;

title1 "Empirical Logit against Binned &var";
proc sgplot data=work.bins;
   reg y=elogit x=bin /
       curvelabel="Linear Relationship?"
       curvelabelloc=outside
       lineattrs=(color=ligr);
   series y=elogit x=bin;
run;
```





What is the relationship between **LAST_GIFT_AMT** and **target_b**?
The logit plots show evidence of a negative linear relationship.

6. Subset Selection Methods

a. Include the program pmlr03e03.sas or use the PMLR_UpToDate macro. Compute a BIC-based significance level using the sample size for n. Then use the forward selection method to detect important two-factor interactions. Use the INCLUDE option to include all 26 screened variables and use the macro variable **&ex_screened**. Using the NAMELEN= option in the PROC LOGISTIC statement, set the maximum length of effect names to 50.

Note: An electronic copy of the solution program is in **pmlr03s06.sas**.

```
%global sl ex screened ex selected;
title "P-Value for Entry and Retention";

proc sql;
   select 1-probchi(log(sum(target b ge 0)),1) into :sl
   from pmlr.pva_train_imputed_swoe;
quit;
```

```
P-Value for Entry and Retention
_______
0.002449
```

```
title1 "Interaction Detection using Forward Selection";
proc logistic data=pmlr.pva train imputed swoe namelen = 50;
   model target_b(event='1')= &ex_screened LIFETIME_GIFT_COUNT|
         LAST GIFT AMT| MEDIAN HOME VALUE|FREQUENCY STATUS 97NK|
         MONTHS_SINCE_LAST_GIFT|nses_|mi_DONOR_AGE|PCT_MALE_VETERANS|
         PCT_MALE_MILITARY|PCT_WWII_VETERANS|LIFETIME_AVG_GIFT_AMT|
         cluster swoe|PEP STAR|nurbu|nurbt|home01|nurbr|DONOR AGE|
         STATUS_FL|MOR_HIT_RATE|nses4|INCOME_GROUP|RECENT_STAR_STATUS|
         IN HOUSE|WEALTH RATING|nurbs @2 / include=26 clodds=pl
         selection=forward slentry=&sl;
run;
```

Partial Output

Summary of Forward Selection

Step	Effect Entered	DF	Number In	Score Chi-Square	Pr > ChiSq
1	LAST_GIFT_AMT*LIFETIME_AVG_GIFT_AMT	1	27	23.1980	<.0001
2	LIFETIME_AVG_GIFT_AMT*RECENT_STAR_STATUS	1	28	12.8362	0.0003
3	LIFETIME_GIFT_COUNT*MONTHS_SINCE_LAST_GIFT	1	29	10.1872	0.0014

Which interactions were detected?

The interactions were LAST_GIFT_AMT*LIFETIME_AVG_GIFT_AMT, LIFETIME_AVG_GIFT_AMT*RECENT_STAR_STATUS, and LIFETIME_GIFT_COUNT*MONTHS_SINCE_LAST_GIFT.

b. Use the backward elimination method in PROC LOGISTIC to find a subset of inputs. Use the screened variables and the interactions that are detected in the forward selection method. Specify the FAST option, the HIERARCHY=SINGLE option, the BIC-based significance level, and the profile likelihood confidence intervals.

```
title1 "Backward Selection for Variable Annuity Data Set";
proc logistic data=pmlr.pva_train_imputed_swoe namelen=50;
   model target b(event='1')= &ex screened
         LAST_GIFT_AMT*LIFETIME_AVG_GIFT_AMT
         LIFETIME AVG GIFT AMT*RECENT STAR STATUS
         LIFETIME_GIFT_COUNT*MONTHS_SINCE_LAST_GIFT
         / clodds=pl selection=backward slstay=&sl hier=single fast;
run;
```

Partial Output

Summary of Backward Elimination

Step	Effect Removed	DF	Number In	Wald Chi-Square	Pr > ChiSq
1	STATUS_FL	1	28	0.0003	0.9859
1	home01	1	27	0.0020	0.9645
1	nurbs	1	26	0.0289	0.8650
1	PCT_MALE_VETERANS	1	25	0.1600	0.6892
1	PCT_MALE_MILITARY	1	24	0.1767	0.6742
1	nurbt	1	23	0.3333	0.5637
1	nurbr	1	22	0.1894	0.6634

1	MOR_HIT_RATE	1	21	0.4537	0.5006
1	WEALTH_RATING	1	20	0.4980	0.4804
1	nses4	1	19	0.6128	0.4337
1	nurbu	1	18	0.4164	0.5187
1	mi_DONOR_AGE	1	17	1.4611	0.2268
1	IN_HOUSE	1	16	1.5824	0.2084
1	PCT_WWII_VETERANS	1	15	2.2087	0.1372
1	nses_	1	14	2.4131	0.1203
1	DONOR_AGE	1	13	5.1482	0.0233
1	LIFETIME_GIFT_COUNT*MONTHS_SINCE_LAST_GIFT	1	12	8.4930	0.0036
1	LIFETIME_GIFT_COUNT	1	11	1.6454	0.1996
1	LIFETIME_AVG_GIFT_AMT*RECENT_STAR_STATUS	1	10	8.6003	0.0034
1	RECENT_STAR_STATUS	1	9	0.2137	0.6439

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	0.1482	0.2498	0.3520	0.5530
LAST_GIFT_AMT	1	-0.0139	0.00426	10.6085	0.0011
MEDIAN_HOME_VALUE	1	0.000095	0.000026	13.4329	0.0002
FREQUENCY_STATUS_97NK	1	0.1565	0.0258	36.6681	<.0001
MONTHS_SINCE_LAST_GIFT	1	-0.0334	0.00620	29.1079	<.0001
LIFETIME_AVG_GIFT_AMT	1	-0.0138	0.00615	5.0581	0.0245
cluster_swoe	1	1.0032	0.1492	45.1805	<.0001
PEP_STAR	1	0.3191	0.0533	35.8039	<.0001
INCOME_GROUP	1	0.0484	0.0154	9.8491	0.0017
LAST_GIFT_AMT*LIFETIME_AVG_GIFT_AMT	1	0.000227	0.000061	13.9023	0.0002

Association of Predicted Probabilities and Observed Responses

Percent Concordant	63.1	Somers' D	0.262
Percent Discordant	36.9	Gamma	0.262
Percent Tied	0.0	Tau-a	0.098
Pairs	17595830	c	0.631

Odds Ratio Estimates and Profile-Likelihood Confidence Intervals

Effect	Unit	Estimate	95% Confidence Limits	
MEDIAN_HOME_VALUE	1.0000	1.000	1.000	1.000
FREQUENCY_STATUS_97NK	1.0000	1.169	1.112	1.230
MONTHS_SINCE_LAST_GIFT	1.0000	0.967	0.955	0.979
cluster_swoe	1.0000	2.727	2.037	3.656
PEP_STAR	1.0000	1.376	1.239	1.528
7. INCOME_GROUP	1.0000	1.050	1.018	1.082

How many inputs were in the final model?

There are nine inputs in the final model.

c. Generate fit statistics for the models that were generated in the all subsets selection. Use the **fitstat** macro to score each of the models.

```
%macro fitstat(data=,target=,event=,inputs=,best=,priorevent=);

ods select none;
ods output bestsubsets=work.score;

proc logistic data=&data namelen=50;
   model &target(event="&event")=&inputs / selection=score best=&best;
run;

proc sql noprint;
   select variablesinmodel into :inputs1 -
   from work.score;
   select NumberOfVariables into :ic1 -
   from work.score;
quit;

%let lastindx = &SQLOBS;

%do model_indx=1 %to &lastindx;

%let im=&&inputs&model_indx;
%let ic=&&ic&model_indx;

ods output scorefitstat=work.stat&ic;
proc logistic data=&data namelen=50;
   model &target(event="&event")=&im;
   score data=&data out=work.scored fitstat
         priorevent=&priorevent;
run;

proc datasets
   library=work
   nodetails
   nolist;
   delete scored;
run;
quit;

%end;

data work.modelfit;
   set work.stat1 - work.stat&lastindx;
   model = _n_;
run;

%mend fitstat;
```

```
%fitstat(data=pmlr.pva_train_imputed_swoe,target=target_b,event=1,
         inputs=&ex screened LAST GIFT AMT*LIFETIME_AVG_GIFT_AMT
         LIFETIME_AVG_GIFT_AMT*RECENT_STAR_STATUS
         LIFETIME GIFT COUNT*MONTHS_SINCE_LAST_GIFT,best=1,
         priorevent=0.05);

proc sort data = work.modelfit;
   by bic;
run;

title1 "Fit Statistics from Models selected from Best-Subsets";
ods select all;
proc print data=work.modelfit;
   var model auc aic bic misclass adjrsquare brierscore;
run;
```

Fit Statistics from Models selected from Best-Subsets

Obs	model	AUC	AIC	BIC	MisClass	Adj RSquare	Brier Score
1	9	0.631652	14779.62	14851.4	0.2498	0.063068	0.222593
2	10	0.632213	14774.72	14853.68	0.2498	0.063921	0.222532
3	8	0.630349	14789.23	14853.84	0.2499	0.061631	0.222709
4	11	0.632442	14771.15	14857.29	0.2498	0.064609	0.222463
5	12	0.632987	14764.68	14858.01	0.2497	0.065654	0.222331
6	7	0.628417	14802.91	14860.34	0.2499	0.059688	0.222876
7	13	0.633555	14760.16	14860.66	0.2497	0.066458	0.222274
8	6	0.626502	14814.34	14864.59	0.2499	0.058021	0.223026
9	14	0.633561	14759.11	14866.79	0.2497	0.066835	0.222234
10	5	0.625815	14826.13	14869.2	0.2500	0.056308	0.223329
11	15	0.633848	14758.26	14873.11	0.2497	0.067187	0.222202
12	4	0.622972	14844.22	14880.11	0.2500	0.053807	0.223532
13	16	0.634088	14758.3	14880.33	0.2497	0.067428	0.222179
14	17	0.63403	14758.47	14887.69	0.2497	0.067653	0.222161
15	18	0.634047	14759.89	14896.28	0.2497	0.067725	0.222152
16	19	0.634239	14761.15	14904.72	0.2497	0.067816	0.22215
17	3	0.619008	14879.2	14907.92	0.2500	0.049189	0.223944
18	20	0.634277	14762.5	14913.25	0.2497	0.067897	0.22214
19	21	0.634355	14763.95	14921.88	0.2497	0.067964	0.222137
20	22	0.63435	14765.75	14930.86	0.2497	0.067988	0.222135
21	23	0.634421	14767.34	14939.63	0.2497	0.068039	0.222132
22	24	0.634453	14769.14	14948.61	0.2497	0.068063	0.222131
23	25	0.634478	14770.93	14957.57	0.2497	0.06809	0.222128
24	2	0.607981	14937.08	14958.62	0.2500	0.041674	0.224492
25	26	0.634514	14772.88	14966.7	0.2497	0.068096	0.222126
26	27	0.634635	14774.81	14975.81	0.2497	0.068104	0.222149
27	28	0.63464	14776.78	14984.96	0.2497	0.068107	0.222149
28	29	0.63465	14778.78	14994.14	0.2497	0.068108	0.222149
29	1	0.586151	15028.27	15042.63	0.2500	0.029887	0.225337

What is the *c* statistic for the model selected by best subsets selection with the lowest BIC?

The c statistic for the model with the lowest BIC is 0.632.

```
proc sql;
   select VariablesInModel into :ex_selected
   from work.score
   where numberofvariables=9;
quit;
```

```
                                Variables Included in Model
_____________________________________________________________________________________________

LIFETIME_GIFT_COUNT LAST_GIFT_AMT MEDIAN_HOME_VALUE FREQUENCY_STATUS_97NK cluster_swoe PEP_STAR
INCOME_GROUP LAST_GIFT_AMT*LIFETIME_AVG_GIFT_AMT LIFETIME_GIFT_COUNT*MONTHS_SINCE_LAST_GIFT
```

End of Solutions

Solutions to Student Activities (Polls/Quizzes)

3.01 Multiple Choice Poll – Correct Answer

Which of the following statements is *true* regarding missing values in predictive modeling applications?

a. In complete case analysis, the loss of data is inconsequential when the missing values are spread across many variables.

b. Observations with missing values are scored in PROC LOGISTIC.

c. Missing indicator variables can be used to capture the relationship between the target variable and missing inputs.

d. The missing completely at random assumption is valid in most predictive modeling applications.

11

3.02 Multiple Choice Poll – Correct Answer

For the cell GRP_RESP=0 and GRP_AMT=0, what replaced the missing value for DONOR_AGE?

a. 57

b. 65

c. 58

d. 68

16

3.03 Multiple Choice Poll – Correct Answer

Which of the following statements is ***true*** regarding Greenacre's method for collapsing levels of contingency tables?

a. The levels are hierarchically clustered where the levels that give the largest reduction in the chi-square test of association are merged.
b. Levels with similar marginal response rates are merged.
c. The method does not account for the sample size in each level.
d. The method is appropriate for any categorical input.

3.04 Multiple Choice Poll – Correct Answer

What is the optimum number of clusters for the **cluster_code** variable?

a. 3
b. 4
c. 5
d. 6

3.05 Multiple Choice Poll – Correct Answer

How many clusters were in the final solution?

a. 28
b. 30
c. 32
d. 35

55

3.06 Multiple Choice Poll – Correct Answer

Which of the following statements is ***true*** regarding best subsets selection?

a. The models are ranked by the likelihood ratio chi-square.
b. The method uses a forward selection to find the best model.
c. The method is relatively efficient for a small number of variables (for example, fewer than 50).
d. The FAST option increases the efficiency of the best subsets selection.

78

3.07 Multiple Choice Poll – Correct Answer

What is the c statistic for the model selected by best subsets selection with the lowest BIC?

a. .610

b. .621

c. .628

d. .632

Chapter 4 Measuring Classifier Performance

4.1 Honest Assessment

Objectives

- Describe the purpose of comparing the training and validation data fit statistics with model complexity.
- Prepare the input variables on the validation data set.

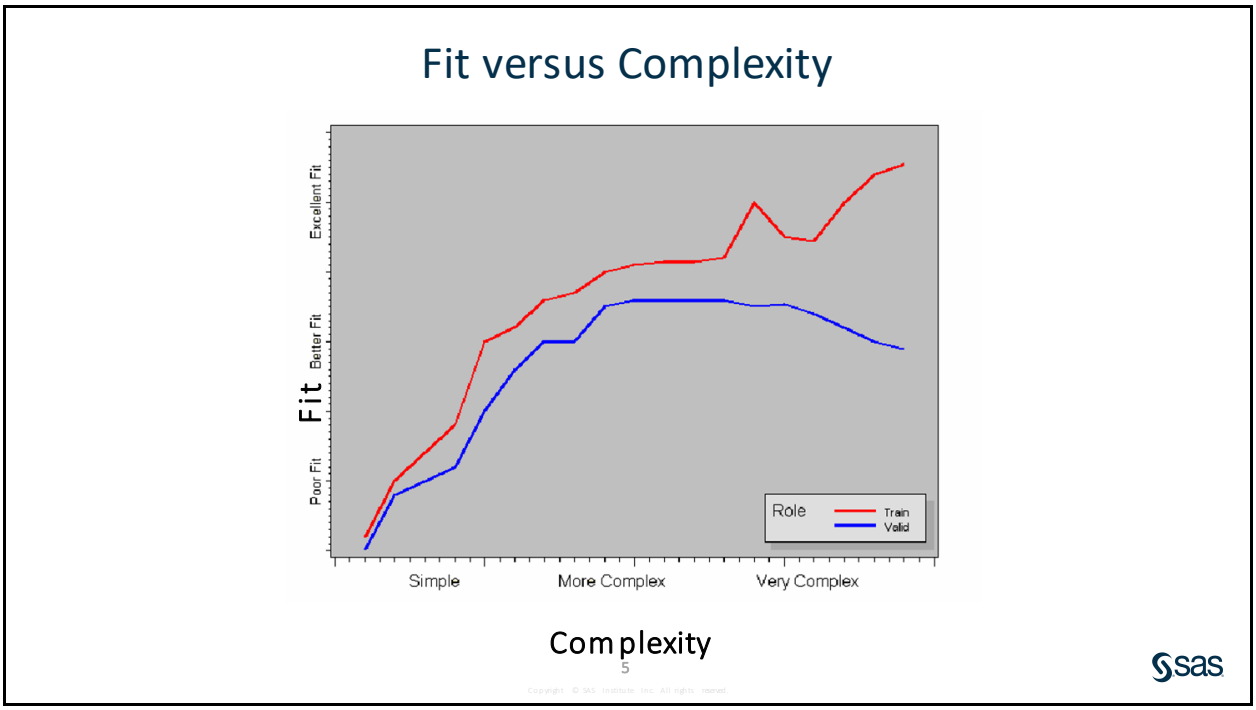
3

Fool's Gold

My model fits the training data perfectly. *I struck it rich!*

4

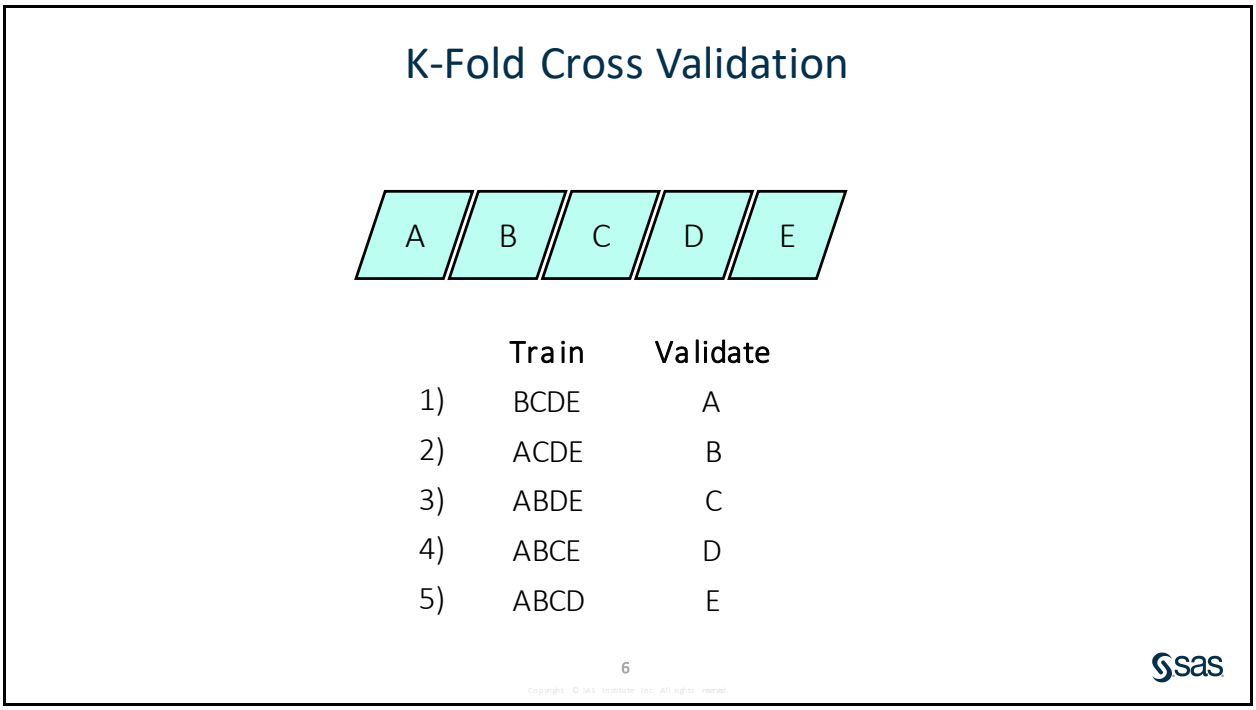
The purpose of predictive modeling is generalization, which is the performance of the predictions on new data. As was stated before, evaluating the model on the same data that the model was fit on usually leads to an optimistically biased assessment. The simplest strategy for correcting the optimism bias is data splitting, where a portion of the data is used to fit the model and the rest is removed for empirical validation.



To compare many models, an appropriate fit statistic (or statistics) must be selected. These statistics are typically measured on the validation data set. For a series of models, which could be generated by an automatic selection routine, it is possible to plot a fit measure against some index of complexity. For standard logistic regression models, this index is likely equivalent to the degrees of freedom in the model.

Typically, model performance follows a straightforward trend. As the complexity increases (that is, as terms are added), the fit on the training data improves. After a point, the fit might plateau, but on the training data, the fit gets better as model complexity increases. Some of this increase is attributable to the model capturing relevant trends in the data. Detecting these trends is the goal of modeling. Some of the increase, however, is due to the model identifying vagaries in the training data set. This behavior is called *overfitting*. Because these deviations are not likely to be repeated in the validation data or in future observations, it is reasonable to want to eliminate those models. Hence, the model fit on the validation data, for models of varying complexity, is also plotted. The typical behavior of the validation fit line is an increase (as more complex models detect more usable patterns) followed by a plateau, which might finally result in a decline in performance. The decline in performance is due to overfitting. The plateau indicates more complicated models that have no fit-based arguments for their use. A reasonable rule is to select the model that is associated with the complexity that has the highest validation fit statistic.

Plotting the training and validation results together permits a further assessment of the model's generalizing power. Typically, the performance deteriorates from the training data to the validation data. This phenomenon, sometimes known as *shrinkage*, is an additional statistic that some modelers use to obtain a measure of the generalizing power of the selected model. For example, the rule used to select a model from the many different plotted models might be the following: "Choose the simplest model that has the highest validation fit measure, with no more than 10% shrinkage from the training to the validation results."



Data splitting is a simple but costly technique. When the test set is small, the performance measures might be unreliable because of high variability. For small and moderate data sets, *k*-fold cross validation (Breiman et al. 1984; Ripley 1996; Hand 1997) is a better strategy. In five-fold cross validation, for example, the data is split into five equal sets. The entire modeling process is redone on each four-fifths of the data and uses the remaining one-fifth for assessment. The five assessments are then averaged. In this way, all the data are used for both training and assessment.

Another approach that is frugal with the data is to assess the model on the same data set that was used for training but to penalize the assessment for optimism (Ripley 1996). The appropriate penalty can be determined theoretically or by using computationally intensive methods such as the bootstrap.

Preparing the Validation Data

Example: Prepare the validation data set to be scored by the model fitted by the training data set.

The validation data is used for assessment. Consequently, it needs to be treated as if it were truly new data where the target variable value is unknown. The results of the analysis on the training data need to be applied to the validation data, not recalculated.

To assess the model generalization performance, the validation data need to be prepared for scoring the same way that the training data were prepared for model building. Missing values need to be imputed, new inputs need to be created, and any transformations need to be applied. PROC MEANS can be used to see which inputs need imputation on this validation data set.

```
/* pmlr04d01.sas */
title1 "Variables with Missing Values on the Validation Data Set";
proc means data = work.valid nmiss;
   var SavBal DDA CD Sav MM IRA IRABal ATMAmt ILS NSF SDB CCBal Inv
       DepAmt Dep ATM CC;
run;
```

Variables with Missing Values on the Validation Data Set

Variable	Label	N Miss
SavBal	Saving Balance	0
DDA	Checking Account	0
CD	Certificate of Deposit	0
Sav	Saving Account	0
MM	Money Market	0
IRA	Retirement Account	0
IRABal	IRA Balance	0
ATMAmt	ATM Withdrawal Amount	0
ILS	Installment Loan	0
NSF	Number Insufficient Fund	0
SDB	Safety Deposit Box	0
CCBal	Credit Card Balance	1350
Inv	Investment	1350
DepAmt	Amount Deposited	0
Dep	Checking Deposits	0
ATM	ATM	0
CC	Credit Card	1350

In the validation data set, missing values should be replaced with the medians from the training data set. Because the variables **CC**, **CCBal**, and **Inv** have missing values, PROC UNIVARIATE is used to create an output data set with the medians of those variables. The PCTLPTS option requests the 50th percentile, and the PCTLPRE option specifies the prefix for the variable names in the output data set.

```
proc univariate data=work.train_imputed_swoe_bins noprint;
   var cc ccbal inv;
   output out=work.medians pctlpts=50 pctlpre=cc ccbal inv;
run;
```

The DATA step first combines values from a single observation in one data set (**medians**) with all the observations in another data set (**valid_imputed_swoe_bins**). Array **X** contains the variables with missing values, and array **med** contains the variables with the medians. The DO loop replaces the missing values with the medians. The **branch_swoe** variable and the **B_DDABal** rank-transformed input are all added at this stage with the %INCLUDE statements.

```
data work.valid_imputed_swoe_bins(drop=cc50 ccbal50 inv50 i);
   if  N  = 1 then set work.medians;
   set work.valid;
   array x(*) cc ccbal inv;
   array med(*) cc50 ccbal50 inv50;
      do i = 1 to dim(x);
         if x(i)=. then x(i)=med(i);
      end;
   %include brswoe;
   if not dda then ddabal = &mean;
   %include rank;
run;
```

Now that the validation data are prepared for scoring, you could use any of the techniques discussed in Chapter 2 to score it. However, this prompts a question: What metrics can you use to measure model performance? This question and the related question of optimal use of a model are the focus of the next sections.

If you want to score new data, you need to examine the new data set for missing values and impute accordingly.

End of Demonstration

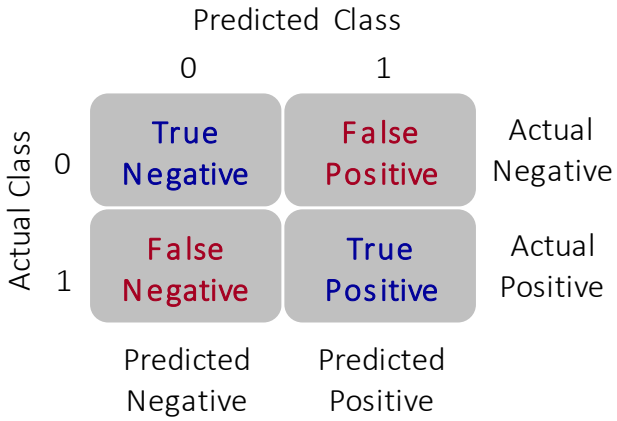
4.2 Misclassification

Objectives

- List several model performance measures.
- Adjust the confusion matrix for oversampling.
- Create ROC curves and lift charts on the validation data set.

11

Confusion Matrix



12

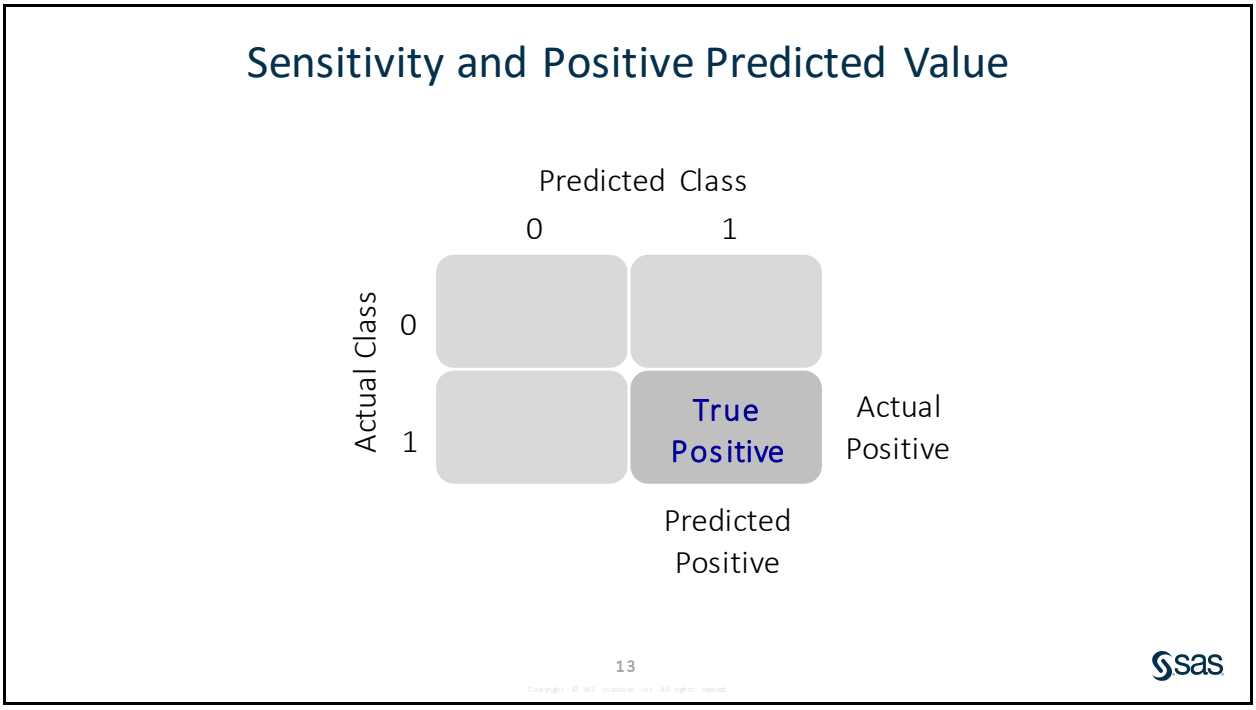
Supervised classification does not usually end with an estimate of the posterior probability. An allocation rule corresponds to a threshold value (cutoff) of the posterior probability. For example, all cases with probabilities of default greater than .04 might be rejected for a loan. For a given cutoff, how well does the classifier perform?

The fundamental assessment tool is the confusion matrix. The *confusion matrix* is a crosstabulation of the actual and predicted classes. It quantifies the confusion of the classifier. The event of interest, whether it is unfavorable (like fraud, churn, or default) or favorable (like response to offer), is often called a positive, although this convention is arbitrary. The simplest performance statistics are *accuracy*:

(true positives and negatives) / (total cases)

and *error rate*:

(false positives and negatives) / (total cases).



Two specialized measures of classifier performance are *sensitivity*:

(true positives) / (total actual positives)

and *positive predicted value* (PV+):

(true positives) / (total predicted positives)

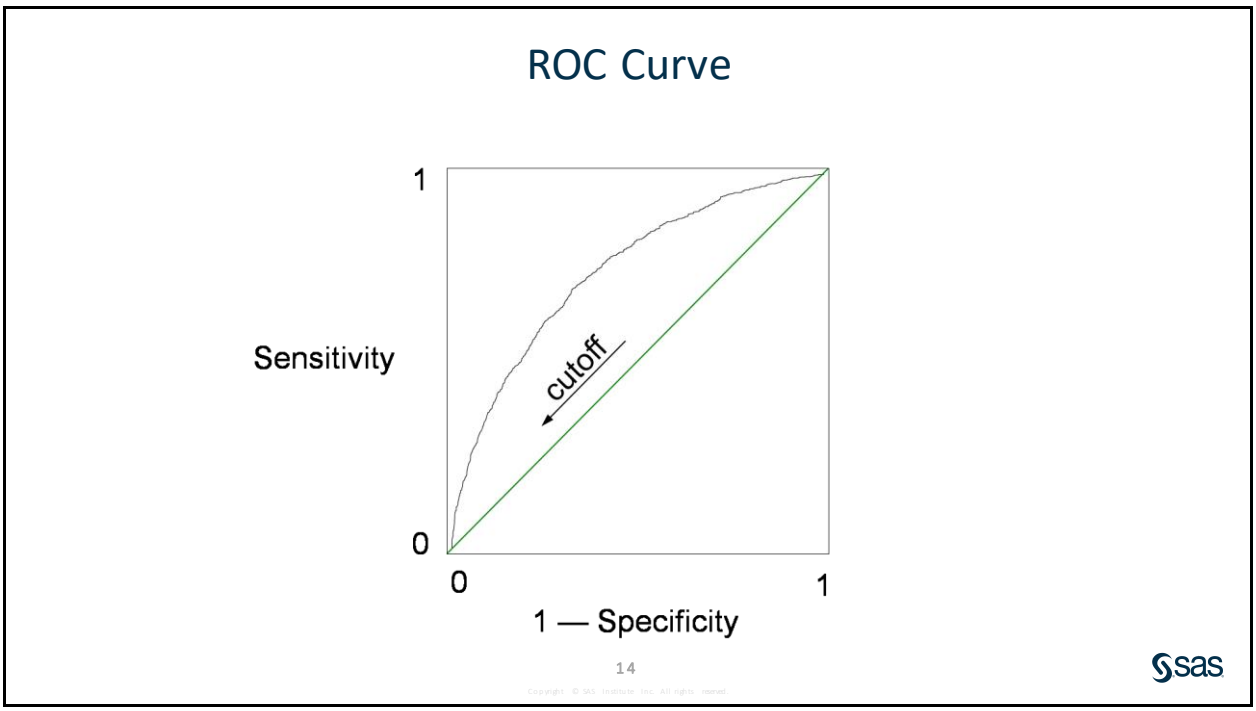
The analogs to these measures for true negatives are *specificity*:

(true negatives) / (total actual negatives)

and *negative predicted value* (PV–):

(true negatives) / (total predicted negatives).

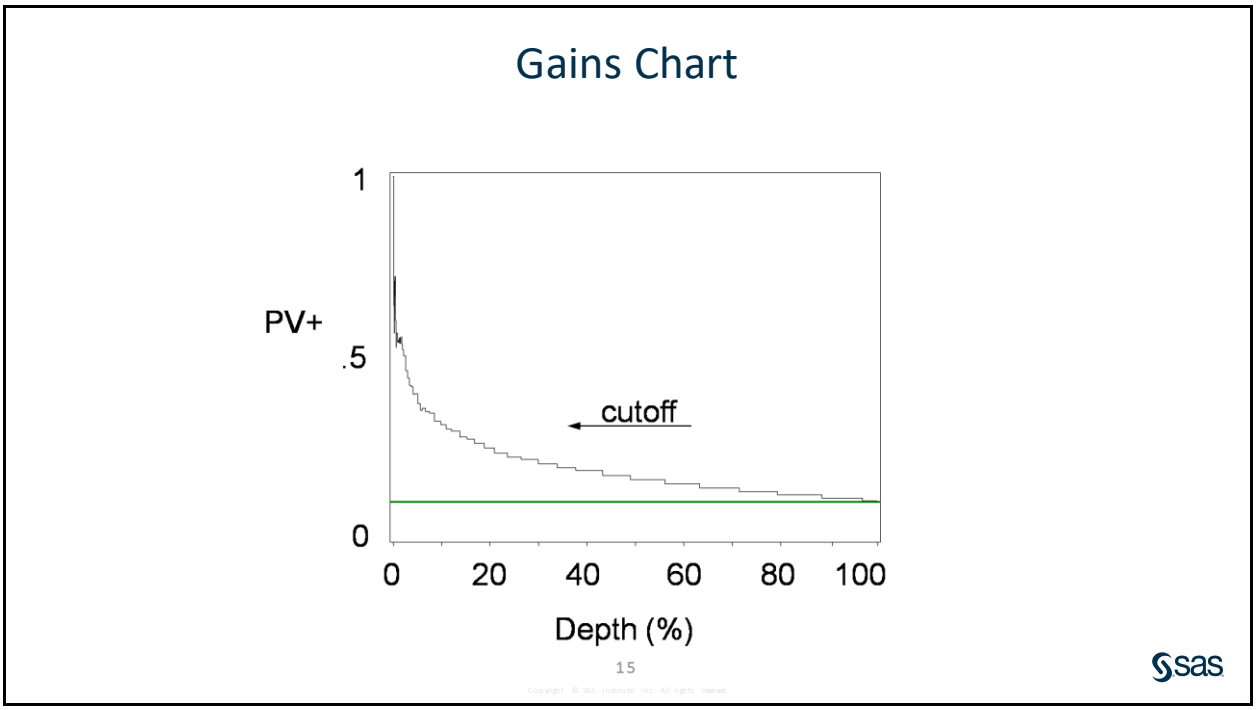
Large sensitivities do not necessarily correspond to large values of PV+. Ideally, you want large values for all these statistics. The context of the problem determines which of these measures is the primary concern. For example, a database marketer might be most concerned with PV+ because it gives the response rate for the customers that receive an offer. In contrast, a fraud investigator might be most concerned with sensitivity because it gives the proportion of frauds that would be detected.



The *receiver operating characteristic* (ROC) curve was adapted from signal detection theory for the assessment of classifiers. The ROC curve displays the sensitivity and specificity for the entire range of cutoff values.

- As the cutoff decreases, more cases are allocated to class 1. Hence, the sensitivity increases and specificity decreases.
- As the cutoff increases, more cases are allocated to class 0. Hence, the sensitivity decreases and specificity increases.

Consequently, the ROC curve intersects (0,0) and (1,1). If the posterior probabilities were arbitrarily assigned to the cases, then the ratio of false positives to true positives would be the same as the ratio of the total actual negatives to the total actual positives. Consequently, the baseline (random) model is a 45° angle going through the origin. As the ROC curve bows above the diagonal, the predictive power increases. A perfect model would reach the (0,1) point where both sensitivity and specificity equal 1.



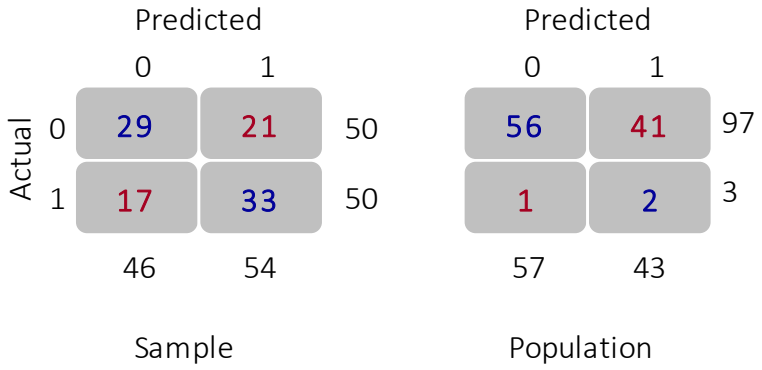
The *depth* of a classification rule is the total proportion of cases that were allocated to class 1. The (cumulative) *gains chart* displays the positive predicted value and depth for a range of cutoff values. As the cutoff decreases, more and more cases are allocated to class 1. Hence, the depth increases and the PV+ approaches the marginal event rate. When the cutoff is minimum, 100% of the cases are selected and the adjusted response rate is π_1. As the cutoff increases, the depth decreases. A model with good predictive power has increasing PV+ (response rate) as the depth decreases. If the posterior probabilities were arbitrarily assigned to the cases, then the gains chart is a horizontal line at π_1.

The gains chart is widely used in database marketing to decide how deep in a database to go with a promotion. The simplest way to construct this curve is to sort and bin the predicted posterior probabilities (for example, deciles). The gains chart is easily augmented with revenue and cost information.

The *lift* is $\frac{PV+}{\pi_1}$, so for a given depth, there are (lift) × more responders targeted by the model than by random chance.

A plot of sensitivity versus depth is sometimes called a *Lorenz* curve, *concentration* curve, or a *lift* curve (although lift value is not explicitly displayed).

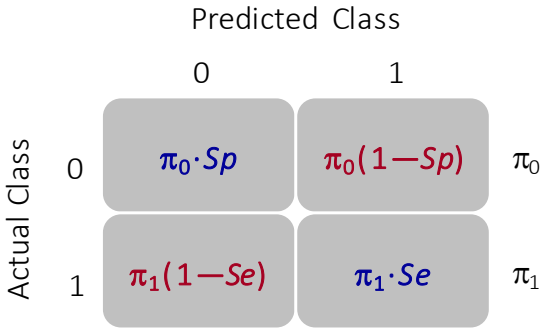
Oversampled Test Set



16

If the holdout data were obtained by splitting oversampled data, then they are oversampled as well. If the proper adjustments were made when the model was fitted, then the predicted posterior probabilities are correct. However, the confusion matrices are incorrect (regarding the population), because the event cases are over-represented. Consequently, PV+ (response rate) might be badly overestimated. Sensitivity and specificity, however, are not affected by separate sampling because they do not depend on the proportion of each class in the sample.

Adjustments for Oversampling



17

Knowing sensitivity, specificity, and the priors is sufficient for adjusting the confusion matrix for oversampling. For example, if the sample represented the population, then $n\pi_1$ cases are in class 1. The proportion of those that were allocated to class 1 is *Se*. Thus, there are $n\pi_1 \cdot Se$ true positives. Notice that these adjustments are equivalent to multiplying the cell counts by their sample weights. See the example below.

$$\mathrm{TP}_{\mathrm{sample}} \cdot wt_1 = \mathrm{TP}_{\mathrm{sample}} \frac{\pi_1}{\rho_1} = \mathrm{TP}_{\mathrm{sample}} \cdot \pi_1 \frac{n}{\mathrm{Tot\,Pos}_{\mathrm{sample}}} = n \cdot \pi_1 \cdot Se$$

where **TP** is the proportion of true positives, and sample weights are defined as π_i / ρ_i for class *i*.

4.01 Multiple Choice Poll

What is the formula for sensitivity?

a. true positives / total actual positives
b. true positives / total predicted positives
c. true negatives / total actual negatives
d. true negatives / total predicted negatives

Assessing Classifier Performance

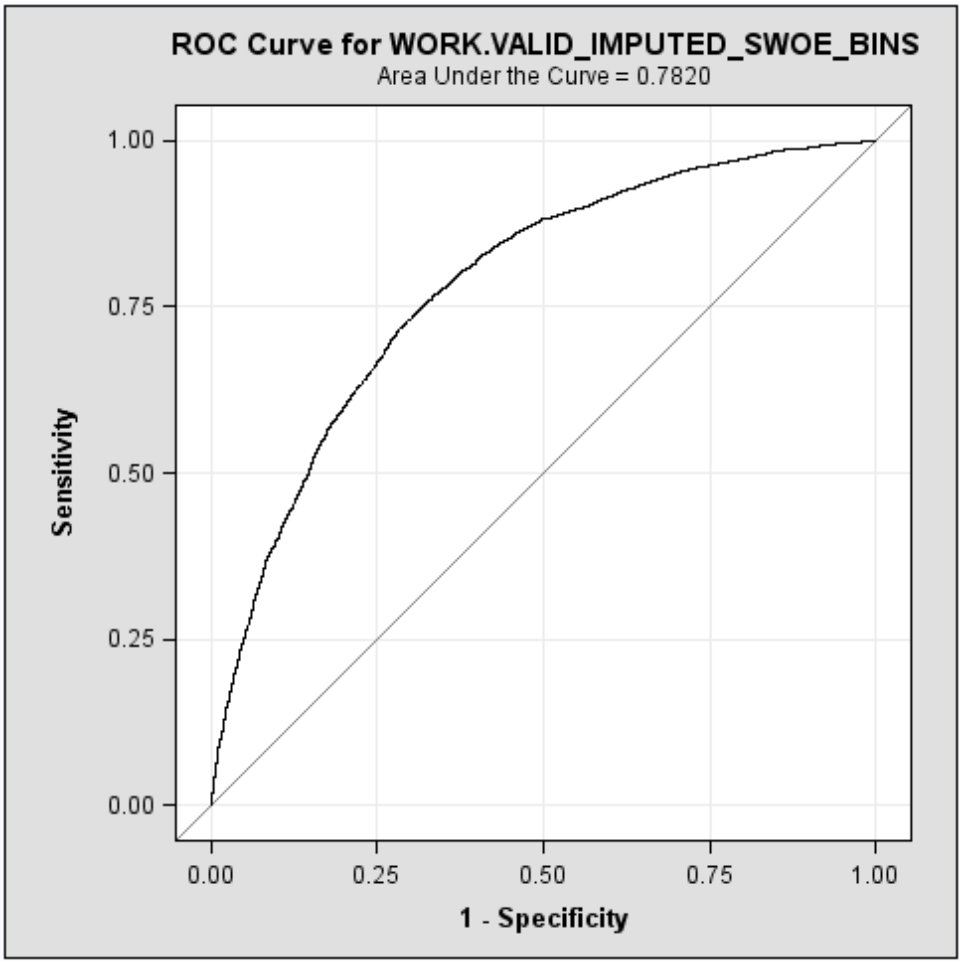
Example: Calculate performance measures on the validation data set. Use the SCORE statement in PROC LOGISTIC to score the validation data set with the model fit on the training data set. Use the OUTROC= option to create a data set with many of the statistics that are necessary for assessment. When you specify the OUTROC= option in the SCORE statement, the ROC curve for the scored data set is displayed. Use the FITSTAT option to generate model fit statistics.

The OUTROC= option creates an output data set with sensitivity (**_SENSIT_**) and one minus specificity (**_1MSPEC_**) calculated for a full range of cutoff probabilities (**_PROB_**). The other statistics in the OUTROC= data set are not useful when the data are oversampled. The two variables **_SENSIT_** and **_1MSPEC_** in the OUTROC= data set are correct whether the validation data are oversampled. The variable **_PROB_** is correct, provided the PRIOREVENT= was set to π_1. If the variables were not corrected, then **_PROB_** needs to be adjusted using the following formula:

$$\hat{p}_i = \frac{\hat{p}_i^* \rho_0 \pi_1}{(1-\hat{p}_i^*)\rho_1 \pi_0 + \hat{p}_i^* \rho_0 \pi_1}$$

where $\hat{p}_i^*$ is the unadjusted estimate of the posterior probability (**_PROB_**). The **scoval** (scored validation) data set is used later.

```
/* pmlr04d02.sas */

ods select roccurve scorefitstat;
proc logistic data=work.train_imputed_swoe_bins;
   model ins(event='1')=&selected;
   score data=work.valid_imputed_swoe_bins out=work.scoval
         priorevent=&pi1 outroc=work.roc fitstat;
run;
```



Fit Statistics for SCORE Data

Data Set	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC
WORK.VALID_IMPUTED_SWOE_BINS	10752	-12661.1	0.3406	25394.13	25394.38	25656.31

Data Set	SC	R-Square	Max-Rescaled R-Square	AUC	Brier Score
WORK.VALID_IMPUTED_SWOE_BINS	25656.31	0.316954	0.338919	0.78197	0.308581

The *c* statistic for the training data set was 0.788, which is not much higher than 0.782. This small difference in the *c* statistics between the training and validation data sets indicates that this model can generalize very well to new data.

```
title1 "Statistics in the ROC Data Set";
proc print data=roc(obs=10);
   var _prob_ _sensit_ _1mspec_;
run;
```

Statistics in the ROC Data Set

Obs	_PROB_	_SENSIT_	_1MSPEC_
1	1.00000	.000537057	.000000000
2	1.00000	.000805585	.000000000
3	1.00000	.001074114	.000000000
4	0.99999	.001342642	.000000000
5	0.99997	.001611171	.000000000
6	0.99948	.001879699	.000000000
7	0.99896	.002148228	.000000000
8	0.99890	.002416756	.000000000
9	0.99875	.002416756	.000142288
10	0.99823	.002416756	.000284576

Knowledge of the population priors, sensitivity, and specificity is sufficient to fill in the confusion matrices. Several additional statistics can be calculated in a DATA step.

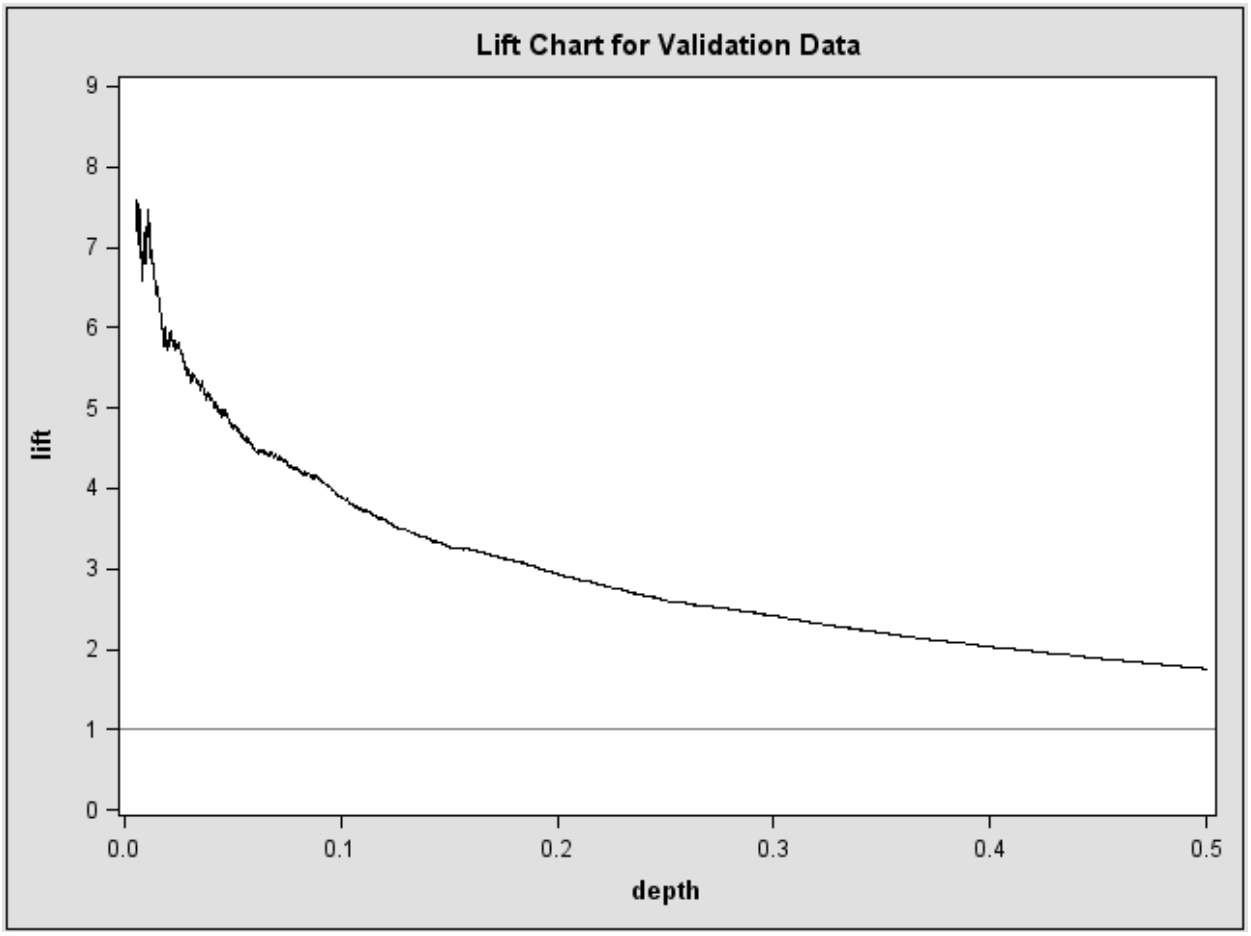
- TP = proportion of true positives
- FN = proportion of false negatives
- TN = proportion of true negatives
- FP = proportion of false positives
- POSPV = positive predicted value
- NEGPV = negative predicted value
- ACC = accuracy
- DEPTH = proportion allocated to class 1
- LIFT = positive predicted value/π_1

Each row in the OUTROC= data set corresponds to a cutoff (**_PROB_**). The selected cutoffs occur where values of the estimated posterior probability change, provided the posterior probabilities are more than .0001 apart. Otherwise, they are grouped. Consequently, the maximum number of rows in the OUTROC= data set is 9999, but it is usually much less. The grouping can be made coarser by using the ROCEPS= option in the MODEL statement.

```
data work.roc;
   set work.roc;
   cutoff=_PROB_;
   specif=1- 1MSPEC ;
   tp=&pi1*_SENSIT_;
   fn=&pi1*(1- SENSIT );
   tn=(1-&pi1)*specif;
   fp=(1-&pi1)*_1MSPEC_;
   depth=tp+fp;
   pospv=tp/depth;
   negpv=tn/(1-depth);
   acc=tp+tn;
   lift=pospv/&pi1;
   keep cutoff tn fp fn tp
         SENSIT   1MSPEC  specif depth
        pospv negpv acc lift;
run;
```

Example: Create a lift chart by plotting **lift** against **depth**. Add a reference line at the baseline (a lift of 1). Restrict the focus to the region where depth is greater than 0.5% (less erratic) and less than 50%.

```
title1 "Lift Chart for Validation Data";
proc sgplot data=work.roc;
   where 0.005 <= depth <= 0.50;
   series y=lift x=depth;
   refline 1.0 / axis=y;
   yaxis values=(0 to 9 by 1);
run; quit;
```



The lift chart shows that at a depth of 0.1, the lift is approximately 4.0. This means that if you target the top 10% of your customers based on the predicted probabilities, you get four times as many responses compared to targeting a random sample of 10%.

End of Demonstration

Exercises

1. Model Assessment

a. Include the program **pmlr03e03.sas** or use the **PMLR_UpToDate** macro. Prepare the validation data set to be scored by the model fitted by the training data set.

1) Use PROC MEANS to examine which variables in the validation data set have missing values. Use the inputs from the model fitted by the training data set.

2) Use PROC UNIVARIATE to create a data set with the medians from the training data set of the variables with missing values.

3) Use the DATA step to impute the variables with missing values and to include the scoring code to create the smoothed weight of evidence for **cluster_code**.

4) Which input variable has missing values?

b. Fit a logistic regression model on the training data set using **TARGET_B** as the target variable and the macro variable **EX_SELECTED** as the input variables.

1) Use the EVENT= option to model the probability that **TARGET_B**=1.

2) Use the SCORE statement to score the validation data set and adjust for oversampling.

3) Use the OUTROC= option to create a data set and an ROC curve for the validation data set.

4) Use the FITSTAT option to generate model fit statistics.

5) What is the *c* statistic for the validation data set?

c. Use the OUTROC= data set. Write a DATA step to compute the proportion of true positives, the proportion of false negatives, the proportion of true negatives, the proportion of false positives, the positive predicted value, the negative predicted value, the accuracy, the proportion allocated to class 1 (depth), and the lift.

1) Use PROC SGPLOT to create a lift chart.

2) Add a reference line at a lift of 1 and restrict the focus to the region where depth is greater than 0.5% and less than 50%.

3) Restrict the Y axis from 0 to 4 by 1.

4) What is the lift at a depth of 10%?

End of Exercises

4.02 Multiple Choice Poll

What is the lift at a depth of 10%?

a. 1.5
b. 1.9
c. 2.3
d. 2.7

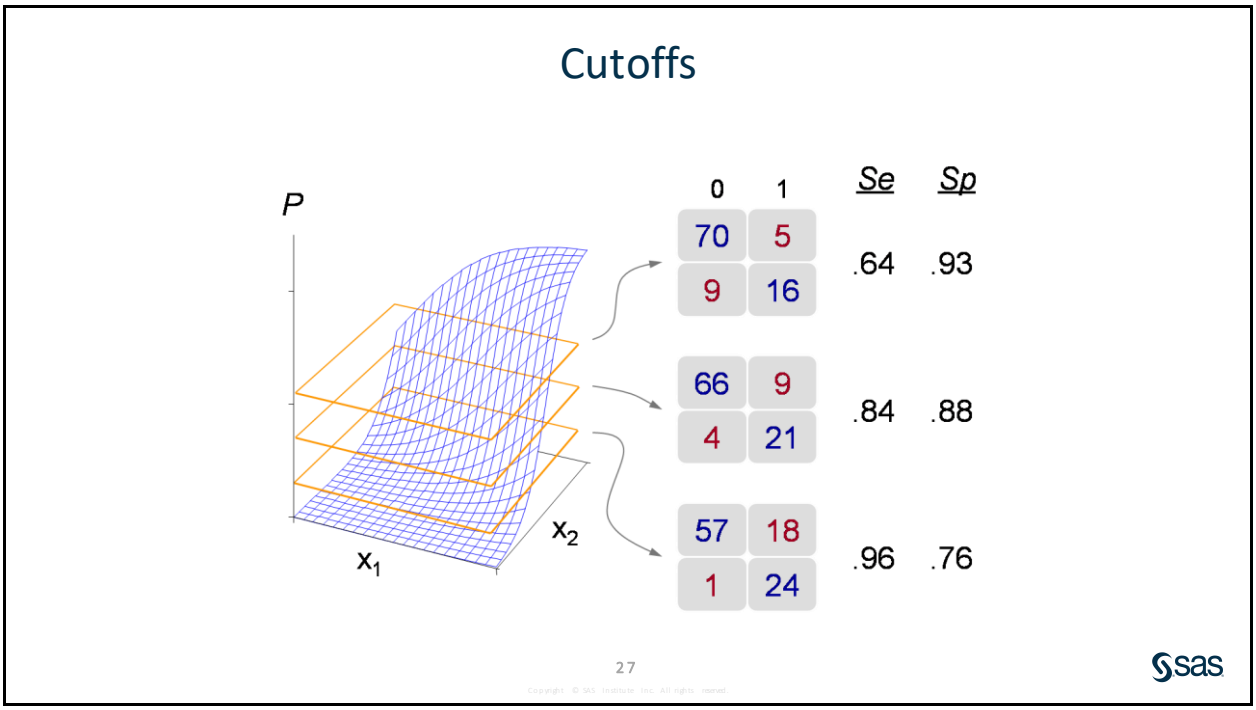
22

4.3 Allocation Rules

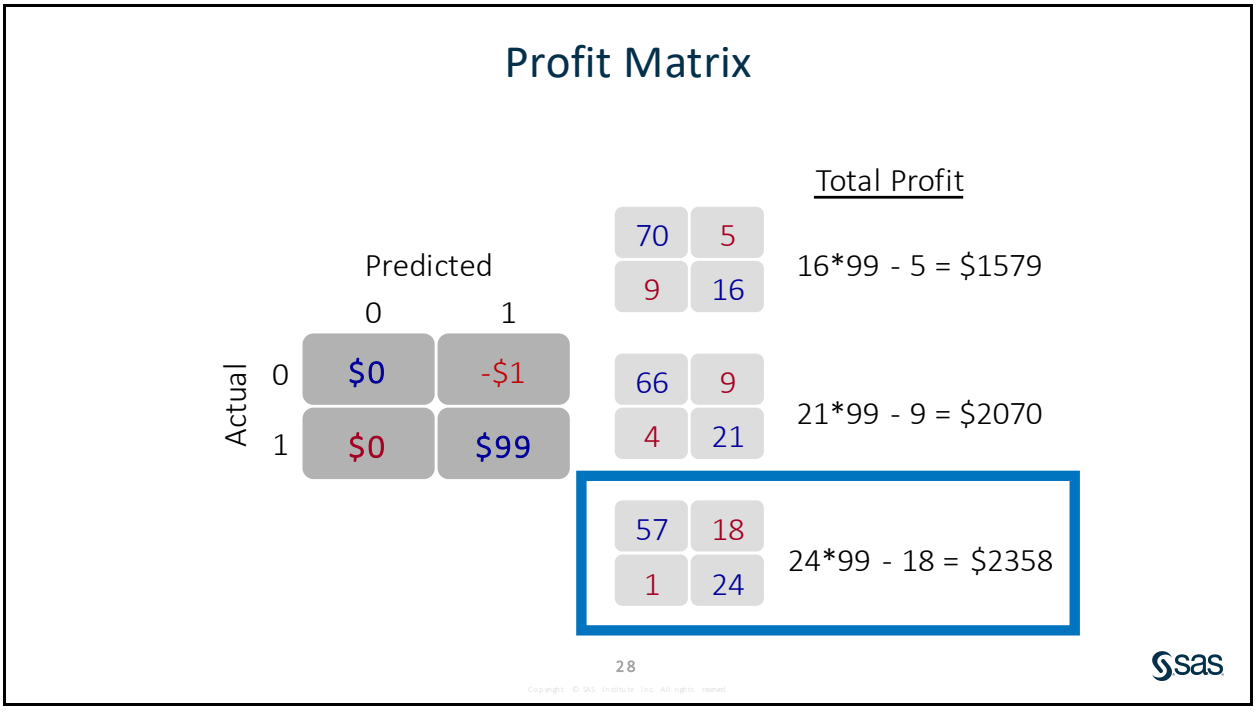
Objectives

- Describe the decision rule that maximizes the expected profit.
- Explain the profit matrix and how to use it to estimate the profit per scored customer.
- Graph the average validation profit against different cutoffs and different depths.

26



Different cutoffs produce different allocations and different confusion matrices. To determine the optimal cutoff, a performance criterion needs to be defined. If the goal were to increase the sensitivity of the classifier, then the optimal classifier would allocate all cases to class 1. If the goal were to increase specificity, then the optimal classifier would be to allocate all cases to class 0. For realistic data, there is a trade-off between sensitivity and specificity. Higher cutoffs decrease sensitivity and increase specificity. Lower cutoffs decrease specificity and increase sensitivity.



A formal approach to determining the optimal cutoff uses statistical decision theory (McLachlan 1992, Ripley 1996, Hand 1997). The decision-theoretic approach starts by assigning profit margins to true positives and loss margins to false positives. The optimal decision rule maximizes the total expected profit.

The profit matrix in the slide is consistent with a marketing effort that costs \$1, and when successful, garners revenue of \$100. Hence, the profit for soliciting a non-responder is -\$1, and the profit for soliciting a responder is \$100-\$1 = \$99. Given that each individual has a posterior probability p_i, you can use to simple algebra to find the optimum cutoff.

A typical decision rule is as follows: Solicit if the expected profit for soliciting, given the posterior probability, is higher than the expected profit for ignoring the customer.

Solicit if…:

E(Profit | p_i, solicit)	>	E(Profit | p_i, do not solicit)
p_i*99 + (1-p_i)*(-1)	>	p_i*0 + (1-p_i)*(0)
99*p_i -1 + p_i	>	0
100*p_i -1	>	0
p_i	>	0.01.

This cutoff of 0.01 can be used to calculate the expected profit of using this rule with the current model.

Profit Matrix

Decision

Actual Class	0	1
0	δ_{TN}	δ_{FP}
1	δ_{FN}	δ_{TP}

Bayes Rule:

Decision 1 if

$$P > \frac{1}{1+\left(\frac{\delta_{TP}-\delta_{FN}}{\delta_{TN}-\delta_{FP}}\right)}$$

29

The *Bayes rule* is the decision rule that maximizes the expected profit. In the two-class situation, the Bayes rule can be determined analytically. Using the symbols in the profit matrix above, if a customer is solicited, then the expected profit is as follows:

$$p(\delta_{TP}) + (1-p)(\delta_{FP})$$

where p is the true posterior probability that a case belongs to class 1.

If a customer is not solicited, then the expected profit is the following:

$$p(\delta_{FN}) + (1-p)(\delta_{TN})$$

Therefore, the optimal rule allocates a case to class 1 if the following is true:

$$p(\delta_{TP}) + (1-p)\,(\delta_{FP}) > p(\delta_{FN}) + (1-p)(\delta_{TN})$$

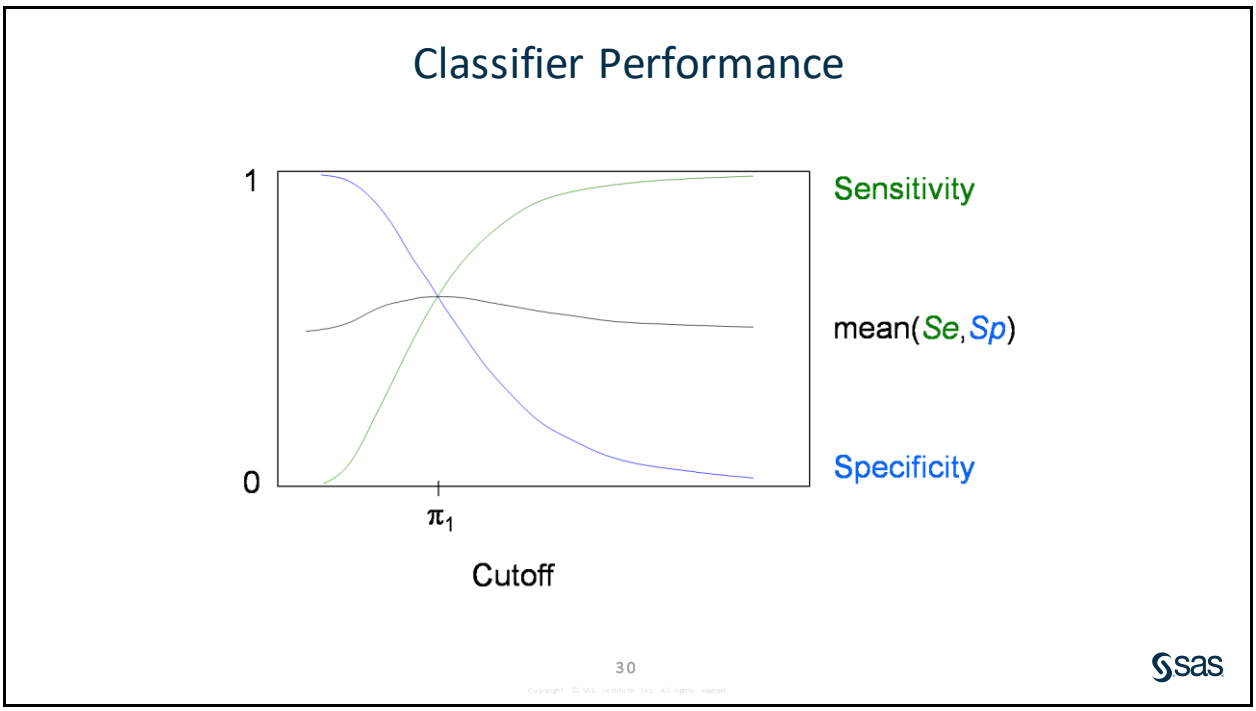
Otherwise, allocate the case to class 0. Solving for p gives the optimal cutoff probability. Because p must be estimated from the data, the *plug-in Bayes rule* is used in practice.

$$\hat{p} > \frac{1}{1+\left(\frac{\delta_{TP}-\delta_{FN}}{\delta_{TN}-\delta_{FP}}\right)}$$

Consequently, the plug-in Bayes rule might not achieve the maximum profit, if the estimate of the posterior probability is poorly estimated.

Using the profit matrix defined earlier, the optimal cutoff becomes the following:

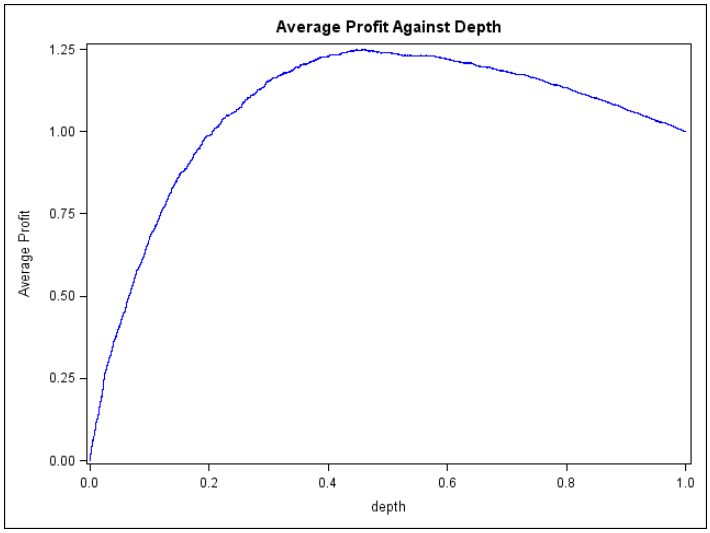
$$\hat{p} > \frac{1}{1+\left(\frac{99-0}{0-(-1)}\right)} \text{ or } 0.01$$



In many situations, gathering profit information can be difficult. One recommendation is to use a cutoff of π_1. The central cutoff, π_1, tends to maximize the mean of sensitivity and specificity. Because increasing sensitivity usually corresponds to decreasing specificity, the central cutoff tends to equalize sensitivity and specificity.

If separate samples were taken with equal allocation (50% events and 50% nonevents), then using the unadjusted cutoff of 0.5 on the biased sample is equivalent to using the central cutoff, π_1, on the population.

Using Profit to Assess Fit



31

Defining a *profit matrix* can create a useful statistic for measuring classifier performance. The model yields posterior probabilities. Those probabilities (in conjunction with a profit matrix) classify individuals into likely positives and likely negatives. On the validation data, the behavior of these individuals is known. Hence, it is feasible to calculate each individual's expected profit, and hence, it is also feasible to calculate a total profit. This total profit can be used as a model selection and assessment criterion.

If profit information can be used, it permits a more familiar scale for comparing models. Consumers of models might not have a feel for what type of lift they expect, or what constitutes a good value for sensitivity. Using total or average profit as an assessment statistic might avoid those issues.

4.03 Multiple Choice Poll

If the profit margin of true positives is nine times higher than the loss margins of false positives, then according to the Bayes rule, what is the cutoff that maximizes the expected profit (assuming 0 profit and loss for true negatives and false negatives)?

a. 0.90
b. 0.09
c. 1/9
d. 0.10

Using Profit to Assess Fit

Example: Use a cutoff of 0.01 and the profit matrix of -$1 for soliciting a non-responder and $99 for soliciting a responder to calculate profit per individual. Use PROC MEANS to sum and average the profits and use the WEIGHT statement to account for oversampling.

To calculate total and average profit comparable to what is achieved in the population, weights must be calculated. Sampling weights adjust the data so that it better represents the true population. When a rare target event is oversampled, class 0 is under-represented in the sample. Consequently, a class-0 case should count more in the analysis than a class-1 case.

$$weight_i = \begin{cases} \dfrac{\pi_1}{\rho_1} & \text{if } y_i = 1 \\ \dfrac{\pi_0}{\rho_0} & \text{if } y_i = 0 \end{cases}$$

The weights adjust the number of cases in the sample to be $n\pi_0$ and $n\pi_1$, in class 0 and 1 respectively. The classes now are in the same proportion in the adjusted sample as they are in the population.

The macro variables for the proportion of events in the population and sample were created earlier. The decision variable is created as a flag. It indicates whether the predicted probability is greater than the cutoff, 0.01. Using the information about decision and response, the profit per individual is calculated.

```
/* pmlr04d03.sas */

%global rho1;
proc SQL noprint;
  select mean(INS) into :rho1 from pmlr.develop;
quit;

data work.scoval;
   set work.scoval;
   sampwt = (&pi1/&rho1)*(INS)
            + ((1-&pi1)/(1-&rho1))*(1-INS);
   decision = (p 1 > 0.01);
   profit = decision*INS*99
            - decision*(1-INS)*1;
run;

title1 "Total and Average Profit";
proc means data=work.scoval sum mean;
   weight sampwt;
   var profit;
run;
```

Total and Average Profit

The MEANS Procedure

Analysis Variable : profit

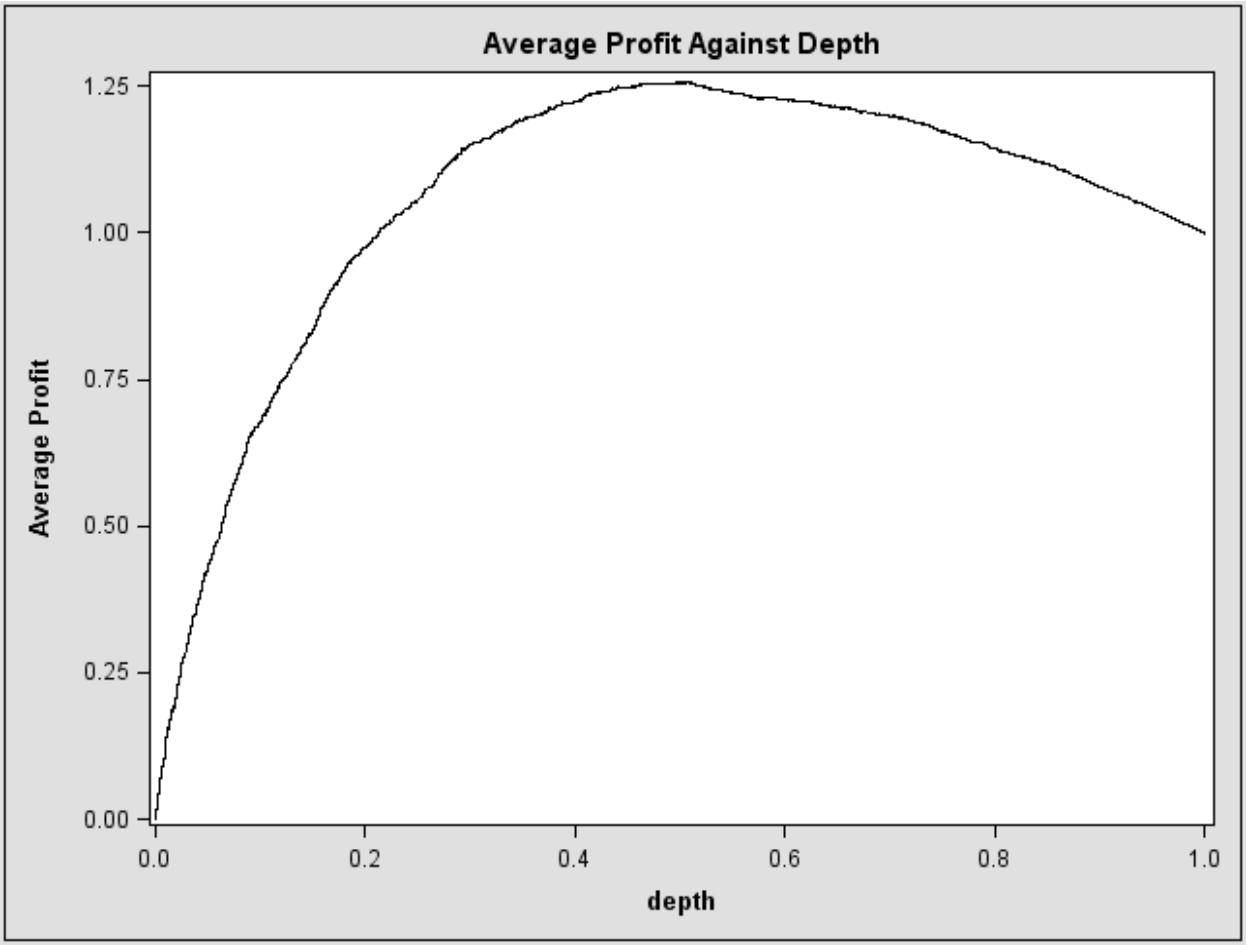
Sum	Mean
13434.53	1.2494791

Using this model to score a population of 1,000,000 individuals and soliciting only those with p_i greater than 0.01 yields a total expected profit of $1,249,479.10. With the above profit matrix and π_1=0.02, the "solicit everyone" rule generates a profit of 1,000,000*0.02*99-1,000,000*.98*1= $1,000,000. Using the model and some elementary decision theory leads to better decisions and more profit. Other models can be compared to this current model with this statistic.

Example: To see how other cutoffs perform, use the information in the **ROC** data set to draw a plot of how average profit changes as a function of the solicited depth and the cutoff. For the plot showing the cutoff, restrict the plot to the region around the cutoff of 0.01.

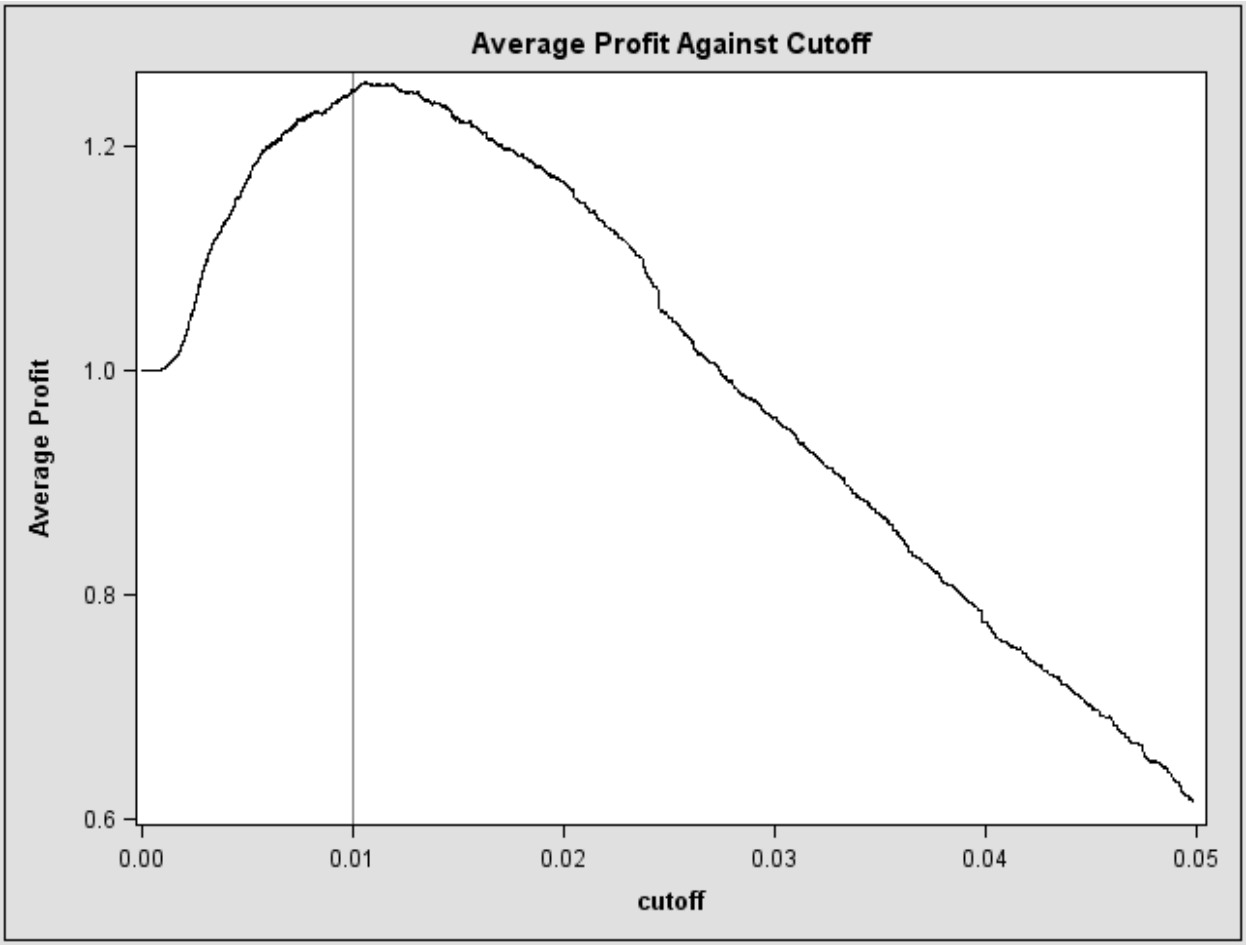
Using the true positive and false positive rates calculated earlier, you can calculate the average profit for each of the cutoffs considered in the **ROC** data set. A false positive (**fp**) is an individual who is solicited but does not respond. Hence, the cost of soliciting the false positives, on average, is the $1 solicitation cost times the false positive rate. Likewise, the profit associated with individuals who are solicited and respond is $99 times the true positive rate, **tp**. The difference of these two terms is the average profit.

```
data work.roc;
   set work.roc;
   AveProf = 99*tp - 1*fp;
run;

title "Average Profit Against Depth";
proc sgplot data=work.roc;
   series y=aveProf x=depth;
   yaxis label="Average Profit";
run;
```



The plot shows that the highest average profit occurs at a solicited depth of 50% to 60%.

```
title "Average Profit Against Cutoff";
proc sgplot data=work.roc;
   where cutoff le 0.05;
   refline .01 / axis=x;
   series y=aveProf x=cutoff;
   yaxis label="Average Profit";
run;
```



The plot shows that the highest average profit occurs at a cutoff at approximately 0.011.

End of Demonstration

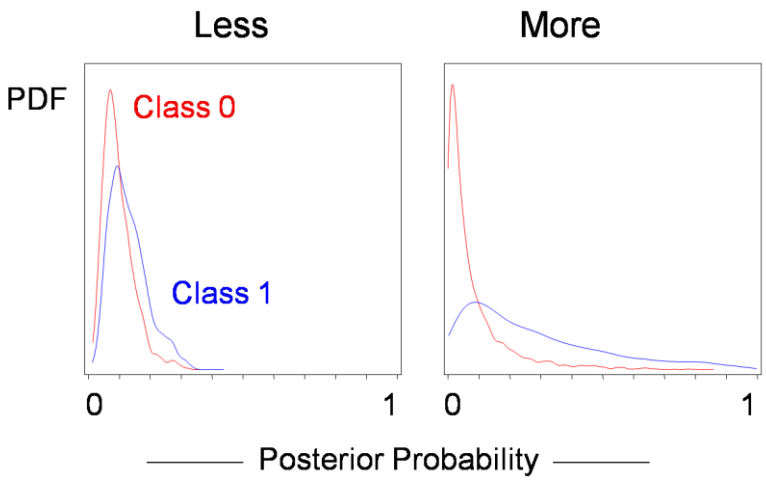
4.4 Overall Predictive Power

Objectives

- Describe the use of the Kolmogorov-Smirnov statistic.
- Compute the Kolmogorov-Smirnov statistic in PROC NPAR1WAY.

37

Class Separation

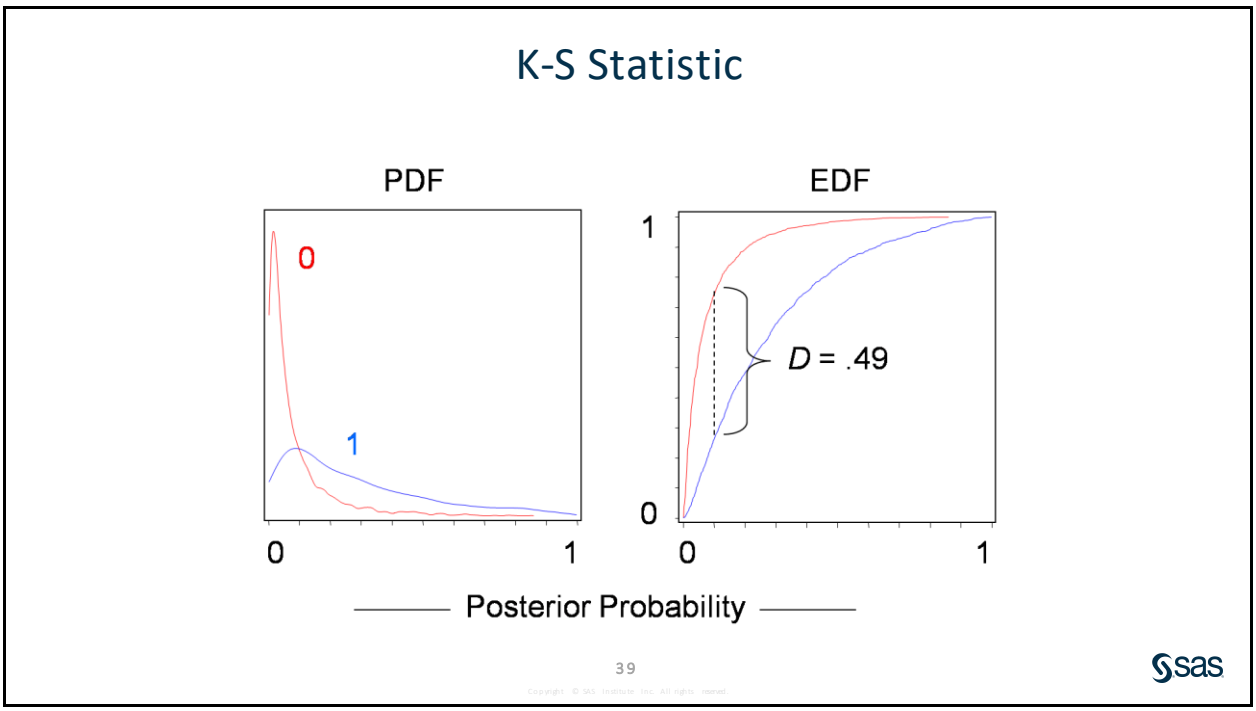


38

Statistics, such as sensitivity, positive predictive value, and risk, depend on the choice of cutoff value. Statistics that summarize the performance of a classifier across a range of cutoffs can also be useful for assessing global discriminatory power. One approach is to measure the separation between the predicted posterior probabilities for each class. When the distributions have more overlap, the model is weaker.

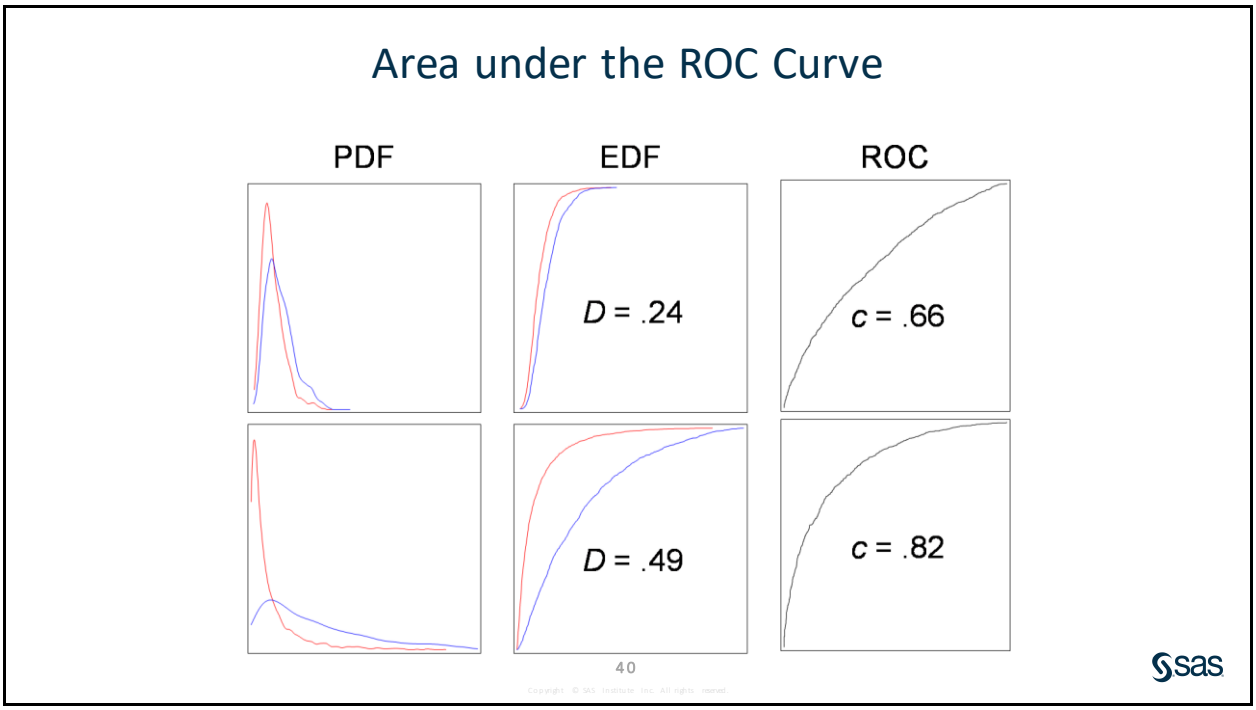
The simplest statistics are based on the difference between the means of the two distributions. In credit scoring, the *divergence* statistic is a scaled difference between the means (Nelson 1997). Hand (1997) discusses several summary measures based on the difference between the means.

The well-known *t* test for comparing two distributions is based on the difference between the means. The *t* test has many optimal properties when the two distributions are symmetric with equal variance (and have light tails). However, the distributions of the predicted posterior probabilities are typically asymmetric with unequal variance. Many other two-sample tests were devised for nonnormal distributions (Conover 1980).



The Kolmogorov-Smirnov two-sample test is based on the distance between the empirical distribution functions (Conover 1980). The test statistic, *D*, is the maximum vertical difference between the cumulative distributions. If *D* equals zero, the distributions are identical. If *D* is greater than 0, then there is some posterior probability where the distributions differ. The maximum value of the K-S statistic, 1, occurs when the distributions are perfectly separated. Use of the K-S statistic for comparing predictive models is popular in credit risk modeling.

An oversampled validation data set does not affect *D* because the empirical distribution function is unchanged if each case represents more than one case in the population. Furthermore, when you use the central cutoff, π_1, you maximize the *D* statistic.



The Kolmogorov-Smirnov two-sample test is sensitive to all types of differences between the distributions: location, scale, and shape. In the predictive modeling context, it could be argued that location differences are paramount. Because of its generality, the K-S test is not particularly powerful at detecting location differences. The most powerful nonparametric two-sample test is the Wilcoxon-Mann-Whitney test. Remarkably, the Wilcoxon-Mann-Whitney test statistic is also equivalent to the area under the ROC curve (Hand 1997).

The Wilcoxon version of this popular two-sample test is based on the ranks of the data. In the predictive modeling context, the predicted posterior probabilities are ranked from smallest to largest. The test statistic is based on the sum of the ranks in the classes. The area under the ROC curve, c, can be determined from the rank-sum in class 1.

$$c = \frac{\sum_{\{i|y=1\}}^{n_1} R_i - \frac{1}{2} n_1 (n_1 + 1)}{n_1 \cdot n_0}$$

The first term in the numerator is the sum of the ranks in class 1.

A perfect ROC curve would be a horizontal line at one. That is, sensitivity and specificity would both equal one for all cutoffs. In this case, the c statistic equals one. The c statistic technically ranges from zero to one. In practice, it should not be much lower than one-half. A perfectly random model, where the posterior probabilities were assigned arbitrarily, gives a 45° angle straight ROC curve that intersects the origin. Hence, it gives a c statistic of 0.5.

Oversampling does not affect the area under the ROC curve because sensitivity and specificity are unaffected. The area under the ROC curve is also equivalent to the Gini coefficient, which is used to summarize the performance of a Lorenz curve (Hand 1997).

4.04 Multiple Choice Poll

Which of the following statements is ***true*** regarding the K-S statistic?

a. The test statistic, D, is the minimum vertical distance between the cumulative distributions.

b. An oversampled validation data set does affect the test statistic D.

c. The K-S statistic is statistically equivalent to the area under the ROC curve.

d. The K-S statistic is not as powerful for detecting location differences as the Wilcoxon-Mann-Whitney test.

Calculating the K-S Statistic

Example: Use PROC NPAR1WAY to calculate the K-S statistic on the scored validation data set. Also, create an empirical distribution function plot.

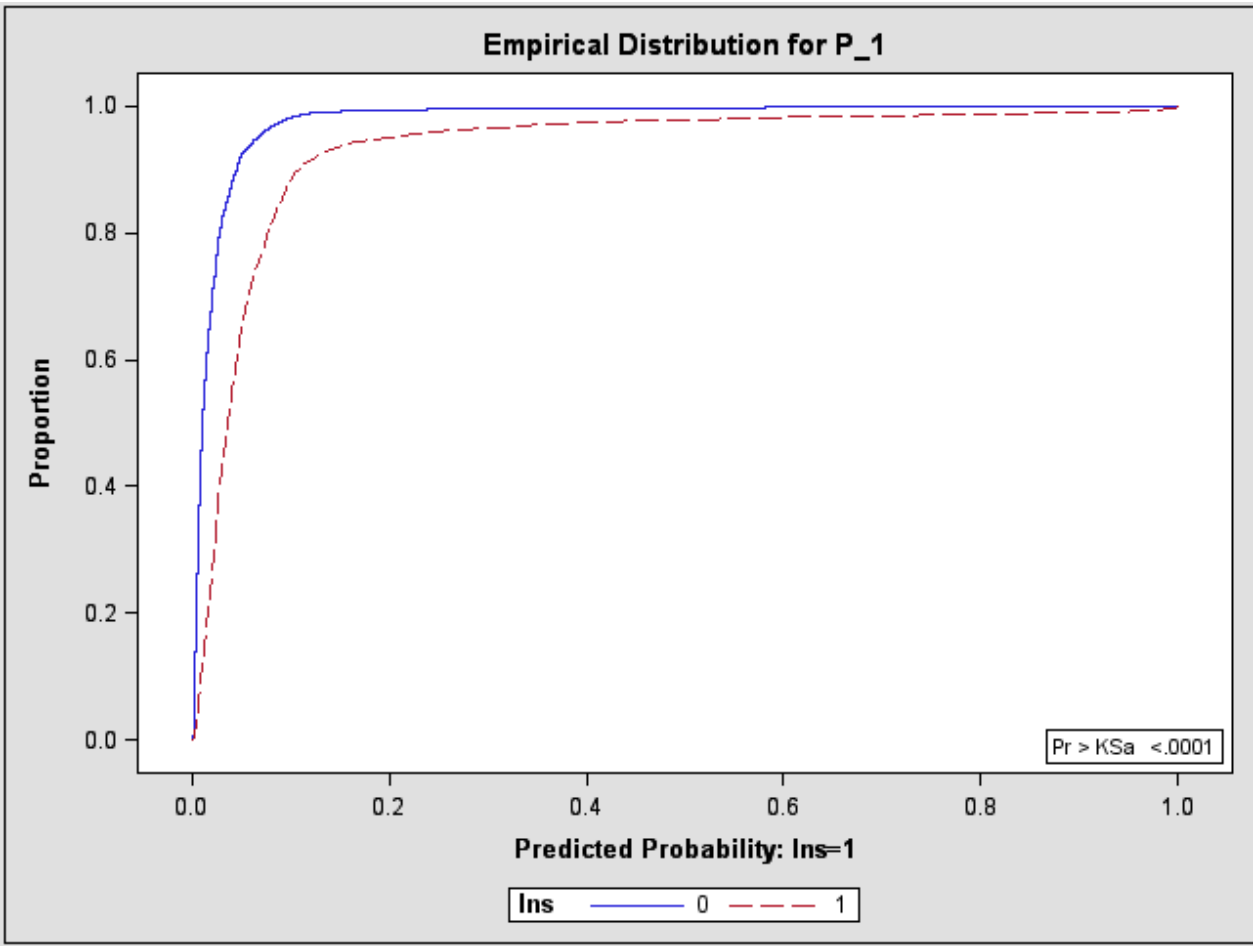
If you use the scored validation data set, the K-S statistic can be computed in the NPAR1WAY procedure. The EDF option in PROC NPAR1WAY requests the Kolmogorov-Smirnov test. The test compares the values of the variable listed in the VAR statement between the groups listed in the CLASS statement.

```
/* pmlr04d04.sas */

title1 "K-S Statistic for the Validation Data Set";
proc npar1way edf data=work.scoval;
   class ins;
   var p_1;
run;
```

Partial Output

```
                    K-S Statistic for the Validation Data Set

                              The NPAR1WAY Procedure

                      Kolmogorov-Smirnov Test for Variable P_1
                             Classified by Variable Ins

                                        EDF at    Deviation from Mean
Ins                    N               Maximum        at Maximum

0                   7028              0.710302         12.624590
1                   3724              0.275510        -17.343164
Total              10752              0.559710

           Maximum Deviation Occurred at Observation 2898
                 Value of P_1 at Maximum = 0.020933

           Kolmogorov-Smirnov Two-Sample Test (Asymptotic)

                KS    0.206877    D          0.434791
                KSa  21.451471    Pr > KSa     <.0001
```

The results show that the two-sample Kolmogorov statistic D is 0.43. This represents the largest separation between the two cumulative distributions.



The empirical distribution function plot shows the proportion of observations less than a given probability for the responders and nonresponders. The *D* statistic is the maximum vertical difference between the two distributions.

End of Demonstration

4.5 Model Selection Plots

Objectives

- Demonstrate the use of the ROC and ROCCONTRAST statements in PROC LOGISTIC.
- Use the ASSESS and FITANDSCORE macros to generate and evaluate many models.

46

ROC and ROCCONTRAST Statements

General form of the ROC and ROCCONTRAST statements:

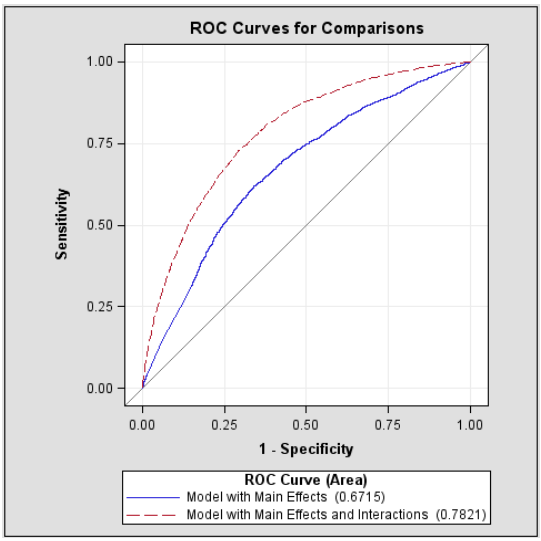
```
ROC <'label'> <specification> </ options>;
ROCCONTRAST <'label'> <contrast> </ options>;
```

47

You can use the ROC and ROCCONTRAST statements if you want to compare ROC curves from several models. The ROC statements specify models to be used in the ROC comparisons. You can specify more than one ROC statement. ROC statements are identified by their label. The specification can be either a list of effects that were previously specified in the MODEL statement, or PRED=variable, where the variable does not need to be specified in the MODEL statement. The PRED= option enables you to write a criterion produced outside PROC LOGISTIC to input.

The ROCCONTRAST statement compares the different ROC models. You can specify only one ROCCONTRAST statement.

Comparing ROC Curves



If ODS Graphics is enabled, then all ROC curves are displayed individually and are also overlaid in a final display.

Comparing ROC Curves

Example: Compare the ROC curves on the validation data set for the two models that correspond to the model fit in Chapter 2 and the model fit in Chapter 4.

To obtain the validation performance, you need to score the validation data with the two models.

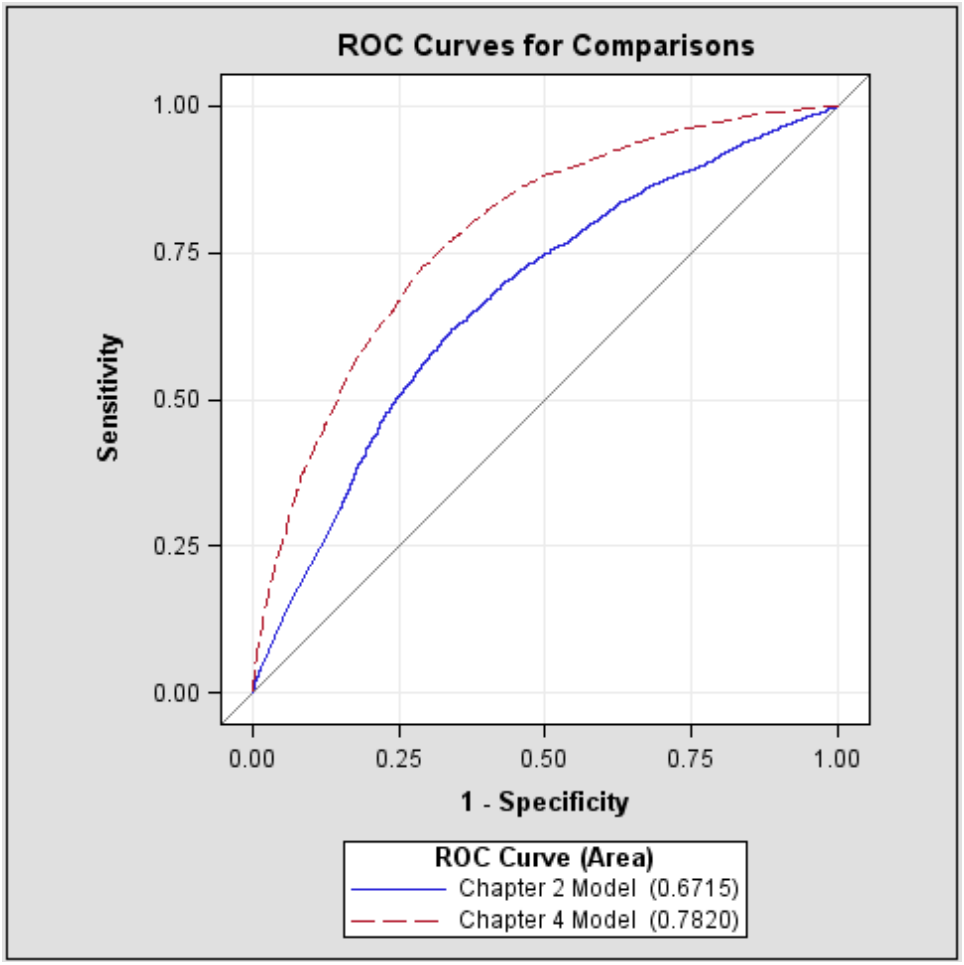
```
/* pmlr04d05.sas */

proc logistic data=work.train_imputed_swoe_bins noprint;
   class res;
   model ins(event='1')=dda ddabal dep depamt checks res;
   score data=work.valid_imputed_swoe_bins
         out=work.sco_validate(rename=(p_1=p_ch2));
run;

proc logistic data=work.train_imputed_swoe_bins noprint;
   model ins(event='1')=&selected;
   score data=work.sco_validate
         out=work.sco_validate(rename=(p_1=p_sel));
run;
```

Using the ROC and ROCCONTRAST statements, you can create and compare the ROC curves for the two models.

```
title1 "Validation Data Set Performance";
ods select ROCOverlay ROCAssociation ROCContrastTest;
proc logistic data=work.sco validate;
   model ins(event='1')=p_ch2 p_sel / nofit;
   roc "Chapter 2 Model" p ch2;
   roc "Chapter 4 Model" p_sel;
   roccontrast "Comparing the Two Models";
run;
```

Selected MODEL statement option:

NOFIT performs the global score test without fitting the model. This option is used so that only the models specified in the ROC statements are compared.



Validation Data Set Performance

The LOGISTIC Procedure

ROC Association Statistics

		Mann-Whitney					
		Standard	95% Wald		Somers' D		
ROC Model	Area	Error	Confidence Limits		(Gini)	Gamma	Tau-a
Chapter 2 Model	0.6715	0.00543	0.6609	0.6821	0.3430	0.3475	0.1553
Chapter 4 Model	0.7820	0.00456	0.7730	0.7909	0.5639	0.5640	0.2554

ROC Contrast Test Results

Contrast	DF	Chi-Square	Pr > ChiSq
Comparing the Two Models	1	528.4259	<.0001

The validation results show that the *c* statistic is much lower for the model fit in Chapter 2. However, because these inputs were selected arbitrarily, it is highly unlikely that the model in Chapter 2 can outperform the model fit in Chapter 4.

End of Demonstration

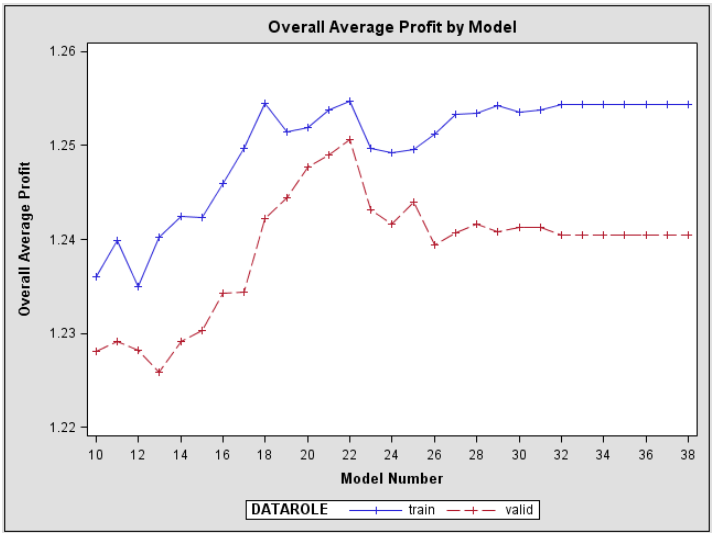
ASSESS and FITANDSCORE Macros

- Macros take a series of models generated by the best subsets logistic regression and compare them on the validation data performance.
- Plots of the results can be generated. This shows the performance gains as a function of model complexity, which should be a very useful tool for final model selection.

50

The process of selecting inputs for a model, fitting that model, and evaluating that model's fit on the validation data set can be automated with macro programming. This enables you to consider many candidate models in a small time frame, which should lead to better model generalization.

Comparing Models by Role and Average Profit



51

The macros use the profit matrix to estimate the overall average profit by model. The macros also separate the results by the training and validation data sets.

Comparing and Evaluating Models

Example: Use the **Assess** and **Fitandscore** macros to generate a series of models by best subsets logistic regression and compare them on the validation data set performance.

This demonstration uses the validation performance of many different models as a guide to selecting models that generalize well. Hence, the validation data must be prepared in the same way as the training data. The process begins with the screened variables from the demonstration in Chapter 3.

```
/* pmlr04d06.sas */
%put &screened;
```

```
SavBal Dep DDA CD Sav CC ATM MM branch_swoe Phone IRA IRABal B_DDABal ATMAmt
ILS POS NSF CCPurc SDB DepAmt CCBal Inv InArea Age CashBk MICRScor Income
```

The variable **MICRScor** needs to be created on the validation data set.

```
data work.valid_imputed_swoe_bins;
   set work.valid imputed swoe_bins;
   MICRScor = (crscore = .);
   resr=(res='R');
   resu=(res='U');
run;
title1 "Variables with Missing Values on the Validation Data Set";
proc means data=valid_imputed_swoe_bins nmiss;
   var &screened;
run;
```

Variables with Missing Values on the Validation Data Set

Variable	Label	N Miss
SavBal	Saving Balance	0
Dep	Checking Deposits	0
DDA	Checking Account	0
CD	Certificate of Deposit	0
Sav	Saving Account	0
CC	Credit Card	0
ATM	ATM	0
MM	Money Market	0
branch_swoe		0
Phone	Number Telephone Banking	1350
IRA	Retirement Account	0
IRABal	IRA Balance	0
B_DDABal		0
ATMAmt	ATM Withdrawal Amount	0
ILS	Installment Loan	0
POS	Number Point of Sale	1350
NSF	Number Insufficient Fund	0
CCPurc	Credit Card Purchases	1350
SDB	Safety Deposit Box	0
DepAmt	Amount Deposited	0
CCBal	Credit Card Balance	0
Inv	Investment	0

```
InArea          Local Address                0
Age             Age                       2061
CashBk          Number Cash Back             0
MICRScor                                     0
Income          Income                    1866
```

Several input variables have missing values on the validation data set.

Instead of using PROC UNIVARIATE to impute missing variables for selected variables, PROC STDIZE can be used to write to output a data set that contains the relevant information about the imputed values for the selected inputs. In addition, PROC STDIZE can also be used to take that information and use it to impute the training data set values in the validation data set.

As before, use PROC STDIZE with the REPONLY option to impute missing values on the training data. In addition, specify the OUTSTAT= option to save the imputed values in a separate data set, called **med**.

```
proc stdize data = work.train_imputed_swoe_bins method=median reponly
            OUTSTAT=med;
   var &screened;
run;
```

After the **med** data set is created, it can be used to impute for missing values in a different data set. The code below replaces the missing values on the validation data set with the medians from the imputed training data. The option to specify the data set with the median information is METHOD=IN(*data-set-name*).

```
proc stdize data=work.valid_imputed_swoe_bins reponly method=in(med)
            out=work.valid_imputed_swoe_bins;
   var &screened;
run;
```

Two macros are compiled. One, **%Assess**, assesses the performance of a model on a particular data set, and appends a record that summarizes that model's performance to the **results** data set. The **%Assess** macro is called in the **%Fitandscore** macro. The **%Fitandscore** macro fits and scores many different models (here, the series of models created by the best subsets algorithm). The parameters for the macro are the data set's name, the name of the target variable, the list of parameters in the model, the unique index number for the model, the π_1 value, the number of models for each data set to be displayed for each model size, and the profit matrix values. The predictor variables are the screened variables from an earlier demonstration, the dummy variables for **res**, and the interactions detected in the forward selection method.

```
%include "&PMLRfolder\pmlr04d06a.sas";
%include "&PMLRfolder\pmlr04d06b.sas";

%fitandscore(data_train=train_imputed_swoe_bins,
             data validate=valid imputed swoe bins,
             target=ins,predictors=&screened resr resu SavBal*B_DDABal
             MM*B DDABal branch swoe*ATMAmt B DDABal*Sav SavBal*SDB
             SavBal*DDA ATMAmt*DepAmt B_DDABal*ATMAmt SavBal*ATMAmt
             SavBal*IRA SavBal*MM SavBal*CC Sav*NSF DDA*ATMAmt Dep*ATM
             IRA*B_DDABal CD*MM MM*IRABal CD*Sav B_DDABal*CashBk
             Sav*CC,best=2,profit00=0,profit01=-1,profit10=0,
             profit11=99,pi1=0.02);
```

The macros are shown in an appendix.

The result of the **%Fitandscore** macro is a **results** data set.

```
title1 "Model Performance Measures for Training and Validation Data "
       "Sets";
proc print data=work.results(obs=24);
run;
```

Model Performance Measures for Training and Validation Data Sets

Obs	DATAROLE	INPUT_ COUNT	TOTAL_ PROFIT	OVERALL_ AVG_PROFIT	ASE	C	index
1	train	1	23129.89	1.07521	0.32753	0.65653	1
2	valid	1	11480.13	1.06770	0.32773	0.65483	1
3	train	1	21511.97	1.00000	0.32920	0.48375	2
4	valid	1	10751.16	0.99991	0.32932	0.47975	2
5	train	2	23760.31	1.10451	0.32416	0.68932	3
6	valid	2	11883.25	1.10520	0.32438	0.68850	3
7	train	2	24629.62	1.14492	0.32564	0.69135	4
8	valid	2	12105.74	1.12589	0.32588	0.68752	4
9	train	3	24966.72	1.16059	0.31447	0.73595	5
10	valid	3	12572.67	1.16931	0.31586	0.73638	5
11	train	3	24852.80	1.15530	0.32238	0.71582	6
12	valid	3	12211.33	1.13571	0.32262	0.71388	6
13	train	4	25865.37	1.20237	0.31313	0.75060	7
14	valid	4	12814.52	1.19181	0.31338	0.74923	7
15	train	4	25413.61	1.18137	0.31422	0.74006	8
16	valid	4	12528.90	1.16524	0.31574	0.73772	8
17	train	5	25843.04	1.20133	0.31157	0.76373	9
18	valid	5	12829.41	1.19319	0.31199	0.75949	9
19	train	5	25824.53	1.20047	0.31372	0.75299	10
20	valid	5	12911.14	1.20079	0.31514	0.75457	10
21	train	6	26009.50	1.20907	0.31205	0.76347	11
22	valid	6	12992.10	1.20832	0.31362	0.76183	11
23	train	6	26543.43	1.23389	0.31013	0.77402	12
24	valid	6	13110.44	1.21933	0.31051	0.77099	12

The results data set has the *c* statistic, average overall profit, and the ASE or average squared error. ASE is related to MSE (mean square error), but it has a different divisor.

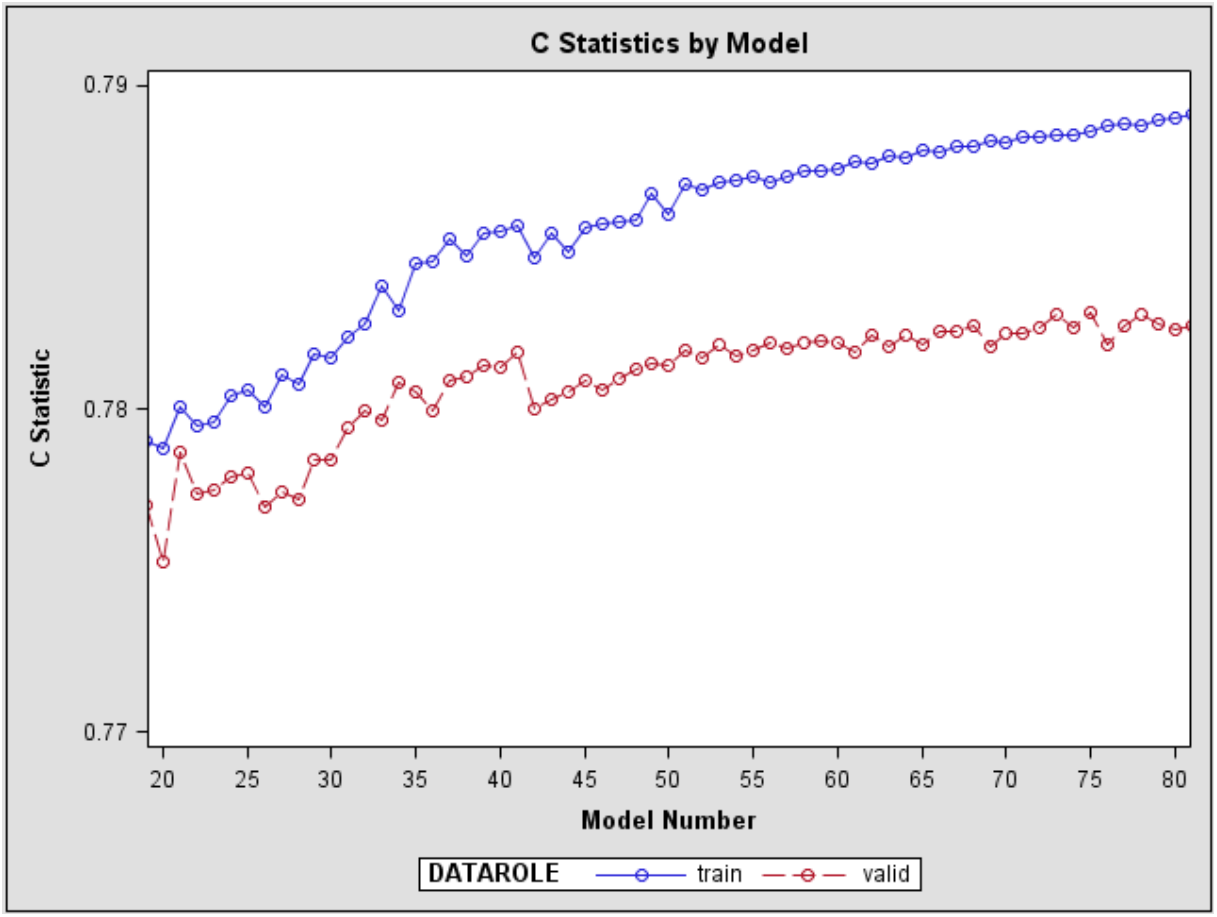
Example: Generate a graph of the *c* statistics by model and a graph of the overall average profit by model. Separate the plots by the training and validation data sets and show only the model with an index greater than 18 (10 inputs or more).

```
title1 "C Statistics by Model";
proc sgplot data=work.results;
   where index > 18;
   series y=c x=index / group=datarole markerattrs=(symbol=circle)
                        markers;
   yaxis label="C Statistic" Values=(0.770 to 0.790 by 0.001);
   xaxis label="Model Number" Values=(20 to 80 by 5);
run;
```

Selected SERIES statement options:

GROUP= specifies a variable that is used to group the data. A separate plot is created for each unique value of the grouping variable. The plot elements for each group value are automatically distinguished by different visual attributes.

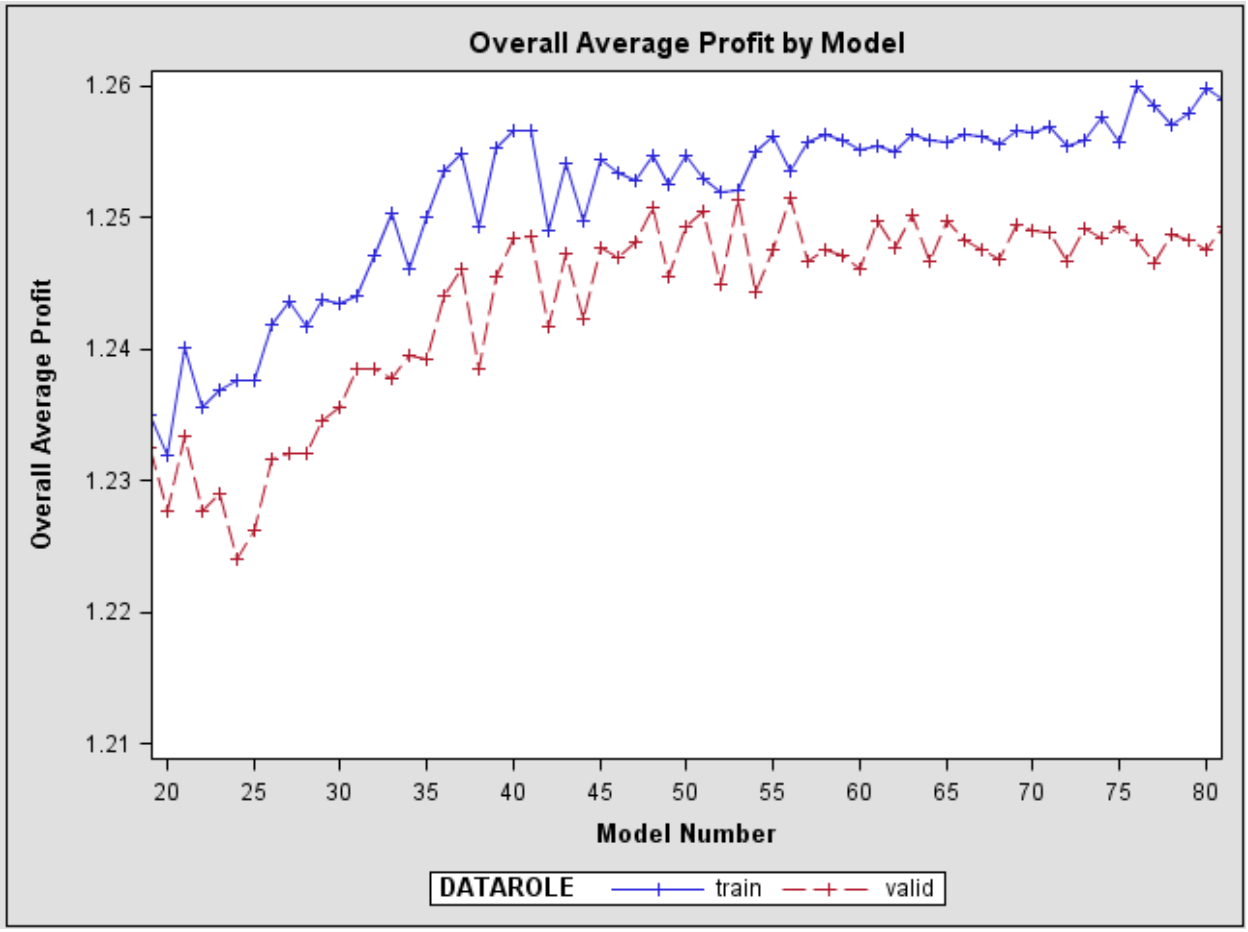
MARKERS adds data point markers to the series plot data points.



The plot seems to reach an early peak near index 41. Because the goal is to find a model that generalizes well, the simplest model that has good performance on the validation data is probably the best candidate.

Of course, it is difficult to quantify the trade-off between a slightly more complex model and a slightly higher *c* statistic. The trade-off might be easier to think about in terms of profit. What is it worth to the organization to use a more complex model? A plot of profit might help visualize and quantify this trade-off.

```
title1 "Overall Average Profit by Model";
proc sgplot data=work.results;
   where index > 18;
   series y=overall_avg_profit x=index / group=datarole
            markerattrs=(symbol=plus) markers;
   yaxis label="Overall Average Profit"
         Values=(1.21 to 1.26 by 0.010);
   xaxis label="Model Number" Values=(20 to 80 by 5);
run;
```



The plot seems to show that the validation data performance peaks at index 56. Plotting the ASE shows similar results as well.

Example: Find the model with the highest overall average profit on the validation data set. Show the results for that model, score the validation data set, and create an output data set with the results. Finally, compute a *c* statistic on the validation data set.

```
title1 "Model Number with Highest Profit";
%global index;
proc sql;
   select index into :index
   from work.results
   where datarole='valid'
   having overall_avg_profit=max(overall_avg_profit);
quit;
```

```
                Model Number with Highest Profit

                          index
                    ____________
                              56
```

Model number 56 is the model with the highest overall average profit.

The **%CMPRES** autocall macro removes all the blanks from the macro variable index.

```
%let index=%cmpres(&index);

title1 "Logistic Model with Highest Profit";
proc logistic data= work.train imputed swoe_bins;
   model ins(event='1')= &&inputs&index;
   score data= work.valid_imputed_swoe_bins out=work.scoval2 fitstat;
run;
```

```
                    Logistic Model with Highest Profit

                              Model Information

Data Set                      WORK.TRAIN_IMPUTED_SWOE_BINS
Response Variable             Ins
Number of Response Levels     2
Model                         binary logit
Optimization Technique        Fisher's scoring

          Number of Observations Read       21512
          Number of Observations Used       21512

                          Response Profile

            Ordered                       Total
              Value          Ins     Frequency

                1            0         14061
                2            1          7451

                Probability modeled is Ins=1.

                          Model Convergence Status

       Convergence criterion (GCONV=1E-8) satisfied.

                          Model Fit Statistics

                                        Intercept
                              Intercept           and
          Criterion                Only     Covariates

          AIC                27759.675      22665.951
          SC                 27767.651      22897.265
          -2 Log L           27757.675      22607.951

           Testing Global Null Hypothesis: BETA=0

Test                 Chi-Square       DF     Pr > ChiSq

Likelihood Ratio      5149.7241       28         <.0001
Score                 4479.4862       28         <.0001
Wald                  3529.6934       28         <.0001
```

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-1.9217	0.0926	430.2719	<.0001
SavBal	1	0.000164	9.545E-6	293.4418	<.0001
DDA	1	-0.2544	0.0491	26.8705	<.0001
CD	1	0.9672	0.0522	343.7165	<.0001
Sav	1	0.8459	0.0844	100.5593	<.0001
MM	1	1.7664	0.1462	146.0518	<.0001
branch_swoe	1	0.9408	0.0795	139.8803	<.0001
IRA	1	1.1164	0.1968	32.1891	<.0001
B_DDABal	1	0.0284	0.000996	811.3105	<.0001
ATMAmt	1	0.000201	0.000029	47.0271	<.0001
ILS	1	-0.2437	0.0766	10.1115	0.0015
NSF	1	0.3651	0.0651	31.5015	<.0001
Inv	1	0.4418	0.0980	20.3042	<.0001
SavBal*B_DDABal	1	-1.7E-6	1.219E-7	193.9633	<.0001
MM*B_DDABal	1	-0.0154	0.00213	52.4092	<.0001
branch_swoe*ATMAmt	1	0.000163	0.000022	52.3780	<.0001
Sav*B_DDABal	1	-0.00984	0.00131	56.7718	<.0001
SavBal*SDB	1	-0.00001	3.76E-6	10.0137	0.0016
SavBal*DDA	1	0.000029	4.514E-6	41.8837	<.0001
ATMAmt*DepAmt	1	-1.68E-9	3.34E-10	25.4236	<.0001
B_DDABal*ATMAmt	1	-1.19E-6	2.386E-7	25.0463	<.0001
SavBal*ATMAmt	1	1.396E-9	6.06E-10	5.3006	0.0213
SavBal*MM	1	-0.00003	6.677E-6	16.9925	<.0001
SavBal*CC	1	-0.00002	4.557E-6	26.1877	<.0001
DDA*ATMAmt	1	0.000076	0.000018	17.3257	<.0001
Dep*ATM	1	-0.1332	0.0120	122.6319	<.0001
IRA*B_DDABal	1	-0.0117	0.00290	16.2117	<.0001
CD*MM	1	-0.4142	0.1177	12.3753	0.0004
Sav*CC	1	0.3521	0.0528	44.5078	<.0001

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
ILS	0.784	0.674	0.911
NSF	1.441	1.268	1.637
Inv	1.555	1.284	1.885

Association of Predicted Probabilities and Observed Responses

Percent Concordant	78.7	Somers' D	0.574
Percent Discordant	21.3	Gamma	0.575
Percent Tied	0.0	Tau-a	0.260
Pairs	104768511	c	0.787

Fit Statistics for SCORE Data

Data Set	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC
WORK.VALID_IMPUTED_SWOE_BINS	10752	-5750.7	0.2672	11559.46	11559.62	11770.66

Data Set	SC	R-Square	Max-Rescaled R-Square	AUC	Brier Score
WORK.VALID_IMPUTED_SWOE_BINS	11770.66	0.197976	0.273139	0.782223	0.1768

End of Demonstration

4.6 Chapter Summary

One of the most important steps in predictive modeling is to assess the performance of the model. To correct for the optimism bias, a common strategy is to isolate a portion of the development data for assessment. The LOGISTIC procedure can then be used to score the data set used for assessment. Statistics that measure the predictive accuracy of the model include sensitivity and positive predicted value. Graphics such as the ROC curve, the gains chart, and the lift chart can also be used to assess the performance of the model.

If the assessment data set was obtained by splitting oversampled data, then the assessment data set needs to be adjusted. This can be accomplished by using the sensitivity, specificity, and the prior probabilities.

In predictive modeling, the ultimate use of logistic regression is to allocate cases to classes. To determine the optimal cutoff probability, the plug-in Bayes rule can be used. The information that you need is the ratio of the costs of false negatives to the cost of false positives. This optimal cutoff minimizes the total expected cost (or maximize the total expected profit).

The profit itself can be used as an assessment statistic, presuming that some reasonable estimate of the appropriate financial figures can be found.

When the target event is rare, the cost of a false negative is usually greater than the cost of a false positive. For example, the cost of not soliciting a responder is greater than the cost of sending a promotion to someone who does not respond. Such considerations specify cutoffs that are usually much less than .50, the cutoff that maximizes accuracy.

A popular statistic that summarizes the performance of a model across a range of cutoffs is the Kolmogorov-Smirnov statistic. However, this statistic is not as powerful in detecting location differences as the Wilcoxon-Mann-Whitney test. Furthermore, the Wilcoxon-Mann-Whitney test statistic is equivalent to the area under the ROC curve (the *c* statistic). Thus, the *c* statistic should be used to assess the performance of a model across a range of cutoffs.

The output of a predictive model should be predictions that generalize well over time. Using the validation data set for honest assessment, a series of models can be compared according to their fitness to this task. Appropriate measures of fit and complexity might vary.

If there is no profit information, the MSE (or, equivalently, ASE) can be used as a model fit metric to detect models with low bias or lack-of-fit.

General form of PROC NPAR1WAY:

```
PROC NPAR1WAY DATA=SAS-data-set <options>;
     CLASS variable;
     VAR variable;
RUN;
```

4.7 Solutions

Solutions to Exercises

1. Model Assessment

a. Include the program pmlr03e03.sas or use the **PMLR_UpToDate** macro. Prepare the validation data set to be scored by the model fitted by the training data set.

1) Use PROC MEANS to examine which variables in the validation data set have missing values. Use the inputs from the model fitted by the training data set.

2) Use PROC UNIVARIATE to create a data set with the medians from the training data set of the variables with missing values.

3) Use the DATA step to impute the variables with missing values and to include the scoring code to create the smoothed weight of evidence for **cluster_code**.

Note: An electronic copy of the solution program is in **pmlr04s01.sas**.

```
title1 "Variables with Missing Values on the Validation Data Set";
proc means data = pmlr.pva_valid nmiss;
   var LIFETIME_GIFT_COUNT LAST_GIFT_AMT MEDIAN_HOME_VALUE
       FREQUENCY_STATUS_97NK PEP_STAR INCOME_GROUP
       LIFETIME_AVG_GIFT_AMT MONTHS_SINCE_LAST_GIFT;
run;
```

```
Variables with Missing Values on the Validation Data Set

                                N
Variable                     Miss
_________________________________

LIFETIME_GIFT_COUNT             0
LAST_GIFT_AMT                   0
MEDIAN_HOME_VALUE               0
FREQUENCY_STATUS_97NK           0
PEP_STAR                        0
INCOME_GROUP                 2229
LIFETIME_AVG_GIFT_AMT           0
MONTHS_SINCE_LAST_GIFT          0
_________________________________
```

4) Which input variable has missing values?
The input variable income_group has missing values.

```
proc univariate data=pmlr.pva_train_imputed_swoe noprint;
   var INCOME_GROUP;
   output out=work.medians
          pctlpts=50
          pctlpre=income_group;
run;
```

```
data pmlr.pva_valid_imputed_swoe(drop=income_group50 i);
   if  N  = 1 then set work.medians;
   set pmlr.pva_valid;
   array x(*) income group;
   array med(*) income_group50;
      do i = 1 to dim(x);
         if x(i)=. then x(i)=med(i);
      end;
   %include clswoe;
run;
```

b. Fit a logistic regression model on the training data set using **TARGET_B** as the target variable and the macro variable **EX_SELECTED** as the input variables.

1) Use the EVENT= option to model the probability that **TARGET_B**=1.

2) Use the SCORE statement to score the validation data set and adjust for oversampling.

3) Use the OUTROC= option to create a data set and an ROC curve for the validation data set.

4) Use the FITSTAT option to generate model fit statistics.

```
title1 "Training Data Set Model";
proc logistic data= pmlr.pva train imputed swoe;
   model target_b(event='1')=&ex_selected;
   score data= pmlr.pva_valid_imputed_swoe  priorevent=&ex_pi1
         outroc=work.roc fitstat;
run;
```

```
                              Training Data Set Model

                                 Model Information

                  Data Set                      PMLR.PVA_TRAIN_IMPUTED_SWOE
                  Response Variable             TARGET_B
                  Number of Response Levels     2
                  Model                         binary logit
                  Optimization Technique        Fisher's scoring

                            Number of Observations Read        9687
                            Number of Observations Used        9687

                                 Response Profile

                           Ordered                      Total
                             Value     TARGET_B     Frequency

                                 1            0          7265
                                 2            1          2422

                          Probability modeled is TARGET_B=1.

                               Model Convergence Status

                        Convergence criterion (GCONV=1E-8) satisfied.
```

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	10897.230	10514.106
SC	10904.409	10585.892
-2 Log L	10895.230	10494.106

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	401.1240	9	<.0001
Score	405.4144	9	<.0001
Wald	382.4438	9	<.0001

Analysis of Maximum Likelihood Estimates

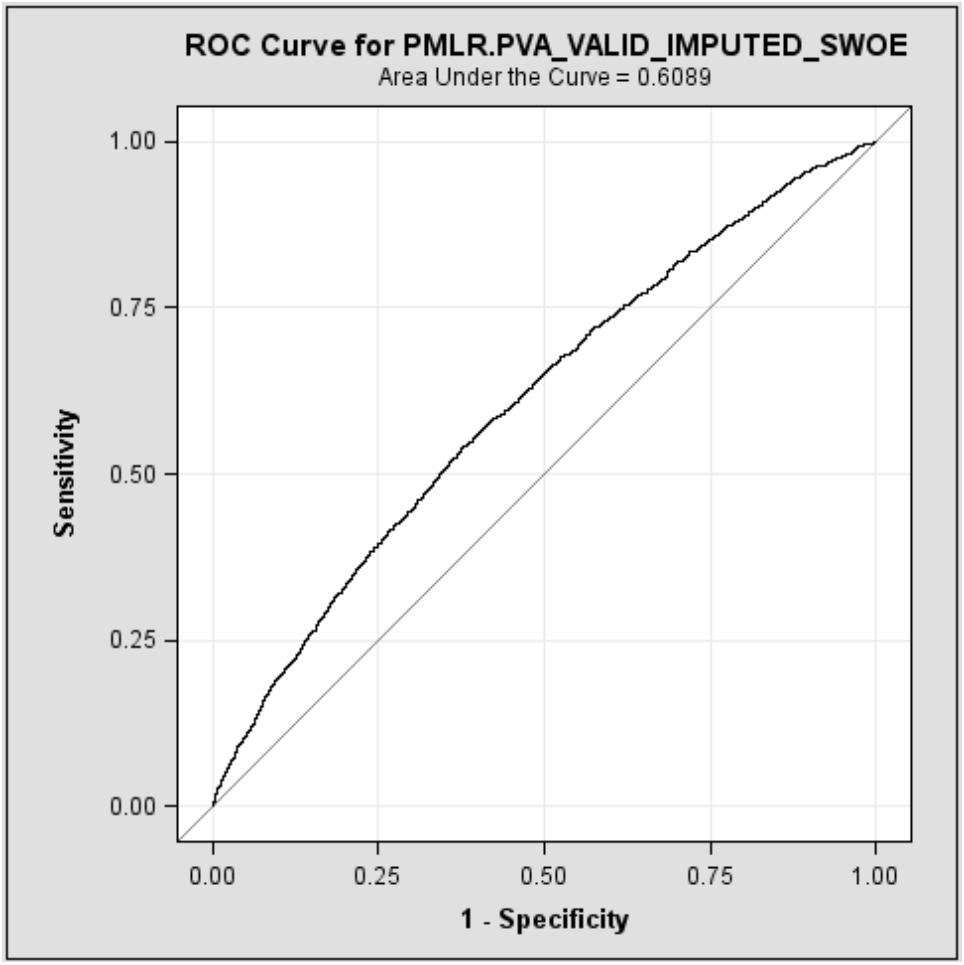
Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	-0.6217	0.2112	8.6698	0.0032
LIFETIME_GIFT_COUNT	1	0.0401	0.00670	35.9259	<.0001
LAST_GIFT_AMT	1	-0.0183	0.00398	21.1735	<.0001
MEDIAN_HOME_VALUE	1	0.000095	0.000026	13.4529	0.0002
FREQUENCY_STATUS_97N	1	0.1720	0.0253	46.3852	<.0001
cluster_swoe	1	0.9869	0.1493	43.6931	<.0001
PEP_STAR	1	0.3248	0.0614	27.9318	<.0001
INCOME_GROUP	1	0.0471	0.0154	9.3146	0.0023
LAST_GIFT*LIFETIME_A	1	0.000167	0.000050	11.1250	0.0009
LIFETIME_*MONTHS_SIN	1	-0.00211	0.000366	33.3864	<.0001

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
MEDIAN_HOME_VALUE	1.000	1.000	1.000
FREQUENCY_STATUS_97N	1.188	1.130	1.248
cluster_swoe	2.683	2.002	3.595
PEP_STAR	1.384	1.227	1.561
INCOME_GROUP	1.048	1.017	1.080

Association of Predicted Probabilities and Observed Responses

Percent Concordant	63.2	Somers' D	0.263
Percent Discordant	36.8	Gamma	0.263
Percent Tied	0.0	Tau-a	0.099
Pairs	17595830	c	0.632



Fit Statistics for SCORE Data

Data Set	Total Frequency	Log Likelihood	Error Rate	AIC	AICC	BIC
PMLR.PVA_VALID_IMPUTED_SWOE	9685	-7444.5	0.2499	14908.97	14909	14980.76

Data Set	SC	R-Square	Max-Rescaled R-Square	AUC	Brier Score
PMLR.PVA_VALID_IMPUTED_SWOE	14980.76	0.036643	0.046213	0.608916	0.223326

5) What is the *c* statistic for the validation data set?

The *c* statistic for the validation data set is 0.6089.

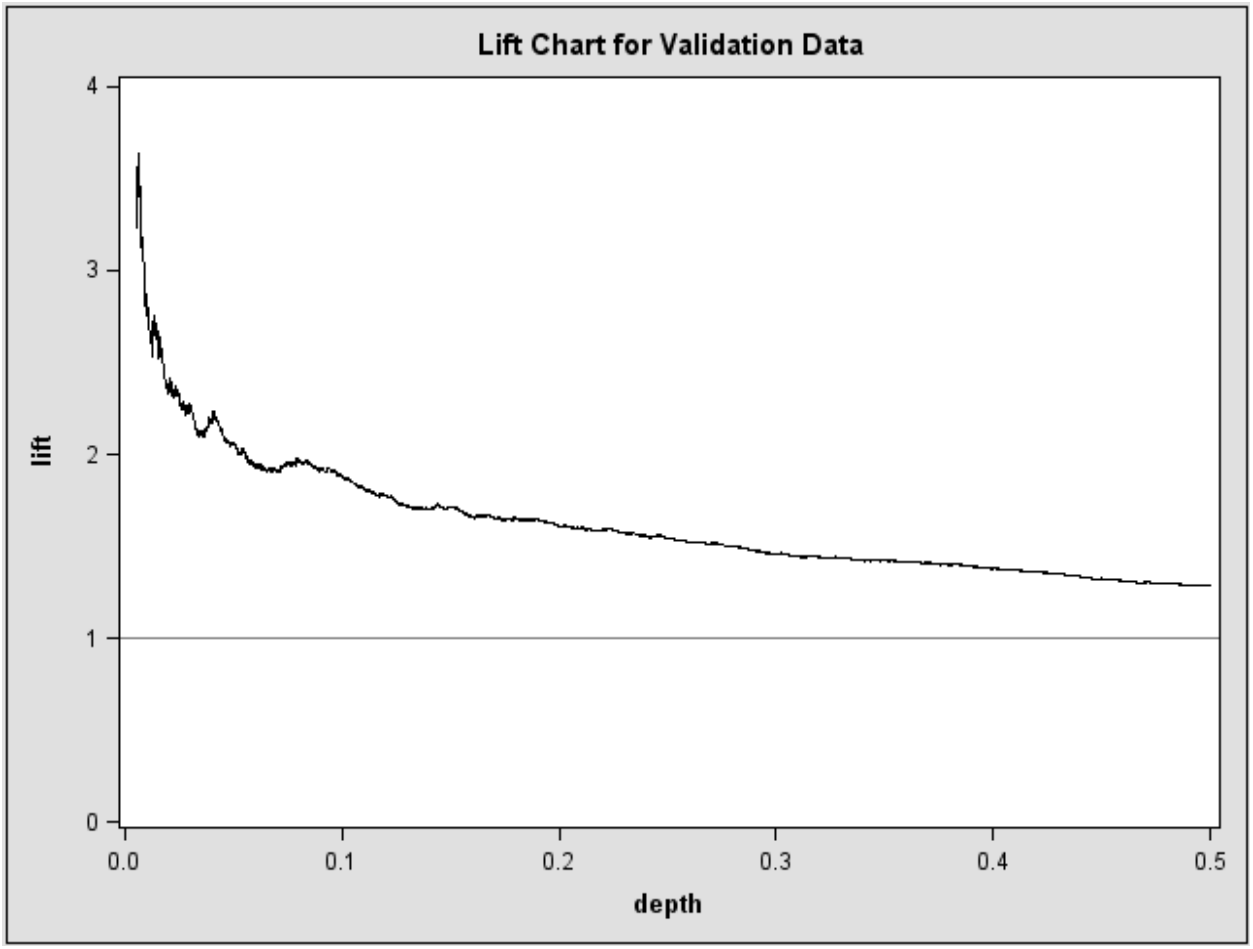
c. Use the OUTROC= data set. Write a DATA step to compute the proportion of true positives, the proportion of false negatives, the proportion of true negatives, the proportion of false positives, the positive predicted value, the negative predicted value, the accuracy, the proportion allocated to class 1 (depth), and the lift.

1) Use PROC SGPLOT to create a lift chart.

2) Add a reference line at a lift of 1 and restrict the focus to the region where depth is greater than 0.5% and less than 50%.

3) Restrict the Y axis from 0 to 4 by 1.

```
data work.roc;
   set work.roc;
   cutoff= PROB ;
   specif=1-_1MSPEC_;
   tp=&ex_pi1*_SENSIT_;
   fn=&ex pi1*(1- SENSIT );
   tn=(1-&ex_pi1)*specif;
   fp=(1-&ex pi1)*_1MSPEC_;
   depth=tp+fp;
   pospv=tp/depth;
   negpv=tn/(1-depth);
   acc=tp+tn;
   lift=pospv/&ex pi1;
   keep cutoff tn fp fn tp
         SENSIT   1MSPEC  specif depth
        pospv negpv acc lift;
run;

title1 "Lift Chart for Validation Data";
proc sgplot data=work.roc;
   where 0.005 <= depth <= 0.50;
   series y=lift x=depth;
   refline 1.0 / axis=y;
   yaxis values=(0 to 4 by 1);
run; quit;
```



4) What is the lift at a depth of 10%?

The lift at a depth of 10% is approximately 1.9.

End of Solutions

Solutions to Student Activities (Polls/Quizzes)

4.01 Multiple Choice Poll – Correct Answer

What is the formula for sensitivity?

a. true positives / total actual positives
b. true positives / total predicted positives
c. true negatives / total actual negatives
d. true negatives / total predicted negatives

4.02 Multiple Choice Poll – Correct Answer

What is the lift at a depth of 10%?

a. 1.5
b. 1.9
c. 2.3
d. 2.7

4.03 Multiple Choice Poll – Correct Answer

If the profit margin of true positives is nine times higher than the loss margins of false positives, then according to the Bayes rule, what is the cutoff that maximizes the expected profit (assuming 0 profit and loss for true negatives and false negatives)?

a. 0.90
b. 0.09
c. 1/9
d. 0.10

4.04 Multiple Choice Poll – Correct Answer

Which of the following statements is ***true*** regarding the K-S statistic?

a. The test statistic, D, is the minimum vertical distance between the cumulative distributions.
b. An oversampled validation data set does affect the test statistic D.
c. The K-S statistic is statistically equivalent to the area under the ROC curve.
d. The K-S statistic is not as powerful for detecting location differences as the Wilcoxon-Mann-Whitney test.

Appendix A References

A.1 References

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. 1984. *Classification and Regression Trees*. London: Chapman and Hall.

Cohen, A. 1991. "Dummy Variables in Stepwise Regression." *The American Statistician*. 45:226–228.

Conover, W. J. 1980. *Practical Nonparametric Statistics*. New York: John Wiley & Sons.

Donner, A. 1982. "The Relative Effectiveness of Procedures Commonly Used in Multiple Regression Analysis for Dealing with Missing Values. *The American Statistician*. 36:378–381.

Duffy, T. J. and D. E. Santner. 1989. *The Statistical Analysis of Discrete Data*. New York: Springer-Verlag.

Greenacre, M. J. 1988. "Clustering Rows and Columns of a Contingency Table." *Journal of Classification*. 5:39–51.

Greenacre, M. J. 1993. *Correspondence Analysis in Practice*. San Diego, CA: Academic Press.

Hand, D. J. 1997. *Construction and Assessment of Classification Rules*. New York: John Wiley & Sons.

Hand, D. J. and W. E. Henley. 1997. "Statistical Classification Methods in Consumer Credit Scoring: A Review." *Journal of the Royal Statistical Society A*. 160:153–541.

Harrell, F. E. 1997. *Predicting Outcomes: Applied Survival Analysis and Logistic Regression*. Charlottesville, Virginia: School of Medicine, University of Virginia.

Hastie, T. J. and R. J. Tibshirani. 1990. *Generalized Additive Models*. London: Chapman and Hall.

Huber, P. J. 1997. "From Large to Huge: A Statistician's Reactions to KDD & DM." *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. Menlo Park, CA: AAAI Press.

Jackson, J. E. 1991. *A Users Guide to Principal Components*. New York: John Wiley & Sons.

Jones, M. P. 1996. "Indicator and Stratification Methods for Missing Explanatory Variables in Multiple Linear Regression." *Journal of the American Statistical Association*. 91:222–230.

Lawless, J. F. and K. Singhal. 1978. "Efficient Screening of Nonnormal Regression Models." *Biometrics*. 34:318–327.

Little, R. J. A. 1992. "Regression with Missing X's: A Review." *Journal of the American Statistical Association*. 87:1227–1237.

Magee, L. 1998. "Nonlocal Behavior in Polynomial Regressions." *The American Statistician*. 52:20–22.

Mantel, N. 1970. "Why Stepdown Procedures in Variable Selection." *Technometrics*. 12:621–625.

McLachlan, G. J. 1992. *Discriminant Analysis and Statistical Pattern Recognition*. New York: John Wiley & Sons.

Nelson, Richard W. 1996. *Credit Card Risk Management*. Mount Holly, NJ: Warren-Taylor Publishing.

Prentice, R. L. and R. Pike. 1979. “Logistic Disease Incidence Models and Case-Control Studies.” *Biometrika*. 66:403–411.

Ripley, B. D. 1996. *Pattern Recognition and Neural Networks*. New York, NY: Cambridge University Press.

Sarle, W. S. 1994. “Neural Networks and Statistical Models.” *Proceedings of the Nineteenth Annual SAS Users Group International Conference*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2016. *Advanced Predictive Modeling Using SAS® Enterprise Miner™ Course Notes*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2017. *Base SAS® 9.4 Procedures Guide, Seventh Edition*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 1995. *Logistic Regression Examples Using the SAS® System, Version 6, First Edition*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2016. *SAS® 9.4 Language Reference: Concepts, Sixth Edition*. Cary, NC: SAS Institute Inc.

SAS Institute Inc. 2015. *SAS/STAT® 14.1 Users Guide*. Cary, NC: SAS Institute Inc.

Scott, A. J. and C. J. Wild. 1986. “Fitting Logistic Regression Models under Case-Control or Choice Based Sampling.” *Journal of the Royal Statistical Society B*. 48:170–182.

Scott, A. J. and C. J. Wild. 1997. “Fitting Regression Models to Case-Control Data by Maximum Likelihood.” *Biometrika*. 84:57–71.

Appendix B Additional Resources

B.1 Sampling Weights

$$weight_i = \begin{cases} \dfrac{\pi_1}{\rho_1} & \text{if } y_i = 1 \\ \dfrac{\pi_0}{\rho_0} & \text{if } y_i = 0 \end{cases}$$

Another method for adjusting for oversampling is to incorporate sampling weights. Sampling weights adjust the data so that it better represents the true population. When a rare target event is oversampled, class 0 is underrepresented in the sample. Consequently, a class-0 case should actually count more in the analysis than a class-1 case. The predicted values are properly corrected by using weights that are inversely proportional to selection probabilities (for each class, the number of cases in the sample divided by the number of cases in the population). It is convenient to use the normalized sample weights because they sum to the original sample size.

$$\sum_{i=1}^{n} weight_i = n_0 \frac{\pi_0}{\rho_0} + n_1 \frac{\pi_1}{\rho_1} = n\pi_0 + n\pi_1 = n$$

The weights adjust the number of cases in the sample to be $n\pi_0$ and $n\pi_1$, in class 0 and 1 respectively. The classes now are in the same proportion in the adjusted sample as they are in the population. The normalization causes less distortion in standard errors and *p*-values. Although statistical inference is not the goal of the analysis, *p*-values are used as tuning parameters in variable selection algorithms.

The offset method and the weighted method are not statistically equivalent. The parameter estimates are not exactly the same, but they have the same large-sample statistical properties. When the linear-logistic model is correctly specified, the offset method (unweighted) analysis is considered superior. However, when the logistic model is merely an approximation to some nonlinear model, weighted analysis has advantages (Scott and Wild 1986).

$$\text{Sampling Weight} = \begin{cases} \dfrac{0.02}{0.35} = 0.058 \text{ if Ins} = 1 \\ \dfrac{0.98}{0.65} = 1.46 \text{ if Ins} = 0 \end{cases}$$

Sampling Weights

Example: Add sampling weights to the data set **develop**. The weights are .058 (.02/.346) for class 1 and 1.5 (.98/.654) for class 0. Then use the WEIGHT statement in PROC LOGISTIC to weight each observation in the input data set by the value of the WEIGHT variable.

The sampling weights can be assigned manually without referencing macro variables. The logical expressions (**Ins=1**) and (**Ins=0**) in the assignment statement return the value 1 when true and 0 when false. Consequently, this syntax is a more compact way of expressing a conditional.

```
%let pi1 = 0.02;

proc sql noprint;
   select mean(ins) into :rho1 from pmlr.develop;
quit;

data develop;
   set pmlr.develop;
   sampwt=((1-&pi1)/(1-&rho1))*(ins=0)+(&pi1/&rho1)*(ins=1);
run;

proc logistic data=develop des;
   weight sampwt;
   model ins = dda ddabal dep depamt cashbk checks / stb;
   score data=pmlr.new out=scored;
run;
```

```
                          The LOGISTIC Procedure

                            Model Information

                Data Set                      WORK.DEVELOP
                Response Variable             Ins
                Number of Response Levels     2
                Weight Variable               sampwt
                Model                         binary logit
                Optimization Technique        Fisher's scoring

                    Number of Observations Read       32264
                    Number of Observations Used       32264
                    Sum of Weights Read            32263.99
                    Sum of Weights Used            32263.99

                            Response Profile

              Ordered                      Total            Total
                Value          Ins     Frequency           Weight

                    1            1         11175          645.281
                    2            0         21089        31618.707

                         Probability modeled is Ins=1.
```

The results show that the response profile table has a new column named **Total Weight**. The figures in the column represent the sample sizes adjusted to the population proportions. Note that the Sum of the Weights equals the total sample size.

Model Convergence Status

Convergence criterion (GCONV=1E-8) satisfied.

Model Fit Statistics

Criterion	Intercept Only	Intercept and Covariates
AIC	6328.271	6194.870
SC	6336.653	6253.542
-2 Log L	6326.271	6180.870

Testing Global Null Hypothesis: BETA=0

Test	Chi-Square	DF	Pr > ChiSq
Likelihood Ratio	145.4014	6	<.0001
Score	230.9722	6	<.0001
Wald	156.2504	6	<.0001

The LOGISTIC Procedure

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq	Standardized Estimate
Intercept	1	-3.1308	0.0756	1714.2522	<.0001	
DDA	1	-0.8398	0.1207	48.4361	<.0001	-0.1583
DDABal	1	0.000024	4.241E-6	32.6018	<.0001	0.0669
Dep	1	-0.0756	0.0376	4.0510	0.0441	-0.0718
DepAmt	1	2.807E-6	6.199E-6	0.2050	0.6507	0.00814
CashBk	1	-0.5691	0.4240	1.8016	0.1795	-0.0461
Checks	1	0.00479	0.0105	0.2080	0.6484	0.0139

Odds Ratio Estimates

Effect	Point Estimate	95% Wald Confidence Limits	
DDA	0.432	0.341	0.547
DDABal	1.000	1.000	1.000
Dep	0.927	0.861	0.998
DepAmt	1.000	1.000	1.000
CashBk	0.566	0.247	1.299
Checks	1.005	0.984	1.026

Association of Predicted Probabilities and Observed Responses

Percent Concordant	53.6	Somers' D	0.282
Percent Discordant	25.3	Gamma	0.358
Percent Tied	21.1	Tau-a	0.128
Pairs	235669575	c	0.641

```
proc print data=scored(obs=20);
   var p_1 dda ddabal dep depamt cashbk checks;
run;
```

Obs	P_1	DDA	DDABal	Dep	DepAmt	Cash Bk	Checks
1	0.016095	1	56.29	2	955.51	0	1
2	0.017385	1	3292.17	2	961.60	0	1
3	0.016880	1	1723.86	2	2108.65	0	2
4	0.041854	0	0.00	0	0.00	0	0
5	0.016233	1	67.91	2	519.24	0	3
6	0.018463	1	2554.58	1	501.36	0	2
7	0.017019	1	0.00	2	2883.08	0	12
8	0.016586	1	2641.33	3	4521.61	0	8
9	0.041854	0	0.00	0	0.00	0	0
10	0.017213	1	52.22	1	75.59	0	0
11	0.019232	1	6163.29	2	2603.56	0	7
12	0.009292	1	431.12	2	568.43	1	2
13	0.041854	0	0.00	0	0.00	0	0
14	0.010447	1	112.82	8	2688.75	0	3
15	0.015850	1	1146.61	3	11224.20	0	2
16	0.041854	0	0.00	0	0.00	0	0
17	0.016535	1	1241.38	3	3538.14	0	15
18	0.018644	1	298.23	0	0.00	0	0
19	0.017152	1	367.04	2	4242.22	0	11
20	0.014986	1	1229.47	4	3514.57	0	10

The probabilities from the weighted analysis are similar but not equivalent to the probabilities estimated by the offset method (pseudo model).

End of Demonstration

B.2 Cluster Imputation Using the FASTCLUS Procedure

PROC FASTCLUS can be used to replace the missing values with the cluster means from the training data set. First, the data set is split into the training and validation data sets. PROC FASTCLUS is used to compute the cluster means on the training data set and save the cluster means on an output data set.

```
proc fastclus data=train impute outiter outseed=seed out=train1
              maxclusters=5;
   var &inputs;
run;
```

Selected PROC FASTCLUS statement options:

IMPUTE requests imputation of missing values after the final assignment of observations to clusters. If an observation that is assigned (or could be assigned) to a cluster has a missing value for variables used in the cluster analysis, the missing value is replaced by the corresponding value in the cluster seed to which the observation is assigned (or could be assigned). If the observation cannot be assigned to a cluster, missing value replacement depends on whether the NOMISS option is specified. If NOMISS is not specified, missing values are replaced by the mean of all observations in the DATA= *data set* having a value for that variable. If NOMISS is specified, missing values are replaced by the mean of only observations used in the analysis. (A weighted mean is used if a variable is specified in the WEIGHT statement.) If you specify the IMPUTE option, the imputed values are not used in computing cluster statistics. If you also request an OUT= data set, it contains the imputed values.

OUTITER writes output information from the iteration history to the OUTSEED= *data set*, including the cluster seeds at each iteration.

OUTSEED= is another name for the MEAN= *data set*. It is provided because the data set can contain location estimates other than means.

MAXCLUSTERS= specifies the maximum number of clusters permitted. If you omit the MAXCLUSTERS= option, a value of 100 is assumed.

PROC FASTCLUS is then used to replace the missing values from the validation data set with the cluster means from the training data set that was computed at the first iteration.

```
proc fastclus data=valid impute seed=seed(where=( iter =1))
     replace=none maxclusters=5 out=validate1 maxiter=0;
   var &inputs;
run;
```

Selected PROC FASTCLUS statement options:

SEED= specifies an input data set from which initial cluster seeds are to be selected. If you do not specify the SEED= option, initial seeds are selected from the DATA= *data set*. The SEED= *data set* must contain the same variables that are used in the data analysis.

REPLACE= specifies how seed replacement is performed. NONE suppresses the seed replacement.

MAXITER= specifies the maximum number of iterations for recomputing cluster seeds.

B.3 Imputation with PROC STDIZE

Instead of using PROC UNIVARIATE to impute missing variables for selected variables, it is possible to automate the imputation step without checking which variables have missing values. PROC STDIZE can be used to write to output a data set that contains the relevant information about the imputed values for every input. In addition, PROC STDIZE can also be used to take that information and use it to impute the training data set values in the validation data set. An example follows.

As before, use PROC STDIZE with the REPONLY option to impute missing values on the training data. In addition, specify the OUTSTAT= option to save the imputed values in a separate data set, called **med**.

```
proc stdize data=train out=train2 method=median reponly OUTSTAT=med;
   var &inputs;
run;
```

After the **med** data set is created, it can be used to impute for missing values in a different data set. The code below creates a data set, **valid2**. It is based on the unimputed valid data with the medians from the imputed training data. The option to specify the data set with the median information is METHOD=IN(*data-set-name*).

```
proc stdize data=valid out=valid2 reponly method=in(med);
   var &inputs;
run;
```

In order to generate a series of models, you probably want to impute for all inputs first. In this way, you can evaluate model performance on the validation data without intervening by hand to impute for missing values.

B.4 %ASSESS Macro

The %ASSESS macro simplifies the assessment of each of the models found by the best subsets procedure. Data for assessment are assumed to be stored in a data set named **SCORED&DATA**, where DATA is a macro parameter with anticipated values TRAIN, VALID, and potentially TEST. The INPUTCOUNT= macro parameter points to the number of parameters in the model in question. The INPUTSINMODEL= macro parameter points to the list of parameters in the model. The INDEX= macro parameter is a unique index for each model in the series. This enables you to have more than one model with 16 inputs, for example.

```
%macro assess(data=,inputcount=,inputsinmodel=,index=,pi1=,rho1=,
              target=,profit11=,profit01=,profit10=,profit00=);
```

The variables π_1, ρ_1, π_0, and ρ_0 are necessary to calculate sampling weights. π_1 must be known *a priori*, and ρ_1 was calculated earlier in PROC MEANS. To simplify the code, macro variables are created to house π_0 and ρ_0 as well. The %SYSEVALF function takes the text string in its argument and performs the calculation. For example, %SYSEVALF(1-**&pi1**) resolves to 0.98 in this instance.

```
%let rho0 = %sysevalf(1-&rho1);
%let pi0 = %sysevalf(1-&pi1);
```

The assessment data are sorted by descending posterior probability. This sorting enables the calculation of a *c* statistic by the **ASSESS** DATA step below.

```
proc sort data=scored&data;
   by descending p_1;
run;
```

The main assessment DATA step begins. The **DATAROLE** variable is defined to contain the type of assessment (for example, VALID). A two-by-two temporary array, **n**, is defined and initialized to all zeros. The **n** array contains the confusion matrix based on the profit matrix specified by decision processing.

A two-by-one temporary array, **w**, is defined and initialized to the ratio of the population and sample marginal averages (π_0/ρ_0) and (π_1/ρ_1). The **w** array is used to adjust the model posterior probabilities as well as to calculate total profit and the confusion matrix.

```
/* create assessment data set */
data assess;
   attrib DATAROLE length=$5;
   retain sse 0 csum 0 DATAROLE "&data";

   /* 2 x 2 count array, or count matrix */
   array n[0:1,0:1] _temporary_ (0 0 0 0);
   /* sample weights array */
   array w[0:1] _temporary_
      (%sysevalf(&pi0/&rho0) %sysevalf(&pi1/&rho1));
   keep DATAROLE INPUT_COUNT INDEX
      TOTAL_PROFIT OVERALL_AVG_PROFIT ASE C;
```

An observation is read from the assessment data.

```
   set scored&data end=last;
```

Assuming the LOGISTIC call that generates the scored data sets corrects for oversampling, you can use the predicted probabilities. If the probabilities are biased due to oversampling, and you do not correct for this before the assessment, you cannot do that here.

Profits for each decision alternative are calculated. The calculation is based on the user-specified decision data.

```
   d1=&Profit11*p_1+&Profit01*p_0;
   d0=&Profit10*p_1+&Profit00*p_0;
```

Variable **t** equals 1 if the target equals the primary target value and 0 otherwise. That is, **t** is a flag for response. Variable **d** equals 1 if the profit for decision 1 exceeds the profit for decision 0. That is, **d** is the profit-maximizing decision. The STRIP function removes trailing and leading blanks. Contrast this with the default behavior of the COMPRESS function, which removes all blanks in a field.

These variables might be defined differently in the presence of different decision rules or different values for the target classes (for example, a different profit matrix or a Y/N target instead of a 1/0 target).

```
/* T is a flag for response */
   t=(strip(&target)="1");
/* D is the decision, based on profit. */
   d=(d1>d0);
```

The appropriate cell of the confusion matrix array **n** is incremented by the appropriate element of the weight vector. The sum of squared errors, SSE, is incremented. The sum used to calculate the *c* statistic, **csum**, is incremented. The increment is positive only when the target value is zero. This is, in effect, a Riemann sum of the receiver operating characteristic (ROC) curve.

```
/* update the count matrix, sse, and c */
  n[t,d] + w[t];
  sse + (&target-p_1)**2;
  csum + ((n[1,1]+n[1,0])*(1-t)*w[0]);
```

On the last time through the DATA step, finalize the fit statistics and output. To finalize the ASE, divide the sum of squared errors by the sample size. Total profit is the product of the sample sizes in each cell of **n** with the appropriate profit amount. The *c* statistic is based on the sum of the area of rectangles that approximate the ROC curve, but those areas are based on the sample size. To restrict this area to be between 0 and 1, the range of the *c* statistic, divide through by the product of n_1 and n_0.

```
   if last then do;
      INPUT_COUNT=&inputcount;
      TOTAL_PROFIT = sum(&Profit11*n[1,1],&Profit10*n[1,0],
                         &Profit01*n[0,1],&Profit00*n[0,0]);
   OVERALL_AVG_PROFIT =
TOTAL_PROFIT/sum(n[0,0],n[1,0],n[0,1],n[1,1]);
      ASE = sse/sum(n[0,0],n[1,0],n[0,1],n[1,1]);
      C = csum/(sum(n[0,0],n[0,1])*sum(n[1,0],n[1,1]));
      index=&index;
      output;
   end;
run;
```

Update the **Results** data set with this data's assessment results.

```
proc append base=results data=assess force;
run;
```

End the %ASSESS macro.

```
%mend assess;
```

B.5 %FITANDSCORE Macro

The %FITANDSCORE macro refit the series of logistic regression models, and create the scored data sets that the %ASSESS macro needs.

```
%macro fitandscore(data_train=,
                   data_validate=,
                   target=,
                   predictors=,
                   best=,
                   profit00=,profit01=,profit10=,profit11=,
                   pi1=);
```

The first part of the macro creates a data set called **SCORE** that has the results of the best subsets selection. The macro variables for the number of models and rho1 are also created.

```
ods select none;
```

```
   ods output bestsubsets=score;
   proc logistic data=&data_train;
      model ins(event='1')=&predictors
          / selection=SCORE best=&best;
   run;

   %global nmodels;
   proc sql;
      select count(*) into :nmodels from score;
   run;

   %global rho1;
   proc sql;
      select mean(INS) into :rho1 from &data_train;
   run;

   %do i=1 %to &nmodels;
      %global inputs&i;
      %global ic&i;
   %end;
```

Macro variables for the variable names and the number of variables are created next.

```
   proc sql noprint;
      select variablesinmodel into :inputs1 -
         from score;
      select NumberOfVariables into :ic1 -
         from score;
    quit;
```

Any previous version of the **Results** data set is removed using PROC DATASETS.

```
proc datasets
   library=work
   nodetails
   nolist;
   delete results;
run;
```

PROC LOGISTIC refits the model with inputs specified in **IM**. The SCORE statement is used to score the training and validation data sets. Only the target value and posterior probabilities are kept. The PRIOREVENT= option corrects the predicted probabilities for oversampling.

```
%do model_indx=1 %to &nmodels;

%let im=&&inputs&model_indx;
%let ic=&&ic&model_indx;
```

```
proc logistic data=&data_train;
   model &target(event='1')=&im;
   score data=&data_train
         out=scored&data_train(keep=ins p_1 p_0)
         priorevent=&pi1;
   score data=&data_validate
         out=scored&data_validate(keep=ins p_1 p_0)
         priorevent=&pi1;
run;

%assess(data=&data_train,inputcount=&ic,inputsinmodel=&im,
       index=&model_indx,pi1=&pi1,rho1=&rho1,target=&target,
       profit11=&profit11, profit01=&profit01,profit10=&profit10,
       profit00=&profit00);
%assess(data=&data_validate,inputcount=&ic,inputsinmodel=&im,
       index=&model_indx,pi1=&pi1,rho1=&rho1,target=&target,
       profit11=&profit11, profit01=&profit01,profit10=&profit10,
       profit00=&profit00);
%end;
ods select all;

%mend fitandscore;
```

B.6 %THRESHOLDING Macro

The **%THRESHOLDING** macro applies thresholding recoding for levels of a categorical column that exceeds the specified cutoff value. The macro requires four inputs: the current data set name, the new data set name, the categorical column name, and the threshold cutoff value. If you want to write over the existing data set, the current data set and the new data set must be the same.

```
%macro THRESHOLDING(CURRENTDSN, NEWDSN, COLNAME, CUTOFF);
   proc sql noprint;
      select distinct left(trim(quote((&COLNAME.)))) into:lt
         separated by ', '
      from &CURRENTDSN.
      group by &COLNAME.
      having count(&COLNAME.) <= &CUTOFF.;

      select distinct left(trim((&COLNAME.))) into:gt
         separated by ' '
      from &CURRENTDSN.
      group by &COLNAME.
      having count(&COLNAME.) > &CUTOFF.;
      quit;
```

```
   data &NEWDSN.;
   set &CURRENTDSN.;
      %do i=1 %to %sysfunc(countw(&gt.)) %by 1;
         D_%scan(&gt,&i.)=(&COLNAME.="%scan(&gt,&i.)");
         label D_%scan(&gt,&i.)="D_%scan(&gt,&i.) from
         &COLNAME";
      %end;
   &COLNAME._OTHER=(&COLNAME. in(&lt.));
      label &COLNAME._OTHER="OTHER category from &COLNAME column";
   run;
%mend;
```

An example of executing the **THRESHOLDING** macro is shown below. Dummy variables with only 500 or more observations in a branch are created. All other branches are collapsed into a dummy variable **branch_other**.

```
%THRESHOLDING(work.train_imputed,work.train_imputed_branch_thres,
              branch,500);

proc print data=train_imputed_branch_thres(obs=10);
   var branch D_B1 D_B2 D_B3 D_B4 D_B5 D_B6 D_B7 D_B8 D_B14 D_B15
              D_B16 D_B17 branch_other;
run;
```

Obs	Branch	D_B1	D_B2	D_B3	D_B4	D_B5	D_B6	D_B7	D_B8	D_B14	D_B15	D_B16	D_B17	branch_ OTHER
1	B2	0	1	0	0	0	0	0	0	0	0	0	0	0
2	B1	1	0	0	0	0	0	0	0	0	0	0	0	0
3	B1	1	0	0	0	0	0	0	0	0	0	0	0	0
4	B14	0	0	0	0	0	0	0	0	1	0	0	0	0
5	B4	0	0	0	1	0	0	0	0	0	0	0	0	0
6	B16	0	0	0	0	0	0	0	0	0	0	1	0	0
7	B9	0	0	0	0	0	0	0	0	0	0	0	0	1
8	B7	0	0	0	0	0	0	1	0	0	0	0	0	0
9	B2	0	1	0	0	0	0	0	0	0	0	0	0	0
10	B1	1	0	0	0	0	0	0	0	0	0	0	0	0