Chapter 8
Sampling and Partitioning

8.1 Introduction

Sampling is used to facilitate the analysis and modeling of large datasets. There are several types of sampling schemes. Three of these schemes are commonly used in data mining: random sampling, balanced sampling, and stratified sampling (Levy and Lemeshow 1999).

In random sampling, a simple random sample, that is sampling without replacement, is drawn from the population to form the mining view. In balanced sampling, the user designs the sample such that the target variable, or some other predefined nominal or ordinal variable, is forced to have a certain composition (e.g., 90% nonresponders and 10% responders). Forcing samples to have such composition must be accompanied by defining and populating a weight variable to revert the proportions to their original values in the population. For example, when investigating fraud, the population may contain only 1% records with fraudulent behavior in them. Using a balanced sample with 10% records containing fraudulent behavior requires the definition of a weight variable with a value of 0.10 for sample records with (`fraud = yes`) and 1.10 for records having (`fraud = No`). These simple weights allow the models to revert the proportions to their original value in the population.

A third, but less common, type of sampling is stratified sampling. In this method, the data is divided into strata using the categories of one or more variables. Then simple random sampling is applied to each stratum to draw the samples. A typical example of this method is used in obtaining stratified samples based on geography using the state or province as the basis for defining the different strata. Other examples include using age or income groups.

Partitioning, on the other hand, is used to set aside a set of records to test on the developed models. Therefore, in most cases the data is partitioned into two partitions: a training partition, which is used fit the model, and a validation partition to validate

the model. A third partition may also be set aside for final testing of the model, as was shown in Figure 2.1.

In most cases, training and validation partitions are formed by random sampling from the mining view with specified percentages or numbers of records for each of them. Best practices suggest that in the case of using two partitions the training partition would contain 60 to 70% of the records of the mining view, with the remaining records used for the validation partition. These values are only guidelines out of experience without known theoretical foundation.

Fortunately, most data mining packages have dedicated algorithms for extracting samples and partitions from datasets. In the following, we present a simple macro built around the `PROC SURVEYSELECT` of SAS/STAT. This macro can be used to obtain random samples from large datasets. The macro simply wraps the `SURVEYSELECT` procedure.

```
%MACRO RandomSample(PopDS, SampleDS, SampleSize);
/* This macro performes simple random sampling */
    PROC SURVEYSELECT
        DATA=&PopDs
        METHOD=srs
        N=&SampleSize
        NOPRINT
        OUT=&SampleDS;
    RUN;
%MEND;
```

This macro can be easily used to extract random samples from the dataset that represent the population. In the following subsections, we will discuss the SAS implementation of the random and balanced sampling as well as other issues related to the use of sampling methods.

8.2 Contents of Samples

Implementing sampling in SAS requires the processing of several `DATA` steps and the use of several procedures including `SORT` and `SURVEYSELECT`. In order to maximize the performance of these procedures, we attempt to reduce the data included in these datasets to a minimum. This is particularly important in all procedures where we attempt to sort the dataset. The reduction of the data can be achieved by removing all the variables from the datasets except the row ID variable (e.g., customer ID), and the value of the dependent variable (e.g., response variable), if the dataset contains one. After obtaining the samples, which contain only the ID field, it is an easy exercise to merge the rest of the variables from the original dataset. We will adopt this approach in our SAS implementations.

8.3 Random Sampling

We have presented the macro `RandomSample()`, which extracts a random sample from a population dataset. However, in almost all data mining models the analyst needs at least two *disjoint* partitions or samples: a training partition and a validation partition. The simplest method of extracting these two partitions from the mining view is to use simple random sampling. The macro `RandomSample()` can do this job. However, what it does not do is ensure that these two partitions are not overlapping. This is a critical feature of training and validation partitions, without which the integrity of the model validation procedure is compromised. We present here a simple macro implementation that can extract two nonoverlapping partitions obtained at random from a dataset. However, before we present the macro we discuss the consistency conditions for random samples.

8.3.1 Constraints on Sample Size

Because the training and validation partitions must always be disjoint, it is important to make sure that we have enough records to satisfy the requirements of each partition. Checking that the available number of records is enough for extracting the required samples is a simple task in the case of random samples. The case of balanced samples is a little more involved, as is presented in the next section.

Given a population dataset S with a total of N records, two nonoverlapping samples, S_1 and S_2, each with N_1 and N_2 records, respectively, the following inequality must be satisfied:

$$N \geq N_1 + N_2, \tag{8.1}$$

for all such samples S_1 and S_2.

In the case of random samples, this is the only constraint we have to check, which is in most cases a trivial task.

8.3.2 SAS Implementation

The steps to obtain two nonoverlapping random samples can be summarized as follows.

1. Calculate the number of records in the population (N).
2. Check the consistency constraint.
3. Draw the first sample S_1 with size N_1.
4. Append the dataset S_1 with a new variable to indicate that these records have already been sampled. Give this new field a value of 1.
5. Match-merge the dataset S_1 with the population S using the row ID. Store the resulting dataset in a temporary dataset, `TEMP`.

6. Draw the second sample S_2, with size N_2, from TEMP under the condition that the record selection indicator added in Step 4 is *not* 1.

There are many ways to implement this algorithm. In our implementation we opted for the following preferences:

- We used PROC SQL to perform simple record counting operations using the syntax: SELECT COUNT(*) FROM ....
- We used PROC SURVEYSELECT to perform the actual sampling, which automatically adds two fields for the estimated selection probability and the record weight. Since these values are based on the assumption that only one sample is drawn from the population, we removed these two fields from the final samples.
- As mentioned, we assumed that for the purpose of sampling we would only operate on the row ID.

The implementation is included in the macro R2Samples(), for which the description follows.

Table 8.1 Parameters of macro R2samples().

Header	R2samples(S,IDVar,S1,N1,S2,N2,M_St);
Parameter	*Description*
S	Population dataset
IDVar	Row ID variable
S1	First random partition
N1	Size of first random partition
S2	Second random partition
N2	Size of second random partition
M_ST	Textual message to report errors and confirm result of data limitation constraint check

Step 1
Calculate the size of the population.

```
proc sql noprint;
  select count(*)into : N from &S;
  run;
quit;
```

Step 2
Check the data limitation constraint. If it is not satisfied, set the message variable and exit the macro without sampling.

```
%let Nx=%eval(&N1 + &N2);
%if &Nx > &N %then %do;
%let &M_st = Not enough records in population to
                generate samples. Sampling canceled. ;
%goto Exit;
                      %end;
/* Otherwise, OK */
%let &M_St=OK;
```

Step 3
Draw the first sample, S1, with size N1.

```
proc surveyselect noprint
     data =&S
     method = srs
     n= &N1
     out=&S1;
run;
```

Step 4
Append S1 with the selection indicator.

```
data &S1;
 set &S1;
  selected =1;
  keep &IDVar Selected;
run;
```

Step 5
Merge S1 with the population into TEMP.

```
proc sort data=&S;
 by &IDVar;
run;
proc sort data=&S1;
 by &IDVar;
run;
Data temp;
 merge &S &S1;
 by &IDvar;
  keep &IDVar Selected;
run;
```

Step 6
Draw the second sample, S2, with size N2 from unselected records in TEMP.

```
proc surveyselect noprint
      data =temp
      method = srs
      n=&N2
      out=&S2;
      where Selected NE 1;
run;
```

Step 7
Clean the workspace, S1 and S2.

```
Data &S1;
 set &S1;
 keep &IDvar;
run;
Data &S2;
 set &S2;
 keep &IDVar;
run;
proc datasets library=work nodetails;
 delete temp;
run;
quit;
```

Step 8
Finish the macro.

```
%exit: ;
%mend;
```

8.4 Balanced Sampling

In balanced sampling, we attempt to draw two samples from a population but with the composition of the dependent variable (DV) in each sample being different from that in the original population. For example, in a population of 100,000 banking customers, only 8% responded to a previous random campaign inviting them to open a new type of a savings account. This means that we have 8,000 responders (DV = 1) in this population dataset.

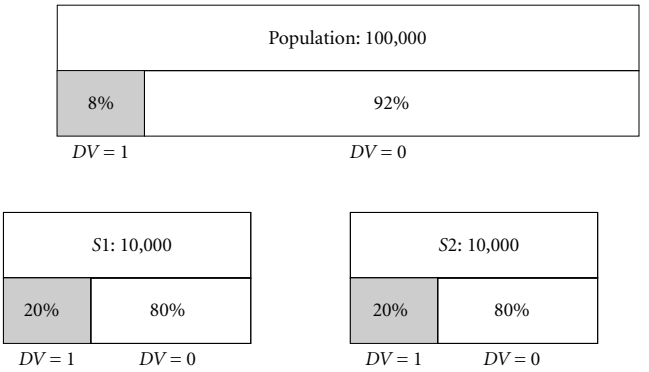

Figure 8.1 Balanced sampling.

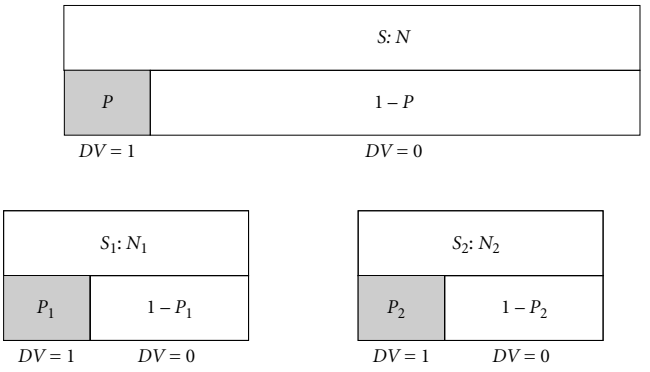

Figure 8.2 Data consistency constraints.

An example of balanced partitioning is when we draw two samples, of 10,000 records each, and with a response rate of 20%. This is illustrated in Figure 8.1.

8.4.1 Constraints on Sample Size

We now turn our attention to the problem of the size of each of the samples in balanced sampling. Given a population dataset S with a total of N records and a dependent variable with a proportion P of $DV = 1$. We seek two nonoverlapping balanced partitions S_1 and S_2 with N_1 and N_2 records, and P_1 and P_2 proportions of $DV = 1$, as shown in Figure 8.2.

In this case, the partitions must satisfy the following inequalities:

$$\begin{aligned} N_1 + N_2 &\leq N, \\ N_1 \bullet P_1 + N_2 \bullet P_2 &\leq N \bullet P, \\ N_1(1 - P_2) + N_2(1 - P_2) &\leq N(1 - P). \end{aligned} \tag{8.2}$$

It is easy to show that the first condition in Equation 8.3 is merely the summation of the second and third.

The first condition guarantees that the two samples do not contain more records than the original dataset. The second and third conditions guarantee that the two samples do not contain more records than the original dataset for each of the categories of the dependent variable. The three inequalities apply to the case of a dependent variable with two categories only, but they can be easily generalized to cases with more categories. We do not present these cases because the majority of classification models have binary dependent variables.

8.4.2 SAS Implementation

The implementation is included in the macro `B2Samples()`, as follows.

Table 8.2 Parameters of macro `B2samples()`.

Header	`B2samples(S,IDVar,DV,S1,N1,P1,S2,N2,P2,M_St);`
Parameter	*Description*
`S`	Population dataset
`IDVar`	Row ID variable
`S1`	First balanced partition
`N1`	Size of first random partition
`P1`	Percentage of DV in S1
`S2`	Second balanced partition
`N2`	Size of second partition
`P2`	Percentage of DV in S2
`M_ST`	Textual message to report errors and confirm result of data limitation constraint check

Step 1
Calculate `N`, `P` for the population.

```
proc sql noprint;
 select count(*)into : N from &S; /* Population size */
 select count(*) into : NP
                 from &S where &DV=1; /* Count of "1" */
```

```
 run;
quit;
%let NPc=%eval(&N - &NP);  /* Count of "0" (compliment)*/
```

Step 2
Check the consistency conditions.

```
%let Nx=%eval(&N1 + &N2);
%if &Nx > &N %then %do;
%let &M_st = Not enough records in population to generate
             samples. Sampling canceled. ;
%goto Exit;
 %end;
/* N1 P1 + N2 P2 <= N P */
%let Nx = %sysevalf((&N1*&P1+ &N2 * &P2), integer);
%if &Nx >&NP %then %do;
     %let &M_st = Count of DV=1 in requested samples
                  exceeds total count in population.
        Sampling canceled.;
 %goto Exit;
  %end;
/* N1(1-P1) + N2(1-P2) <= N(1-P)*/
%let Nx = %sysevalf((&N1*(1-&P1)+&N2*(1-&P2)),integer);
%if &Nx > &NPc %then %do;
     %let &M_st = Count of DV=0 in requested samples
                  exceeds total count in population.
                  Sampling canceled.;
 %goto Exit;
  %end;
/* Otherwise, OK */
%let &M_St=OK;
```

Step 3
Sort the population using the DV in ascending order.

```
proc sort data=&S;
by &DV;
run;
```

Step 4
Draw the sample S1 with size N1 and number of records N1P1, N1(1 - P1) in the strata 1,0 of the DV:

```
%let Nx1=%Sysevalf( (&N1*&P1),integer);
%let Nx0=%eval(&N1 - &Nx1);
proc surveyselect noprint
       data =&S
       method = srs
```

```
       n=( &Nx0 &Nx1)
       out=&S1;
       strata &DV;
run;
```

Step 5

Add a new field to S1—call it (Selected)—and give it a value of 1.

```
data &S1;
 set &S1;
  selected =1;
  keep &IDVar &DV Selected;
run;
```

Step 6

Merge S1 with the population S to find the already selected fields.

```
proc sort data=&S;
by &IDVar;
run;
proc sort data=&S1;
by &IDVar;
run;
Data temp;
 merge &S &S1;
 by &IDvar;
  keep &IDVar &DV Selected;
run;
```

Step 7

Draw the sample S2 with size N2 and number of records N2P2, $N2(1 - P2)$ in the strata 1,0 of the DV, under the condition that Selected is *not* 1.

```
proc sort data=temp;
by &DV;
run;
%let Nx1=%Sysevalf( (&N2*&P2),integer);
%let Nx0=%eval(&N2 - &Nx1);
proc surveyselect noprint
     data =temp
        method = srs
     n=( &Nx0 &Nx1)
out=&S2;
strata &DV;
where Selected NE 1;
run;
```

Step 8
Clean the workspace, S1 and S2, and finish the macro.

```
Data &S1;
 set &S1;
 keep &IDvar &DV;
run;

Data &S2;
 set &S2;
 keep &IDVar &DV;
run;

proc datasets library=work nodetails;
 delete temp;
run;
quit;

/*Label Exit: do nothing, end the macro)*/
  %exit: ;
  %mend;
```

Let us try to use the macro B2Samples() to extract two samples out of a population dataset. The first section of the code generates a population dataset of 100,000 records with a DV (conveniently called DV); then it implements the macro to extract two balanced partitions.

The variable DV in the population has a percentage of 10% of the category 1. On the other hand, the two requested samples will each have a total of 5,000 records and a DV=1 with a percentage of 50%.

```
/* First generate the population dataset */
   Data Population;
    do ID=1 to 100000;
      if ID <=10000 then DV=1;
          else DV=0;
   output;
   end;
   run;

/* Set the parameters needed to call the macro */
   %let S=Population;
   %let IDVar=ID;
   %let DV=DV;
   %let S1=Training;
   %let N1=5000;
   %let P1=0.50;
   %let S2=Validation;
```

```
   %let N2=5000;
   %let P2=0.50;
   %let Status=;

/* call the macro */
     %B2samples(&S,&IDVar,&DV,&S1,&N1,&P1,&S2,&N2,&P2,Status);

/* Display the status variable in the SAS log. */
 %put &status;
```

8.5 Minimum Sample Size

How large should the sample be? This is one of the most difficult questions in data mining, and one of the most asked! There are many anecdotal answers to that question, such as: *The sample size should be at least 10% of the population*. Unfortunately, these quick recipes do not withstand rigorous scrutiny. The answer is a bit more complicated than that.

Most real datasets used in data mining applications involve a combination, very often large, of nominal, ordinal, and continuous variables. A typical banking or retail dataset can contain many millions of records and hundreds of variables. Theoretical formulas providing rigorous answers to the sample size question, on the other hand, are available for very simplified cases. These formulas are mainly concerned with finding the minimum sample size such that the mean or the proportion of a variable is the same as that of the population with a specific confidence level. The use of these formulas in data mining is limited because real datasets are more complicated than the cases that have a theoretical estimate of the minimum sample size.

In the following, we present a summary of these simple cases. They can be used to check the validity of the sampling assumptions considering one variable at a time. Combining the results to check on the assumptions regarding the complete dataset is not possible without additional assumptions about the joint distribution of the variables. In most practical cases, it is not possible to infer much about the joint distribution of the variables.

We will summarize the formulas giving the minimum sample size for the different situations without proof. The complete discussion of these formulas can be found in Levy and Lemeshow (1999).

8.5.1 Continuous and Binary Variables

The following are two interesting situations where simple formulas exist for the calculation of sample size.

1. The average of a continuous variable. For example, the average age of the customers of a direct marketing company.

2. A value representing a proportion. This is typically the case of binary variables—for example, the default rate among credit card customers on their monthly payments.

Now let us introduce some notations. We denote the variable being sampled by x, and the proportion of a certain event in binary variables, say the proportion of the ones, by P_y.

The minimum sample size for estimating the mean of the population is given by

$$n \geq \frac{z^2 N V_x^2}{z^2 V_x^2 + (N-1)\varepsilon^2}, \tag{8.3}$$

where

- n is the sample size
- N is the population size
- V_x is the coefficient of variation of the continuous variable x defined as $V_x = \sigma_x^2 / \bar{X}^2$, with $\bar{X}$ and σ_x the mean and the standard deviation of the population, respectively
- z is the *reliability coefficient,* defined as the normalized z-value for the normal distribution function
- ε is the maximum acceptable percentage difference between the population average and the sample average

In the case of estimating the proportion of a binary variable, the minimum sample size is given by

$$n \geq \frac{z^2 N P_y (1 - P_y)}{(N-1)\varepsilon^2 P_y^2 + z^2 P_y (1 - P_y)}. \tag{8.4}$$

Let us demonstrate the use of the formulas using simple examples.

Example 8.1 The customer population of a telephone company is 10,000,000, for which we are building a cross-selling marketing campaign. We would like to calculate the minimum sample size on the basis of the monthly revenue per customer. We would like to keep a reliability level of 3.0, 99.87%, and accept a deviation of 5%. The average monthly revenue per customer in the entire population has been calculated to be $20.00, with a standard deviation of $6.50.

Using the preceding formulas, we have

$$V_x = \frac{\sigma_x^2}{\bar{X}^2} = \frac{6.5^2}{20^2} = 0.1056$$

$$n \geq \frac{z^2 N V_x^2}{z^2 V_x^2 + (N-1)\varepsilon^2} = \frac{(3)^2(10^7)(0.1056)^2}{(3)^2(0.1056)^2 + (10^7 - 1)(0.05)^2} \approx 41.$$

(The symbol "≈" indicates that the result was rounded to nearest integer.)

This number represents the minimum possible sample that would preserve the average value of the monthly revenue per customer. For most typical data mining, applications, this number would be too small because of the interactions among the many variables in the analysis.

◆

Example 8.2 For the population of the previous example, the average churn rate in the 10,000,000 customers was observed in the past and known to be an average of 4.5% per month. Using a similar reliability level of 3.0, 99.87%, and accepting a deviation of 5%, the minimum sample size to preserve the churn rate indicator would be calculated as follows:

$$n \geq \frac{3^2 10^7 (0.045)(1 - 0.045)}{(10^7 - 1)(0.05)^2(0.045)^2 + 3^2(0.045)(1 - 0.045)} \approx 75,821.$$

The result is *not* trivial.

◆

8.5.2 Sample Size for a Nominal Variable

The case of nominal variables with more than two categories can be handled as a general form of the case of binary variables. In this case we treat each of the categories as the event being sampled and calculate the minimum sample size. We then take the maximum of these sizes.

Note that this approach is based on transforming this nominal variable using 1 to N transformation and then using the resulting independent variables to calculate each minimum sample size. The 1 to N mapping is explained in detail in Chapter 9.

Let us demonstrate this process with an example.

Example 8.3 Consider again the telephone company with 10 million customer records. And assume further that the marital status variable of the customers has the categories and distributions shown in Table 8.3.

Table 8.3 Distribution of marital status categories.

Category	*Percentage* (%)
Single	22
Married	24
Divorced/separated	18
Widowed	10
Unknown	26

We can then define five new dummy variables for each category to calculate the equivalent sample size for each one using Equation 8.4. Using the same assumptions of a reliability of 99.87% and an acceptance error of 5%, the results would be as follows:

$$n_{\text{Single}} \approx 12{,}748$$
$$n_{\text{Married}} \approx 11{,}388$$
$$n_{\text{Divorced/separated}} \approx 16{,}374$$
$$n_{\text{Widowed}} \approx 32{,}296$$
$$n_{\text{Unknown}} \approx 10{,}236$$

The *minimum* sample size to have all the categories would be 32,296 records—again, a *nontrivial* result.
◆

8.6 Checking Validity of Samples

There are several considerations relating to the independent variables (IVs) and dependent variables (DVs) that should be taken into account when selecting a sample. These include:

- *IV range and categories:* Continuous IVs in the sample should have the same range as that of the population, and nominal IVs should have the same categories.
- *IV distribution:* The distribution of the values and categories of the IVs in the sample should be the same as (or at least close to that of) the population. This will not be true of a balanced sample because of the selective nature of the balanced sampling process.
- *DV categories:* The DV in the sample should have all the categories represented in the population. For example, if the DV is a risk indicator with three levels—low, medium, and high—then the sample should also have these three categories.

- *Stratification:* Sometimes it is necessary, or desirable, to divide the population into separate strata to allow for the effect of some measure that is believed to segment the population naturally into distinct groups. In this case, the usual measures to segment the population into such strata are geography, product ownership, customer value (based on simple measures such as total sales or product usage), and similar simple attributes.
- *Missing values:* The main issue with missing values is to guarantee that the frequency of occurrence of missing values in the different variables in the sample matches that of the population.

With the exception of selecting the sample through the different strata, the criteria for sample selection can be tested only after drawing the sample. It is therefore expected that the sampling process might have to be repeated several times before an acceptable sample is reached.