SAS Tutorial

Table of Contents

[1 Reading in Data and Basic Statistical Functions 6](#_Toc486232718)

[1.1 Read in the data. 6](#_Toc486232719)

[a) Read the data in as a .csv file. 6](#_Toc486232720)

[b) Read the data in as a .xls file. 6](#_Toc486232721)

[c) Read the data in as a .json file. 6](#_Toc486232722)

[1.2 Find the dimensions of the data set. 6](#_Toc486232723)

[1.3 Find basic information about the data set. 7](#_Toc486232724)

[1.4 Look at the first 5 observations. 8](#_Toc486232725)

[1.5 Calculate mean of numeric variables. 8](#_Toc486232726)

[1.6 Compute summary statistics of the data set. 9](#_Toc486232727)

[1.7 Descriptive statistics functions applied to columns of the data set. 9](#_Toc486232728)

[1.8 Produce a one-way table to describe the frequency of a variable. 10](#_Toc486232729)

[a) Produce a one-way table of a discrete variable. 10](#_Toc486232730)

[b) Produce a one-way table of a categorical variable. 10](#_Toc486232731)

[1.9 Produce a two-way table to visualize the frequency of two categorical (or discrete) variables. 10](#_Toc486232732)

[1.10 Select a subset of the data that meets a certain criterion. 11](#_Toc486232733)

[1.11 Determine the correlation between two continuous variables. 12](#_Toc486232734)

[2 Basic Graphing and Plotting Functions 12](#_Toc486232735)

[2.1 Visualize a single continuous variable by producing a histogram. 12](#_Toc486232736)

[2.2 Visualize a single continuous variable by producing a boxplot. 13](#_Toc486232737)

[2.3 Visualize two continuous variables by producing a scatterplot. 14](#_Toc486232738)

[2.4 Visualize a relationship between two continuous variables by producing a scatterplot and a plotted line of best fit. 15](#_Toc486232739)

[2.5 Visualize a categorical variable by producing a bar chart. 16](#_Toc486232740)

[2.6 Visualize a continuous variable, grouped by a categorical variable, using side-by-side boxplots. 17](#_Toc486232741)

[More advanced side-by-side boxplot with color. 17](#_Toc486232742)

[3 Basic Data Wrangling and Manipulation 18](#_Toc486232743)

[3.1 Create a new variable in a data set as a function of existing variables in the data set. 18](#_Toc486232744)

[3.2 Create a new variable in a data set using if/else logic of existing variables in the data set. 19](#_Toc486232745)

[3.3 Create a new variable in a data set using mathemtical functions applied to existing variables in the data set. 19](#_Toc486232746)

[3.4 Drop variables from a data set. 20](#_Toc486232747)

[3.5 Sort a data set by a variable. 20](#_Toc486232748)

[a) Sort data set by a continuous variable. 20](#_Toc486232749)

[b) Sort data set by a categorical variable. 21](#_Toc486232750)

[3.6 Compute descriptive statistics of continuous variables, grouped by a categorical variable. 21](#_Toc486232751)

[3.7 Add a new row to the bottom of a data set. 22](#_Toc486232752)

[3.8 Create a user-defined function and apply it to a variable in the data set to create a new variable in the data set. 22](#_Toc486232753)

[4 More Advanced Data Wrangling 23](#_Toc486232754)

[4.1 Drop observations with missing information. 23](#_Toc486232755)

[4.2 Merge two data sets together on a common variable. 25](#_Toc486232756)

[a) First, select specific columns of a data set to create two smaller data sets. 25](#_Toc486232757)

[b) Second, we want to merge the two smaller data sets on the common variable. 25](#_Toc486232758)

[c) Finally, we want to check to see if the merged data set is the same as the original data set. 26](#_Toc486232759)

[4.3 Merge two data sets together by index number only. 26](#_Toc486232760)

[a) First, select specific columns of a data set to create two smaller data sets. 26](#_Toc486232761)

[b) Second, we want to join the two smaller data sets. 27](#_Toc486232762)

[c) Finally, we want to check to see if the joined data set is the same as the original data set. 27](#_Toc486232763)

[4.4 Create a pivot table to summarize information about a data set. 27](#_Toc486232764)

[4.5 Return all unique values from a text variable. 28](#_Toc486232765)

[5 Preparation & Basic Regression 29](#_Toc486232766)

[5.1 Pre-process a data set using principal component analysis. 29](#_Toc486232767)

[5.2 Split data into training and testing data and export as a .csv file. 29](#_Toc486232768)

[5.3 Fit a logistic regression model. 30](#_Toc486232769)

[5.4 Fit a linear regression model. 32](#_Toc486232770)

[6 Regression & Machine Learning: Modeling & Prediction 32](#_Toc486232771)

[6.1 Fit a logistic regression model on training data and assess against testing data. 32](#_Toc486232772)

[a) Fit a logistic regression model on training data. 32](#_Toc486232773)

[b) Assess the model against the testing data. 34](#_Toc486232774)

[6.2 Fit a linear regression model on training data and assess against testing data. 35](#_Toc486232775)

[a) Fit a linear regression model on training data. 35](#_Toc486232776)

[b) Assess the model against the testing data. 36](#_Toc486232777)

[6.3 Fit a decision tree model on training data and assess against testing data. 37](#_Toc486232778)

[a) Fit a decision tree classification model. 37](#_Toc486232779)

[b) Fit a decision tree regression model. 41](#_Toc486232780)

[6.4 Fit a random forest model on training data and assess against testing data. 45](#_Toc486232781)

[a) Fit a random forest classification model. 45](#_Toc486232782)

[b) Fit a random forest regression model. 52](#_Toc486232783)

[6.5 Fit a gradient boosting model on training data and assess against testing data. 58](#_Toc486232784)

[a) Fit a gradient boosting classification model. 58](#_Toc486232785)

[b) Fit a gradient boosting regression model. 59](#_Toc486232786)

[6.6 Fit an extreme gradient boosting model on taining data and assess against testing data. 60](#_Toc486232787)

[a) Fit an extreme gradient boosting classification model. 60](#_Toc486232788)

[6.7 Fit a support vector model on training data and assess against testing data. 62](#_Toc486232789)

[a) Fit a support vector classification model. 62](#_Toc486232790)

[b) Fit a support vector regression model. 64](#_Toc486232791)

[6.8 Fit a neural network model on training data and assess against testing data. 64](#_Toc486232792)

[a) Fit a neural network classification model. 64](#_Toc486232793)

[b) Fit a neural network regression model. 67](#_Toc486232794)

[7 Unsupervised Machine Learning 68](#_Toc486232795)

[7.1 KMeans Clustering 68](#_Toc486232796)

[7.2 Spectral Clustering 71](#_Toc486232797)

[7.3 Ward Hierarchical Clustering 71](#_Toc486232798)

[7.4 DBSCAN 72](#_Toc486232799)

[7.5 Self-organizing map 72](#_Toc486232800)

[8 Forecasting 74](#_Toc486232801)

[8.1 Fit an ARIMA model to a timeseries. 74](#_Toc486232802)

[a) Plot the timeseries. 74](#_Toc486232803)

[b) Fit an ARIMA model, predict 2 years (24 months) out, and plot predictions. 75](#_Toc486232804)

[c) Plot residuals of predictions and known values. 80](#_Toc486232805)

[8.2 Fit a Simple Exponential Smoothing model to a timeseries. 81](#_Toc486232806)

[a) Plot the timeseries. 81](#_Toc486232807)

[b) Fit a Simple Exponential Smoothing model, predict 2 years (24 months) out and plot predictions. 82](#_Toc486232808)

[8.3 Fit a Holt-Winters model to a timeseries. 83](#_Toc486232809)

[a) Plot the timeseries. 83](#_Toc486232810)

[b) Fit a Holt-Winters additive model, predict 2 years (24 months) out and plot predictions. 84](#_Toc486232811)

[9 Model Evaluation & Selection 84](#_Toc486232812)

[9.1 Evaluate the accuracy of regression models. 84](#_Toc486232813)

[a) Evaluation on training data. 84](#_Toc486232814)

[b) Evaluation on testing data. 85](#_Toc486232815)

[9.2 Evaluate the accuracy of classification models. 86](#_Toc486232816)

[a) Evaluation on training data. 86](#_Toc486232817)

[b) Evaluation on testing data. 88](#_Toc486232818)

[9.3 Evaluation with cross validation. 89](#_Toc486232819)

[a) KFold 89](#_Toc486232820)

[b) ShuffleSplit 91](#_Toc486232821)

[10 Text Analytics 93](#_Toc486232822)

[11 Deep Learning 93](#_Toc486232823)

[Appendix 93](#_Toc486232824)

[1 Built-in SAS Data Types 93](#_Toc486232825)

[2 SAS Procedures 94](#_Toc486232826)

[COMPARE Procedure 94](#_Toc486232827)

[CONTENTS Procedure 94](#_Toc486232828)

[CORR Procedure 94](#_Toc486232829)

[FCMP Procedure 94](#_Toc486232830)

[EXPORT Procedure 94](#_Toc486232831)

[FREQ Procedure 94](#_Toc486232832)

[GENMOD Procedure 94](#_Toc486232833)

[HP4SCORE Procedure 94](#_Toc486232834)

[HPFOREST Procedure 94](#_Toc486232835)

[HPSPLIT Procedure 94](#_Toc486232836)

[HPSVM Procedure 94](#_Toc486232837)

[IML Procedure 94](#_Toc486232838)

[IMPORT Procedure 94](#_Toc486232839)

[MEANS Procedure 94](#_Toc486232840)

[PRINCOMP Procedure 94](#_Toc486232841)

[PRINT Procedure 94](#_Toc486232842)

[REG Procedure 94](#_Toc486232843)

[SCORE Procedure 94](#_Toc486232844)

[SGPLOT Procedure 94](#_Toc486232845)

[SGSCATTER Procedure 95](#_Toc486232846)

[SORT Procedure 95](#_Toc486232847)

[SQL Procedure 95](#_Toc486232848)

[SURVEYSELECT Procedure 95](#_Toc486232849)

[3 SAS DATA step 95](#_Toc486232850)

[%include 95](#_Toc486232851)

[if-then/else 95](#_Toc486232852)

[infile 95](#_Toc486232853)

[input 95](#_Toc486232854)

[merge 95](#_Toc486232855)

[output 95](#_Toc486232856)

[set 95](#_Toc486232857)

[where 95](#_Toc486232858)

[Alphabetical Index 95](#_Toc486232859)

[CHAR 95](#_Toc486232860)

[Data Frame 95](#_Toc486232861)

[Dictionary 95](#_Toc486232862)

[DOUBLE 96](#_Toc486232863)

[Series 96](#_Toc486232864)

Before beginning this tutorial, you need to insure that your [SAS environment is connected with an R environment](https://communities.sas.com/t5/General-SAS-Programming/Run-R-code-inside-SAS-easily/td-p/210116) so that the R code that SAS calls at the end of this tutorial from the [IML Procedure](#iml) run successfully.

In SAS,

\* This is a single line comment ;  
/\* This is a paragraph   
 comment \*/

# 1 Reading in Data and Basic Statistical Functions

## 1.1 Read in the data.

The [IMPORT Procedure](#import) is useful for reading in [SAS data sets](#DATASET) of a variety of different types.

### a) Read the data in as a .csv file.

proc import out = student  
 datafile = 'C:/Users/class.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;

### b) Read the data in as a .xls file.

proc import out = student\_xls  
 datafile = 'C:/Users/class.xls'  
 dbms = xls replace;  
 getnames = yes;  
run;

### c) Read the data in as a .json file.

There is more code involved in reading a .json file into SAS so that all the format is correct, however we will not at this time dive into the explanation for all this code, but please see the links below.

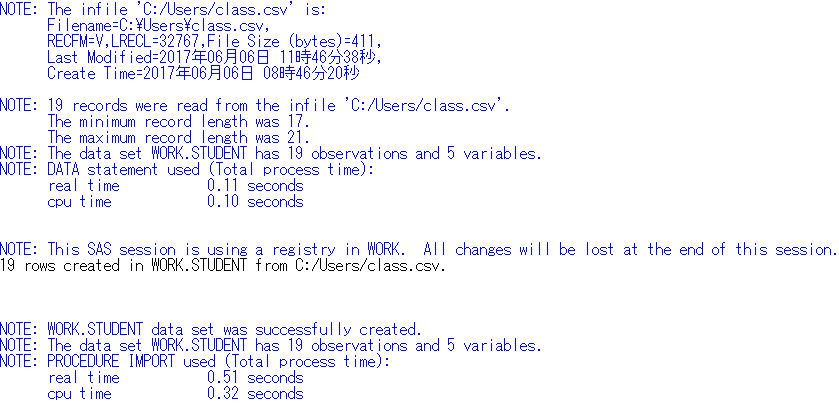
data student\_json;   
 INFILE 'C:/Users/class.json' LRECL = 3456677 TRUNCOVER SCANOVER   
 dsd   
 dlm=",}";   
 INPUT   
 @'"Name":' Name : $12.   
 @'"Sex":' Sex : $2.   
 @'"Age":' Age :   
 @'"Height":' Height :   
 @'"Weight":' Weight :   
 @@;   
run;

[DATA step](#step): [infile](#infile) & [input](#input) statements

## 1.2 Find the dimensions of the data set.

The shape of a SAS data set is available by running the [IMPORT Procedure](#import) and looking at the notes in the log file.

proc import out = student  
 datafile = 'C:/Users/class.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;

Output: 

## 1.3 Find basic information about the data set.

The [CONTENTS procedure](#contents) prints information about a [SAS data set](#DATASET).

proc contents data = student;  
run;

The CONTENTS Procedure  
  
Data Set Name WORK.STUDENT Observations 19  
Member Type DATA Variables 5   
Engine V9 Indexes 0   
Created 06/26/2017 09:25:27 Observation Length 32  
Last Modified 06/26/2017 09:25:27 Deleted Observations 0   
Protection Compressed NO  
Data Set Type Sorted NO  
Label   
Data Representation WINDOWS\_64   
Encoding wlatin1 Western (Windows)   
  
 Engine/Host Dependent Information  
  
Data Set Page Size 65536   
Number of Data Set Pages 1   
First Data Page 1   
Max Obs per Page 2039   
Obs in First Data Page 19   
Number of Data Set Repairs 0   
ExtendObsCounter YES   
Filename C:\Users\ElainePC\AppData\Local\Temp\SAS   
 Temporary   
 Files\\_TD4956\_ELAINEHP\_\student.sas7bdat   
Release Created 9.0401M4   
Host Created X64\_10PRO   
Owner Name ElaineHP\ElainePC   
File Size 128KB   
File Size (bytes) 131072   
  
 Alphabetic List of Variables and Attributes  
   
 # Variable Type Len Format Informat  
  
 3 Age Num 8 BEST12. BEST32.   
 4 Height Num 8 BEST12. BEST32.   
 1 Name Char 7 $7. $7.   
 2 Sex Char 1 $1. $1.   
 5 Weight Num 8 BEST12. BEST32.

## 1.4 Look at the first 5 observations.

The [PRINT procedure](#print) prints a [SAS data set](#DATASET), according to the specifications and options provided.

/\* obs= option tells SAS how many observations to print, starting  
 with the first observation \*/  
proc print data = student (obs=5);  
run;

Obs Name Sex Age Height Weight  
  
 1 Alfred M 14 69 112.5  
 2 Alice F 13 56.5 84  
 3 Barbara F 13 65.3 98  
 4 Carol F 14 62.8 102.5  
 5 Henry M 14 63.5 102.5

## 1.5 Calculate mean of numeric variables.

The [MEANS procedure](#means) prints the mean of all numeric variables of a [SAS data set](#DATASET), as well as other descriptive statistics.

proc means data = student mean;  
run;

The MEANS Procedure  
  
 Variable Mean  
 ------------------------  
 Age 13.3157895  
 Height 62.3368421  
 Weight 100.0263158  
 ------------------------

## 1.6 Compute summary statistics of the data set.

Summary statistics of a [SAS data set](#DATASET) are available by running the [MEANS procedure](#means) and specifying statistics to return.

/\* SAS uses a different method than Python and R to compute  
 quartiles, but the method in each language can be changed \*/  
/\* maxdec= option tells SAS to print at most 2 numbers behind  
 the decimal point \*/  
proc means data = student min q1 median mean q3 max n maxdec=2;   
run;

The MEANS Procedure  
  
 Lower  
 Variable Minimum Quartile Median Mean  
 ------------------------------------------------------------------------  
 Age 11.00 12.00 13.00 13.32  
 Height 51.30 57.50 62.80 62.34  
 Weight 50.50 84.00 99.50 100.03  
 ------------------------------------------------------------------------  
  
 Upper  
 Variable Quartile Maximum N  
 ----------------------------------------------  
 Age 15.00 16.00 19  
 Height 66.50 72.00 19  
 Weight 112.50 150.00 19  
 ----------------------------------------------

## 1.7 Descriptive statistics functions applied to columns of the data set.

/\* The var statement tells SAS which variable to use for the  
 procedure \*/  
proc means data = student stddev sum n max min median maxdec=2;   
 var Weight;   
run;

The MEANS Procedure  
  
 Analysis Variable : Weight   
   
 Std Dev Sum N Maximum Minimum Median  
 ------------------------------------------------------------------------  
 22.77 1900.50 19 150.00 50.50 99.50  
 ------------------------------------------------------------------------

## 1.8 Produce a one-way table to describe the frequency of a variable.

The [FREQ procedure](#freq) prints the frequency of categorical or discrete variables of a [SAS data set](#DATASET).

### a) Produce a one-way table of a discrete variable.

proc freq data = student;   
 tables Age / nopercent norow nocol;   
run;

The FREQ Procedure  
  
 Cumulative  
 Age Frequency Frequency  
 ------------------------------  
 11 2 2   
 12 5 7   
 13 3 10   
 14 4 14   
 15 4 18   
 16 1 19

### b) Produce a one-way table of a categorical variable.

proc freq data = student;   
 tables Sex / nopercent norow nocol;   
run;

The FREQ Procedure  
  
 Cumulative  
 Sex Frequency Frequency  
 ------------------------------  
 F 9 9   
 M 10 19

The tables statement allows you to specify multiple variables at once, separated only by a space, so both of these tables could have been created with one [FREQ procedure](#freq) call. The options on the tables statement (nopercent norow nocol) prevent SAS from printing percents in the table, which are printed by default.

**TRY THIS AT HOME**: Run this procedure without the options on the tables statment.

## 1.9 Produce a two-way table to visualize the frequency of two categorical (or discrete) variables.

/\* The "\*" between two variables on the tables statement  
 indicates to produce a two-way table of the two variables \*/  
proc freq data = student;   
 tables Age\*Sex / nopercent norow nocol;   
run;

The FREQ Procedure  
  
 Table of Age by Sex  
  
 Age Sex  
  
 Frequency|F |M | Total  
 ---------+--------+--------+  
 11 | 1 | 1 | 2  
 ---------+--------+--------+  
 12 | 2 | 3 | 5  
 ---------+--------+--------+  
 13 | 2 | 1 | 3  
 ---------+--------+--------+  
 14 | 2 | 2 | 4  
 ---------+--------+--------+  
 15 | 2 | 2 | 4  
 ---------+--------+--------+  
 16 | 0 | 1 | 1  
 ---------+--------+--------+  
 Total 9 10 19

[FREQ Procedure](#freq)

## 1.10 Select a subset of the data that meets a certain criterion.

The [SAS DATA step](#step) is used for all things data manipulation and in Section 2 we will explore it further.

data females;   
 set student;   
 where Sex = "F";   
run;   
proc print data = females(obs=5);   
run;

Obs Name Sex Age Height Weight  
  
 1 Alice F 13 56.5 84  
 2 Barbara F 13 65.3 98  
 3 Carol F 14 62.8 102.5  
 4 Jane F 12 59.8 84.5  
 5 Janet F 15 62.5 112.5

[DATA step](#step): [set](#set) & [where](#where) statements

**TRY THIS AT HOME**: Run this procedure to return all *male* students.

## 1.11 Determine the correlation between two continuous variables.

/\* The nosimple option reduces the output of this procedure \*/  
proc corr data = student pearson nosimple;  
var Height Weight;  
run;

The CORR Procedure  
  
 2 Variables: Height Weight   
  
 Pearson Correlation Coefficients, N = 19   
 Prob > |r| under H0: Rho=0  
   
 Height Weight  
  
 Height 1.00000 0.87779  
 <.0001  
  
 Weight 0.87779 1.00000  
 <.0001

[CORR Procedure](#corr)

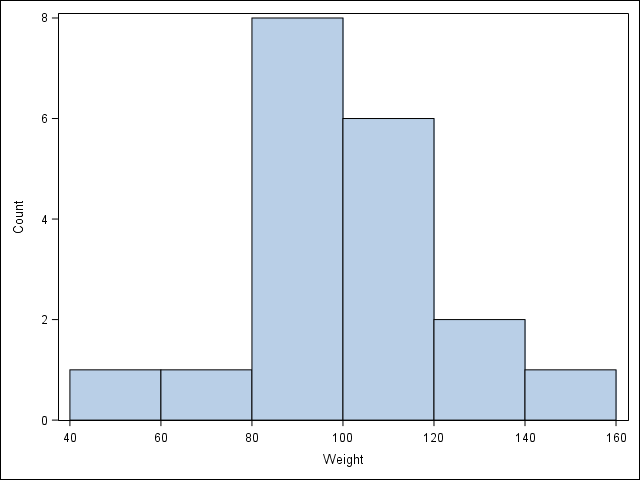
**TRY THIS AT HOME**: Run this procedure and do not reduce the output.

# 2 Basic Graphing and Plotting Functions

The [SGPLOT procedure](#sgplot) is a very useful SAS procedure for producing plots from data. For more information on other statements within the SGPLOT procedure, please see the Appendix Section 2.

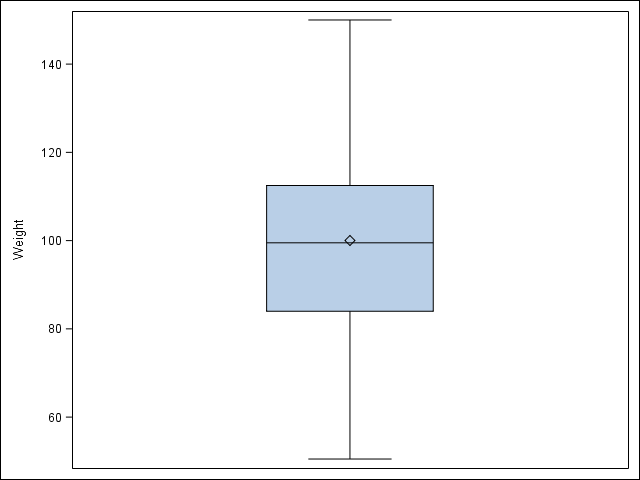
## 2.1 Visualize a single continuous variable by producing a histogram.

/\* Notice the specifcation of the bins, as well as the xaxis values \*/  
/\* SAS denotes "count" as what R & Python denote as "frequency" \*/  
proc sgplot data = student;  
 histogram weight / binwidth=20 binstart=40 scale=count;  
 xaxis values=(40 to 160 by 20);  
run;

Output: 

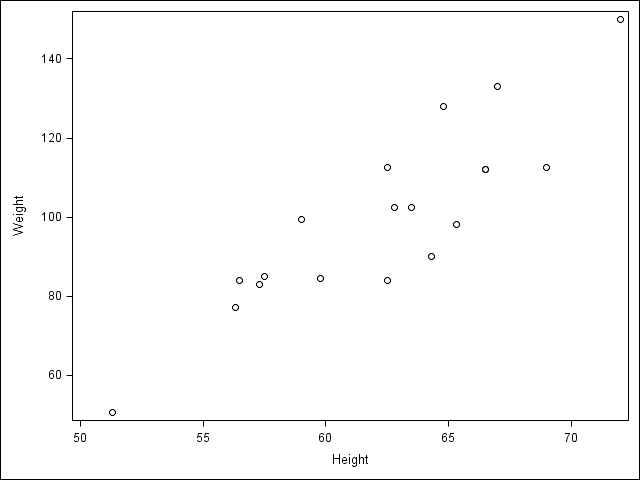
## 2.2 Visualize a single continuous variable by producing a boxplot.

/\* SAS automatically prints the mean on the boxplot \*/  
proc sgplot data = student;   
 vbox Weight;   
run;

Output: 

## 2.3 Visualize two continuous variables by producing a scatterplot.

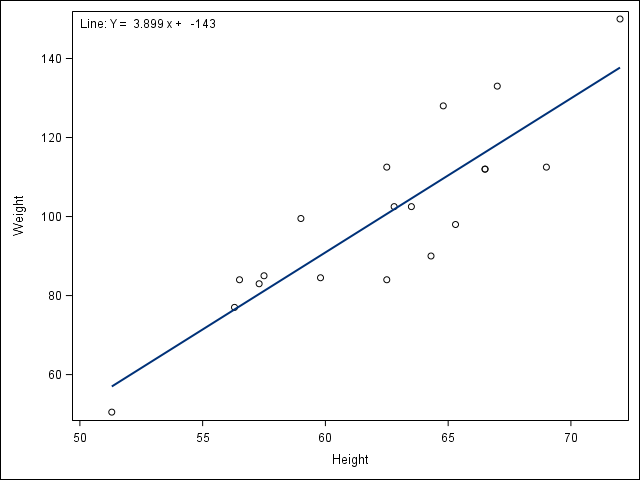
/\* Notice here you specify the y variable followed by the x variable \*/  
proc sgscatter data = student;   
 plot Weight \* Height;   
run;

Output: 

[SGSCATTER Procedure](#sgscatter)

## 2.4 Visualize a relationship between two continuous variables by producing a scatterplot and a plotted line of best fit.

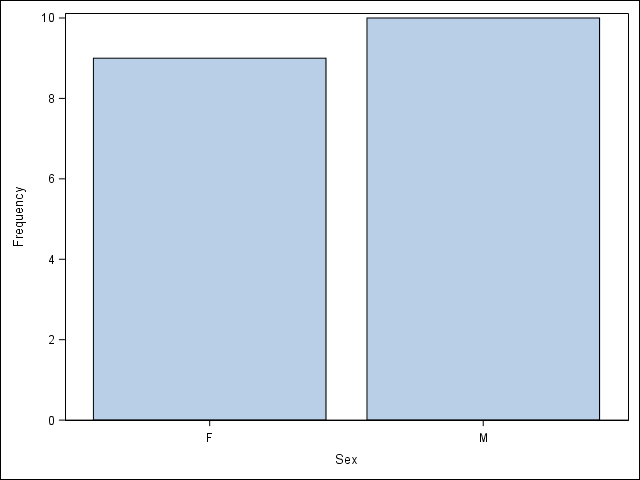
/\* Use proc reg to get the parameter estimates for the line of best fit,   
 but don't print the graph (ods graphics off) \*/  
ods graphics off;   
proc reg data = student;   
 /\* Syntax indicates Weight as a function of Height \*/  
 model Weight = Height;   
 ods output ParameterEstimates=PE;   
run;   
ods graphics on;   
  
/\* data \_null\_ indicates to not create a data set, but  
 run the code within the data step to create macro   
 variables to store the parameter estimates \*/  
data \_null\_;   
 set PE;   
 if \_n\_=1 then call symput('Int', put(estimate, BEST6.));   
 else call symput('Slope', put(estimate, BEST6.));   
run;   
  
/\* Use proc sgplot with the reg statement so it prints the line of best fit,   
 and use the inset statement to print the equation of the line   
 of best fit \*/  
proc sgplot data = student noautolegend;   
 reg y = Weight x = Height;   
 inset "Line: Y = &Slope x + &Int" / position=topleft;   
run;

Output: 

[REG Procedure](#reg) | [DATA step](#step): [set](#set) statement | [macro variables](https://v8doc.sas.com/sashtml/macro/z1071889.htm) | [call symput()](https://v8doc.sas.com/sashtml/macro/z0210266.htm) function

## 2.5 Visualize a categorical variable by producing a bar chart.

/\* Notice here you must first sort by Sex and then plot the vertical   
 bar chart \*/  
proc sort data = student;   
 by Sex;   
run;   
proc sgplot data = student;   
 vbar Sex;   
run;

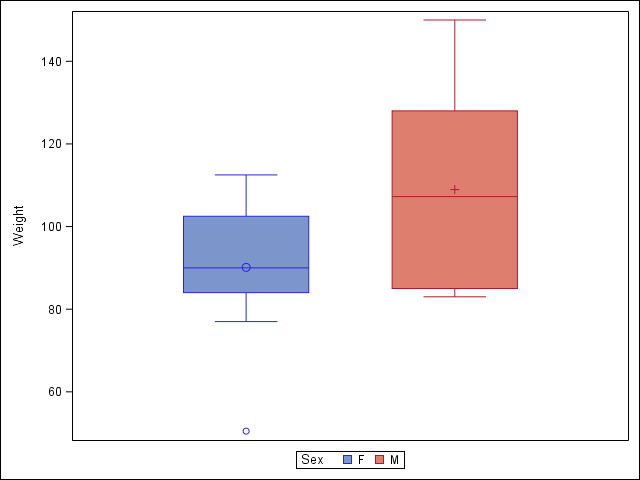
Output: 

[SORT Procedure](#sort)

## 2.6 Visualize a continuous variable, grouped by a categorical variable, using side-by-side boxplots.

### More advanced side-by-side boxplot with color.

proc sgplot data = student;   
 vbox Weight / group=Sex;   
run;

Output: 

# 3 Basic Data Wrangling and Manipulation

Many of the following examples make use of the [SAS DATA step](#step) for manipulating and altering data sets, and a main part of the DATA step is the [set](#set) statement.

## 3.1 Create a new variable in a data set as a function of existing variables in the data set.

data student;   
 set student;   
 BMI = Weight / (Height\*\*2) \* 703;   
run;   
proc print data = student(obs=5);   
run;

Obs Name Sex Age Height Weight BMI  
  
 1 Alfred M 14 69 112.5 16.6115  
 2 Alice F 13 56.5 84 18.4986  
 3 Barbara F 13 65.3 98 16.1568  
 4 Carol F 14 62.8 102.5 18.2709  
 5 Henry M 14 63.5 102.5 17.8703

## 3.2 Create a new variable in a data set using if/else logic of existing variables in the data set.

data student;   
 set student;   
 if (BMI < 19.0) then BMI\_class = "Underweight";   
 else BMI\_class = "Healthy";   
run;   
proc print data = student(obs=5);   
run;

Obs Name Sex Age Height Weight BMI BMI\_class  
  
 1 Alfred M 14 69 112.5 16.6115 Underweight  
 2 Alice F 13 56.5 84 18.4986 Underweight  
 3 Barbara F 13 65.3 98 16.1568 Underweight  
 4 Carol F 14 62.8 102.5 18.2709 Underweight  
 5 Henry M 14 63.5 102.5 17.8703 Underweight

[if-then/else](#if) statement

## 3.3 Create a new variable in a data set using mathemtical functions applied to existing variables in the data set.

Using the log() function, the exp() function, the sqrt() function, and the abs() function.

data student;   
 set student;   
 LogWeight = log(Weight);   
 ExpAge = exp(Age);   
 SqrtHeight = sqrt(Height);   
 if (BMI < 19.0) then BMI\_Neg = -BMI;   
 else BMI\_Neg = BMI;   
 BMI\_Pos = abs(BMI\_Neg);   
 /\* Create a boolean variable, which is handled differently  
 in SAS than in Python and R \*/  
 BMI\_Check = (BMI\_Pos = BMI);   
run;   
proc print data = student(obs=5);   
run;

Obs Name Sex Age Height Weight BMI BMI\_class  
  
 1 Alfred M 14 69 112.5 16.6115 Underweight  
 2 Alice F 13 56.5 84 18.4986 Underweight  
 3 Barbara F 13 65.3 98 16.1568 Underweight  
 4 Carol F 14 62.8 102.5 18.2709 Underweight  
 5 Henry M 14 63.5 102.5 17.8703 Underweight  
  
 Log Sqrt BMI\_  
Obs Weight ExpAge Height BMI\_Neg BMI\_Pos Check  
  
 1 4.72295 1202604.28 8.30662 -16.6115 16.6115 1   
 2 4.43082 442413.39 7.51665 -18.4986 18.4986 1   
 3 4.58497 442413.39 8.08084 -16.1568 16.1568 1   
 4 4.62986 1202604.28 7.92465 -18.2709 18.2709 1   
 5 4.62986 1202604.28 7.96869 -17.8703 17.8703 1

[if-then/else](#if) statement

[log()](http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p0urbseuxrkrlyn1tr04y30nt25s.htm), [exp()](http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#n0ocutx9jgosdln17xg1z75jmkbv.htm), [sqrt()](http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#n0uc20qbw3wi2jn1y1tan8rq8mnm.htm), & [abs()](http://support.sas.com/documentation/cdl/en/lefunctionsref/63354/HTML/default/viewer.htm#p0xkrj83an7dknn1sgukpmnphcje.htm) functions

## 3.4 Drop variables from a data set.

data student;   
 set student (drop = LogWeight ExpAge SqrtHeight BMI\_Neg BMI\_Pos BMI\_Check);   
run;   
proc print data = student(obs=5);   
run;

Obs Name Sex Age Height Weight BMI BMI\_class  
  
 1 Alfred M 14 69 112.5 16.6115 Underweight  
 2 Alice F 13 56.5 84 18.4986 Underweight  
 3 Barbara F 13 65.3 98 16.1568 Underweight  
 4 Carol F 14 62.8 102.5 18.2709 Underweight  
 5 Henry M 14 63.5 102.5 17.8703 Underweight

[drop=](http://support.sas.com/documentation/cdl/en/ledsoptsref/63326/HTML/default/viewer.htm#n15goor3q758g5n1eykstufkpdhy.htm) data set option

## 3.5 Sort a data set by a variable.

### a) Sort data set by a continuous variable.

proc sort data = student;   
 by Age;   
run;   
proc print data = student(obs=5);   
run;

Obs Name Sex Age Height Weight BMI BMI\_class  
  
 1 Joyce F 11 51.3 50.5 13.4900 Underweight  
 2 Thomas M 11 57.5 85 18.0733 Underweight  
 3 James M 12 57.3 83 17.7715 Underweight  
 4 Jane F 12 59.8 84.5 16.6115 Underweight  
 5 John M 12 59 99.5 20.0944 Healthy

[SORT Procedure](#sort)

### b) Sort data set by a categorical variable.

proc sort data = student;   
 by Sex;   
run;   
/\* Notice that the data is now sorted first by Sex and  
 then within Sex by Age \*/  
proc print data = student(obs=5);   
run;

Obs Name Sex Age Height Weight BMI BMI\_class  
  
 1 Joyce F 11 51.3 50.5 13.4900 Underweight  
 2 Jane F 12 59.8 84.5 16.6115 Underweight  
 3 Louise F 12 56.3 77 17.0777 Underweight  
 4 Alice F 13 56.5 84 18.4986 Underweight  
 5 Barbara F 13 65.3 98 16.1568 Underweight

[SORT Procedure](#sort)

## 3.6 Compute descriptive statistics of continuous variables, grouped by a categorical variable.

proc means data = student mean;   
 by Sex;   
 var Age Height Weight BMI;   
run;

---------------------------------- Sex=F ----------------------------------  
  
 The MEANS Procedure  
  
 Variable Mean  
 ------------------------  
 Age 13.2222222  
 Height 60.5888889  
 Weight 90.1111111  
 BMI 17.0510391  
 ------------------------  
  
  
---------------------------------- Sex=M ----------------------------------  
  
 Variable Mean  
 ------------------------  
 Age 13.4000000  
 Height 63.9100000  
 Weight 108.9500000  
 BMI 18.5942434  
 ------------------------

[MEANS Procedure](#means)

## 3.7 Add a new row to the bottom of a data set.

/\* Look at the tail of the data currently \*/  
proc print data = student(firstobs=15);   
run;

Obs Name Sex Age Height Weight BMI BMI\_class  
  
 15 Alfred M 14 69 112.5 16.6115 Underweight  
 16 Henry M 14 63.5 102.5 17.8703 Underweight  
 17 Ronald M 15 67 133 20.8285 Healthy   
 18 William M 15 66.5 112 17.8045 Underweight  
 19 Philip M 16 72 150 20.3414 Healthy

data student;   
 set student end = eof;   
 output;   
 if eof then do;   
 Name = 'Jane';   
 Sex = 'F';   
 Age = 14;   
 Height = 56.3;   
 Weight = 77.0;   
 BMI = 17.077695;   
 BMI\_Class = 'Underweight';   
 output;   
 end;   
run;   
proc print data = student(firstobs=16);   
run;

Obs Name Sex Age Height Weight BMI BMI\_class  
  
 16 Henry M 14 63.5 102.5 17.8703 Underweight  
 17 Ronald M 15 67 133 20.8285 Healthy   
 18 William M 15 66.5 112 17.8045 Underweight  
 19 Philip M 16 72 150 20.3414 Healthy   
 20 Jane F 14 56.3 77 17.0777 Underweight

[if-then/else](#if) & [output] statements, [do loop](#do), [end=](#step) & [firstobs=](http://support.sas.com/documentation/cdl/en/ledsoptsref/69751/HTML/default/viewer.htm#p0wjxoxrco6dsgn1ls5n3mbybcng.htm) data set options

## 3.8 Create a user-defined function and apply it to a variable in the data set to create a new variable in the data set.

proc fcmp outlib=sasuser.userfuncs.myfunc;   
 function toKG(lb);   
 kg = 0.45359237 \* lb;   
 return(kg);   
endsub;  
  
options cmplib=sasuser.userfuncs;   
  
data studentKG;   
 set student;   
 Weight\_KG = toKG(Weight);   
run;  
  
proc print data = studentKG(obs=5);  
run;

Obs Name Sex Age Height Weight  
  
 1 Joyce F 11 51.3 50.5  
 2 Jane F 12 59.8 84.5  
 3 Louise F 12 56.3 77  
 4 Alice F 13 56.5 84  
 5 Barbara F 13 65.3 98  
  
 Weight\_  
 Obs BMI BMI\_class KG  
  
 1 13.4900 Underweight 22.9064  
 2 16.6115 Underweight 38.3286  
 3 17.0777 Underweight 34.9266  
 4 18.4986 Underweight 38.1018  
 5 16.1568 Underweight 44.4521

[FCMP Procedure](#fcmp)

# 4 More Advanced Data Wrangling

## 4.1 Drop observations with missing information.

/\* Notice the use of the fish data set because it has some missing   
 observations \*/  
proc import out = fish   
 datafile='C:/Users/fish.csv'   
 dbms = csv replace;   
 getnames = yes;   
run;   
  
/\* First sort by Weight, requesting those with NA for Weight first,  
 which SAS does automatically \*/  
proc sort data = fish;   
 by Weight;   
run;   
proc print data = fish(obs=5);   
run;

Obs Species Weight Length1 Length2  
  
 1 Bream . 29.5 32  
 2 Roach 0 19 20.5  
 3 Perch 5.9 7.5 8.4  
 4 Smelt 6.7 9.3 9.8  
 5 Smelt 7 10.1 10.6  
  
 Obs Length3 Height Width  
  
 1 37.3 13.9129 5.0728  
 2 22.8 6.4752 3.3516  
 3 8.8 2.112 1.408  
 4 10.8 1.7388 1.0476  
 5 11.6 1.7284 1.1484

data new\_fish;   
 set fish;   
 /\* Notice the not-equal operator (^=) and how SAS denotes  
 missing values (.) \*/  
 if (Weight ^= .);   
run;   
proc print data = new\_fish(obs=5);   
run;

Obs Species Weight Length1 Length2  
  
 1 Roach 0 19 20.5  
 2 Perch 5.9 7.5 8.4  
 3 Smelt 6.7 9.3 9.8  
 4 Smelt 7 10.1 10.6  
 5 Smelt 7.5 10 10.5  
  
 Obs Length3 Height Width  
  
 1 22.8 6.4752 3.3516  
 2 8.8 2.112 1.408  
 3 10.8 1.7388 1.0476  
 4 11.6 1.7284 1.1484  
 5 11.6 1.972 1.16

[SORT Procedure](#sort), [if-then/else](#if) statement

## 4.2 Merge two data sets together on a common variable.

### a) First, select specific columns of a data set to create two smaller data sets.

/\* Notice the use of the student data set again, however we want to reload it   
 without the changes we've made previously \*/  
proc import out = student  
 datafile = 'C:/Users/class.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
data student1;   
 set student(keep= Name Sex Age);   
run;   
proc print data = student1(obs=5);   
run;

Obs Name Sex Age  
  
 1 Alfred M 14  
 2 Alice F 13  
 3 Barbara F 13  
 4 Carol F 14  
 5 Henry M 14

data student2;   
 set student(keep= Name Height Weight);   
run;   
proc print data = student2(obs=5);   
run;

Obs Name Height Weight  
  
 1 Alfred 69 112.5  
 2 Alice 56.5 84  
 3 Barbara 65.3 98  
 4 Carol 62.8 102.5  
 5 Henry 63.5 102.5

[keep=](http://support.sas.com/documentation/cdl/en/ledsoptsref/69751/HTML/default/viewer.htm#p0vw9lyyxk1cxkn0zzfemrsr3t9a.htm) data set option

### b) Second, we want to merge the two smaller data sets on the common variable.

data new;   
 merge student1 student2;   
 by Name;   
run;   
proc print data = new(obs=5);   
run;

Obs Name Sex Age Height Weight  
  
 1 Alfred M 14 69 112.5  
 2 Alice F 13 56.5 84  
 3 Barbara F 13 65.3 98  
 4 Carol F 14 62.8 102.5  
 5 Henry M 14 63.5 102.5

[merge](#merge) statement

### c) Finally, we want to check to see if the merged data set is the same as the original data set.

proc compare base = student compare = new brief;   
run;

The COMPARE Procedure  
 Comparison of WORK.STUDENT with WORK.NEW  
 (Method=EXACT)  
  
NOTE: No unequal values were found. All values compared are exactly equal.

[COMPARE Procedure](#compare)

## 4.3 Merge two data sets together by index number only.

### a) First, select specific columns of a data set to create two smaller data sets.

data newstudent1;   
 set student(keep= Name Sex Age);   
run;   
proc print data = newstudent1(obs=5);   
run;

Obs Name Sex Age  
  
 1 Alfred M 14  
 2 Alice F 13  
 3 Barbara F 13  
 4 Carol F 14  
 5 Henry M 14

data newstudent2;   
 set student(keep= Height Weight);   
run;   
proc print data = newstudent2(obs=5);   
run;

Obs Height Weight  
  
 1 69 112.5  
 2 56.5 84  
 3 65.3 98  
 4 62.8 102.5  
 5 63.5 102.5

[keep=](http://support.sas.com/documentation/cdl/en/ledsoptsref/69751/HTML/default/viewer.htm#p0vw9lyyxk1cxkn0zzfemrsr3t9a.htm) data set option

### b) Second, we want to join the two smaller data sets.

data new2;   
 merge newstudent1 newstudent2;   
run;   
proc print data = new2(obs=5);   
run;

Obs Name Sex Age Height Weight  
  
 1 Alfred M 14 69 112.5  
 2 Alice F 13 56.5 84  
 3 Barbara F 13 65.3 98  
 4 Carol F 14 62.8 102.5  
 5 Henry M 14 63.5 102.5

[merge](#merge) statement

### c) Finally, we want to check to see if the joined data set is the same as the original data set.

proc compare base = student compare = new2 brief;   
run;

The COMPARE Procedure  
 Comparison of WORK.STUDENT with WORK.NEW2  
 (Method=EXACT)  
  
NOTE: No unequal values were found. All values compared are exactly equal.

## 4.4 Create a pivot table to summarize information about a data set.

/\* Notice we are using a new data set that needs to be read into the   
 environment \*/  
proc import out = price  
 datafile = 'C:/Users/price.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
/\* The following code is used to remove the "," and "$" characters from the   
 ACTUAL column so that values can be summed \*/  
data price;  
 set price;  
 num\_actual = input(actual, dollar10.);  
run;  
  
proc sql;   
 create table categorysales as   
 select country, state, prodtype,   
 product, sum(num\_actual) as REVENUE   
 from price   
 group by country, state, prodtype, product;   
quit;   
proc print data = categorysales(obs=5);   
run;

Obs COUNTRY STATE PRODTYPE PRODUCT REVENUE  
  
 1 Canada British Co FURNITURE BED 197706.6  
 2 Canada British Co FURNITURE SOFA 216282.6  
 3 Canada British Co OFFICE CHAI 200905.2  
 4 Canada British Co OFFICE DESK 186262.2  
 5 Canada Ontario FURNITURE BED 194493.6

[input()](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000180357.htm) function, [SQL Procedure](#sql)

## 4.5 Return all unique values from a text variable.

proc iml;  
 use price;  
 read all var {STATE};  
 close price;  
   
 unique\_states = unique(STATE);  
 print(unique\_states);  
quit;

unique\_states  
 COL1 COL2 COL3 COL4 COL5 COL6  
  
 ROW1 Baja Calif British Co California Campeche Colorado Florida   
  
 unique\_states  
 COL7 COL8 COL9 COL10 COL11 COL12  
  
 ROW1 Illinois Michoacan New York North Caro Nuevo Leon Ontario   
  
 unique\_states  
 COL13 COL14 COL15 COL16  
  
 ROW1 Quebec Saskatchew Texas Washington

# 5 Preparation & Basic Regression

## 5.1 Pre-process a data set using principal component analysis.

/\* Notice we are using a new data set that needs to be read into the   
 environment \*/  
proc import out = iris  
 datafile = 'C:/Users/iris.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
data features;  
set iris(drop=Target);  
run;  
  
proc princomp data = features noprint outstat = feat\_princomp;  
var SepalLength SepalWidth PetalLength PetalWidth;  
run;  
  
data eigenvectors;  
 set feat\_princomp;  
 where \_TYPE\_ = "SCORE";  
run;  
proc print data = eigenvectors;  
run;

Sepal Sepal Petal Petal  
 Obs \_TYPE\_ \_NAME\_ Length Width Length Width  
  
 1 SCORE Prin1 0.52237 -0.26335 0.58125 0.56561  
 2 SCORE Prin2 0.37232 0.92556 0.02109 0.06542  
 3 SCORE Prin3 -0.72102 0.24203 0.14089 0.63380  
 4 SCORE Prin4 -0.26200 0.12413 0.80115 -0.52355

[drop=](http://support.sas.com/documentation/cdl/en/ledsoptsref/63326/HTML/default/viewer.htm#n15goor3q758g5n1eykstufkpdhy.htm) data set option, [PRINCOMP Procedure](#princomp)

## 5.2 Split data into training and testing data and export as a .csv file.

/\* outall option tells SAS to add a flag showing which observations were   
 chosen \*/  
/\* seed = 29 specifies the seed for random values so the results are   
 reproducible \*/  
proc surveyselect data = iris outall out = all method = srs samprate = 0.7   
 seed = 29;  
run;  
  
data train (drop = selected);  
 set all;  
 where (selected = 1);  
run;  
data test (drop = selected);  
 set all;  
 where (selected = 0);  
run;  
  
proc export data = train  
 outfile = 'C:\Users\iris\_train.csv'  
 dbms = csv;  
run;  
proc export data = test  
 outfile = 'C:\Users\iris\_test.csv'  
 dbms = csv;  
run;

[SURVEYSELECT Procedure](#surveyselect), [drop=](http://support.sas.com/documentation/cdl/en/ledsoptsref/63326/HTML/default/viewer.htm#n15goor3q758g5n1eykstufkpdhy.htm) data set option, [EXPORT Procedure](#export)

## 5.3 Fit a logistic regression model.

/\* Notice we are using a new data set that needs to be read into the   
 environment \*/  
proc import out = tips  
 datafile = 'C:/Users/tips.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
  
/\* The following code is used to determine if the individual left more than  
 a 15% tip \*/  
data tips;   
 set tips;   
 if (tip > 0.15\*total\_bill) then greater15 = 1;   
 else greater15 = 0;   
run;   
  
/\* The descending option tells SAS to model the probability that   
 greater15 = 1 \*/  
proc genmod data=tips descending;   
 model greater15 = total\_bill / dist = bin link = logit lrci;   
run;

The GENMOD Procedure  
  
 Model Information  
  
 Data Set WORK.TIPS  
 Distribution Binomial  
 Link Function Logit  
 Dependent Variable greater15  
  
 Number of Observations Read 244  
 Number of Observations Used 244  
 Number of Events 135  
 Number of Trials 244  
  
 Response Profile  
   
 Ordered Total  
 Value greater15 Frequency  
  
 1 1 135  
 2 0 109  
  
PROC GENMOD is modeling the probability that greater15='1'.  
  
 Criteria For Assessing Goodness Of Fit  
   
 Criterion DF Value Value/DF  
  
 Log Likelihood -156.8714   
 Full Log Likelihood -156.8714   
 AIC (smaller is better) 317.7428   
 AICC (smaller is better) 317.7926   
 BIC (smaller is better) 324.7371   
  
  
Algorithm converged.   
  
 Analysis Of Maximum Likelihood Parameter Estimates  
   
 Likelihood Ratio  
 Standard 95% Confidence Wald  
 Parameter DF Estimate Error Limits Chi-Square  
  
 Intercept 1 1.6477 0.3547 0.9722 2.3667 21.58  
 total\_bill 1 -0.0725 0.0168 -0.1069 -0.0408 18.65  
 Scale 0 1.0000 0.0000 1.0000 1.0000   
  
 Analysis Of Maximum  
 Likelihood Parameter  
 Estimates  
   
 Parameter Pr > ChiSq  
  
 Intercept <.0001  
 total\_bill <.0001  
 Scale   
  
NOTE: The scale parameter was held fixed.

[if-then/else](#if) statement, [GENMOD Procedure](#genmod)

## 5.4 Fit a linear regression model.

/\* Fit a linear regression model of tip by total\_bill \*/  
proc reg data = tips outest=RegOut;  
 tip\_hat: model tip = total\_bill;  
quit;

The REG Procedure  
 Model: tip\_hat  
 Dependent Variable: tip   
  
 Number of Observations Read 244  
 Number of Observations Used 244  
  
 Analysis of Variance  
   
 Sum of Mean  
 Source DF Squares Square F Value Pr > F  
  
 Model 1 212.42373 212.42373 203.36 <.0001  
 Error 242 252.78874 1.04458   
 Corrected Total 243 465.21248   
  
 Root MSE 1.02205 R-Square 0.4566  
 Dependent Mean 2.99828 Adj R-Sq 0.4544  
 Coeff Var 34.08782   
  
 Parameter Estimates  
   
 Parameter Standard  
 Variable DF Estimate Error t Value Pr > |t|  
  
 Intercept 1 0.92027 0.15973 5.76 <.0001  
 total\_bill 1 0.10502 0.00736 14.26 <.0001

[REG Procedure](#reg)

# 6 Regression & Machine Learning: Modeling & Prediction

## 6.1 Fit a logistic regression model on training data and assess against testing data.

### a) Fit a logistic regression model on training data.

/\* Notice we are using new data sets that need to be read into the   
 environment \*/  
proc import out = train  
 datafile = 'C:/Users/tips\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/tips\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
  
/\* The following code is used to determine if the individual left more than  
 a 15% tip \*/  
data train;   
 set train;   
 if (tip > 0.15\*total\_bill) then greater15 = 1;   
 else greater15 = 0;   
run;   
data test;   
 set test;   
 if (tip > 0.15\*total\_bill) then greater15 = 1;   
 else greater15 = 0;   
run;   
  
/\* The descending option tells SAS to model the probability that   
 greater15 = 1 \*/  
proc genmod data=train descending;   
 model greater15 = total\_bill / dist = bin link = logit lrci;   
 store out = logmod;  
run;

The GENMOD Procedure  
  
 Model Information  
  
 Data Set WORK.TRAIN  
 Distribution Binomial  
 Link Function Logit  
 Dependent Variable greater15  
  
 Number of Observations Read 195  
 Number of Observations Used 195  
 Number of Events 109  
 Number of Trials 195  
  
 Response Profile  
   
 Ordered Total  
 Value greater15 Frequency  
  
 1 1 109  
 2 0 86  
  
PROC GENMOD is modeling the probability that greater15='1'.  
  
 Criteria For Assessing Goodness Of Fit  
   
 Criterion DF Value Value/DF  
  
 Log Likelihood -125.2918   
 Full Log Likelihood -125.2918   
 AIC (smaller is better) 254.5836   
 AICC (smaller is better) 254.6461   
 BIC (smaller is better) 261.1296   
  
  
Algorithm converged.   
  
 Analysis Of Maximum Likelihood Parameter Estimates  
   
 Likelihood Ratio  
 Standard 95% Confidence Wald  
 Parameter DF Estimate Error Limits Chi-Square  
  
 Intercept 1 1.6461 0.3946 0.8973 2.4501 17.40  
 total\_bill 1 -0.0706 0.0185 -0.1088 -0.0359 14.59  
 Scale 0 1.0000 0.0000 1.0000 1.0000   
  
 Analysis Of Maximum  
 Likelihood Parameter  
 Estimates  
   
 Parameter Pr > ChiSq  
  
 Intercept <.0001  
 total\_bill 0.0001  
 Scale   
  
NOTE: The scale parameter was held fixed.

### b) Assess the model against the testing data.

/\* Prediction on testing data \*/  
proc plm source = logmod noprint;  
 score data = test out = preds pred = pred / ilink;  
run;  
  
/\* Determine how many were correctly classified \*/  
data preds;  
 set preds;  
 if (pred < 0.5) then label = 0;  
 else label = 1;  
 if (label = greater15) then Result = "Correct";  
 else Result = "Wrong";  
run;  
  
proc freq data = preds;  
tables Result;  
run;

The FREQ Procedure  
  
 Cumulative Cumulative  
 Result Frequency Percent Frequency Percent  
 ------------------------------------------------------------  
 Correct 34 69.39 34 69.39   
 Wrong 15 30.61 49 100.00

## 6.2 Fit a linear regression model on training data and assess against testing data.

### a) Fit a linear regression model on training data.

/\* Notice we are using new data sets that need to be read into the   
 environment \*/  
proc import out = train  
 datafile = 'C:/Users/boston\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/boston\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
  
proc reg data = train outest=RegOut;  
 predY: model Target = \_0-\_12;  
quit;

The REG Procedure  
 Model: predY  
 Dependent Variable: Target   
  
 Number of Observations Read 354  
 Number of Observations Used 354  
  
 Analysis of Variance  
   
 Sum of Mean  
 Source DF Squares Square F Value Pr > F  
  
 Model 13 22145 1703.47137 68.48 <.0001  
 Error 340 8458.20364 24.87707   
 Corrected Total 353 30603   
  
 Root MSE 4.98769 R-Square 0.7236  
 Dependent Mean 22.48249 Adj R-Sq 0.7131  
 Coeff Var 22.18479   
  
 Parameter Estimates  
   
 Parameter Standard  
 Variable DF Estimate Error t Value Pr > |t|  
  
 Intercept 1 36.10820 6.50497 5.55 <.0001  
 \_0 1 -0.08563 0.04277 -2.00 0.0461  
 \_1 1 0.04603 0.01715 2.68 0.0076  
 \_2 1 0.03641 0.07601 0.48 0.6322  
 \_3 1 3.24796 1.07414 3.02 0.0027  
 \_4 1 -14.87294 4.63609 -3.21 0.0015  
 \_5 1 3.57687 0.53699 6.66 <.0001  
 \_6 1 -0.00870 0.01685 -0.52 0.6059  
 \_7 1 -1.36890 0.25296 -5.41 <.0001  
 \_8 1 0.31312 0.08237 3.80 0.0002  
 \_9 1 -0.01288 0.00460 -2.80 0.0054  
 \_10 1 -0.97690 0.17100 -5.71 <.0001  
 \_11 1 0.01133 0.00336 3.37 0.0008  
 \_12 1 -0.52672 0.06256 -8.42 <.0001

### b) Assess the model against the testing data.

/\* Predicton on testing data \*/  
proc score data = test score=RegOut type=parms predict out = Pred;  
 var \_0-\_12;  
run;  
  
/\* Compute the squared differences between predicted and target \*/  
data Pred;  
 set Pred;  
 sq\_error = (predY - Target)\*\*2;  
run;  
   
/\* Compute the mean of the squared differences (mean squared error) as an   
 assessment of the model \*/  
proc means data = Pred mean;  
 var sq\_error;  
run;

The MEANS Procedure  
  
 Analysis Variable : sq\_error   
   
 Mean  
 ------------  
 17.7713080  
 ------------

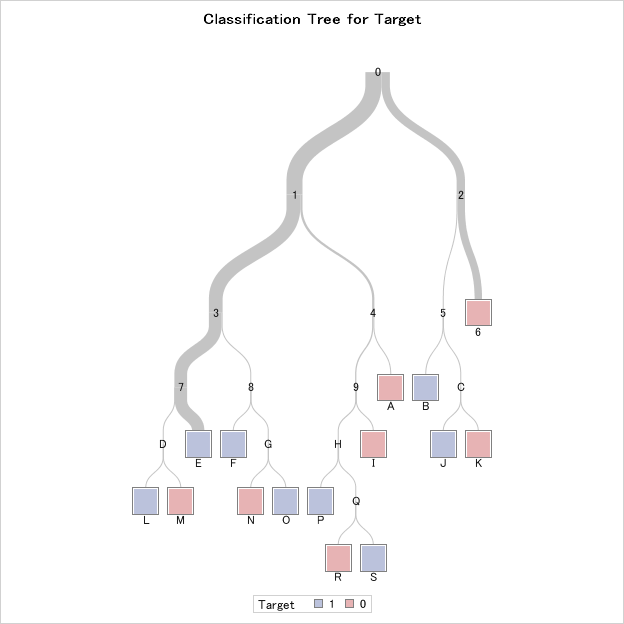
## 6.3 Fit a decision tree model on training data and assess against testing data.

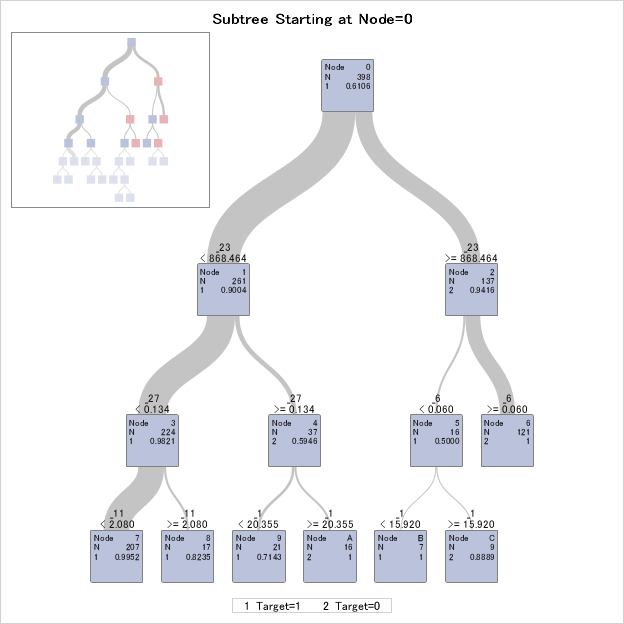
### a) Fit a decision tree classification model.

#### i) Fit a decision tree classification model on training data, plot the tree, and determine variable importance

/\* Notice we are using new data sets that need to be read into the   
 environment \*/  
proc import out = train  
 datafile = 'C:/Users/breastcancer\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/breastcancer\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
/\* HPSPLIT procedure is used to fit a decision tree model \*/  
proc hpsplit data = train;  
 target Target;  
 input \_0-\_29;  
 /\* Export information about variable importance \*/  
 output importance=import;  
 /\* Export the model code so this can be used to score testing data \*/  
 code file='hpbreastcancer.sas';  
run;  
   
/\* Output of this model gives assessment against training data, a plot of   
 the tree, and variable importance \*/

The HPSPLIT Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TRAIN V9 Input On Client  
  
 Model Information  
  
 Split Criterion Used Entropy  
 Pruning Method Cost-Complexity  
 Subtree Evaluation Criterion Cost-Complexity  
 Number of Branches 2  
 Maximum Tree Depth Requested 10  
 Maximum Tree Depth Achieved 6  
 Tree Depth 4  
 Number of Leaves Before Pruning 15  
 Number of Leaves After Pruning 7  
 Model Event Level 1  
  
 Number of Observations Read 398  
 Number of Observations Used 398  
   
   
   
 The HPSPLIT Procedure  
  
 Model-Based Confusion Matrix  
   
 Predicted Error  
 Actual 1 0 Rate  
  
 1 242 1 0.0041  
 0 7 148 0.0452  
  
 Model-Based Fit Statistics for Selected Tree  
   
 N Mis-  
 Leaves ASE class Sensitivity Specificity Entropy Gini RSS  
  
 7 0.0184 0.0201 0.9959 0.9548 0.1135 0.0368 14.6349  
  
 Model-Based Fit Statistics for Selected Tree  
   
 AUC  
  
 0.9858  
  
 Variable Importance  
   
 Training  
 Variable Relative Importance Count  
  
 \_23 1.0000 11.2865 1  
 \_27 0.4072 4.5962 1  
 \_1 0.3487 3.9356 2  
 \_6 0.2355 2.6581 1  
 \_8 0.1674 1.8898 1

Output: 

Output: 

[HPSPLIT Procedure](#hpsplit)

#### ii. Assess the model against the testing data.

/\* Score the test data using the model code \*/  
data scored;  
 set test;  
 %include 'hpbreastcancer.sas';  
run;   
   
/\* Use prediction probabilities to generate predictions, and compare these to   
 the true responses \*/  
/\* If the prediction probability is less than 0.5, classify this as a 0  
 and otherwise classify as a 1. This isn't the best method -- a better   
 method would be randomly assigning a 0 or 1 when a probability of 0.5   
 occurrs, but this insures that results are consistent \*/  
data scored;  
 set scored;  
 if (P\_Target1 < 0.5) then prediction = 0;  
 else prediction = 1;  
 if (Target = prediction) then Result = "Correct";  
 else Result = "Wrong";  
run;  
  
/\* Determine how many were correctly classified \*/  
proc freq data = scored;  
 tables Result;

The FREQ Procedure  
  
 Cumulative Cumulative  
 Result Frequency Percent Frequency Percent  
 ------------------------------------------------------------  
 Correct 157 91.81 157 91.81   
 Wrong 14 8.19 171 100.00

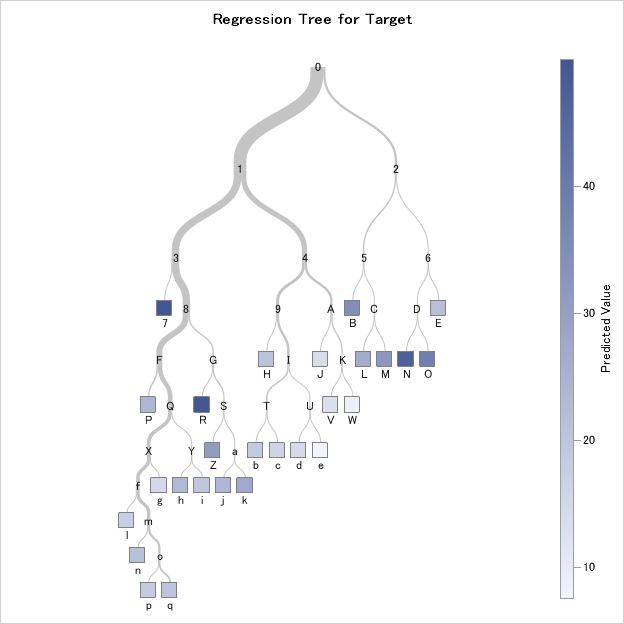
[%include](#include) & [if-then/else](#if) statements, [FREQ Procedure](#freq)

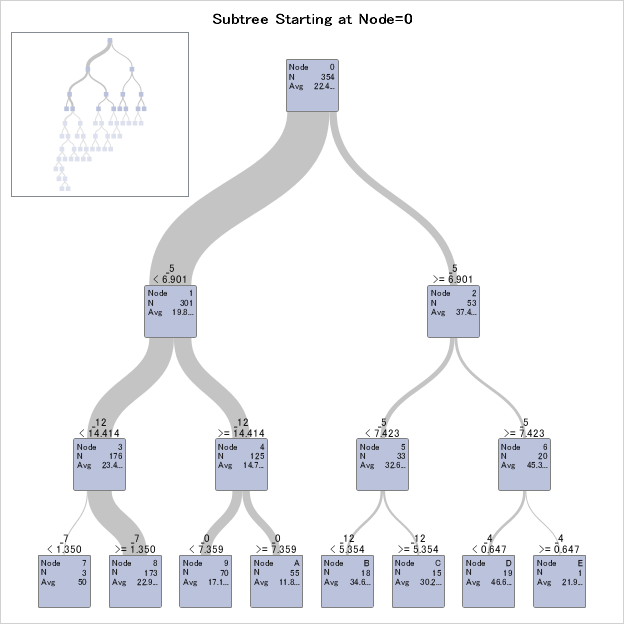
### b) Fit a decision tree regression model.

#### i) Fit a decision tree regression model on training data, plot the tree, and determine variable importance.

/\* Notice we are re-using data sets but it is good to re-read the   
 original versions back into the environment \*/  
proc import out = train  
 datafile = 'C:/Users/boston\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/boston\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
/\* HPSPLIT procedure is used to fit a decision tree model \*/  
proc hpsplit data = train;  
 target Target / level = int;  
 input \_0-\_12;  
 /\* Export information about variable importance \*/  
 output importance=import;  
 /\* Export the model code so this can be used to score testing data \*/  
 code file='hpboston.sas';  
run;  
   
/\* Output of this model gives assessment against training data,   
 a plot of the tree, and variable importance \*/

The HPSPLIT Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TRAIN V9 Input On Client  
  
 Model Information  
  
 Split Criterion Used Variance  
 Pruning Method Cost-Complexity  
 Subtree Evaluation Criterion Cost-Complexity  
 Number of Branches 2  
 Maximum Tree Depth Requested 10  
 Maximum Tree Depth Achieved 10  
 Tree Depth 10  
 Number of Leaves Before Pruning 188  
 Number of Leaves After Pruning 58  
  
 Number of Observations Read 354  
 Number of Observations Used 354  
   
   
   
 The HPSPLIT Procedure  
  
 Model-Based Fit Statistics for Selected Tree  
   
 N  
 Leaves ASE RSS  
  
 58 2.2839 808.5  
  
 Variable Importance  
   
 Training  
 Variable Relative Importance Count  
  
 \_5 1.0000 132.6 8  
 \_12 0.5997 79.5242 8  
 \_7 0.3934 52.1686 5  
 \_4 0.2640 35.0099 9  
 \_0 0.2273 30.1348 3  
 \_9 0.1569 20.8019 7  
 \_10 0.1064 14.1112 4  
 \_6 0.1050 13.9290 5  
 \_11 0.0704 9.3310 4  
 \_8 0.0679 8.9986 2  
 \_2 0.0476 6.3100 2

Output: 

Output: 

/\* Score the test data using the model code \*/  
data scored;  
 set test;  
 %include 'hpboston.sas';  
run;   
   
/\* Compute the squared differences between predicted and target \*/  
data scored;  
 set scored;  
 sq\_error = (P\_Target - Target)\*\*2;  
run;  
   
/\* Compute the mean of the squared differences (mean squared error) as an assessment   
 of the model \*/  
proc means data = scored mean;  
 var sq\_error;  
run;

The MEANS Procedure  
  
 Analysis Variable : sq\_error   
   
 Mean  
 ------------  
 25.5687875  
 ------------

## 6.4 Fit a random forest model on training data and assess against testing data.

### a) Fit a random forest classification model.

#### i) Fit a random forest classification model on training data and determine variable importance.

/\* Notice we are re-using data sets but it is good to re-read the   
 original version back into the environment \*/  
proc import out = train  
 datafile = 'C:/Users/breastcancer\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/breastcancer\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
/\* Output includes information about variable importance \*/  
proc hpforest data = train;  
 input \_0 - \_29 / level = interval;  
 target Target / level = nominal;  
 save file = 'hpbreastcancer2.bin';  
run;

The HPFOREST Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TRAIN V9 Input On Client  
  
 Model Information  
   
 Parameter Value  
  
 Variables to Try 5 (Default)   
 Maximum Trees 100 (Default)   
 Inbag Fraction 0.6 (Default)   
 Prune Fraction 0 (Default)   
 Prune Threshold 0.1 (Default)   
 Leaf Fraction 0.00001 (Default)   
 Leaf Size Setting 1 (Default)   
 Leaf Size Used 1   
 Category Bins 30 (Default)   
 Interval Bins 100   
 Minimum Category Size 5 (Default)   
 Node Size 100000 (Default)   
 Maximum Depth 20 (Default)   
 Alpha 1 (Default)   
 Exhaustive 5000 (Default)   
 Rows of Sequence to Skip 5 (Default)   
 Split Criterion . Gini   
 Preselection Method . BinnedSearch  
 Missing Value Handling . Valid value   
  
 Number of Observations  
   
 Type N  
  
 Number of Observations Read 398  
 Number of Observations Used 398  
  
 Baseline Fit Statistics  
   
 Statistic Value  
  
 Average Square Error 0.238  
 Misclassification Rate 0.389  
 Log Loss 0.669  
  
 Fit Statistics  
   
 Average Average  
 Square Square Misclassification  
 Number Number Error Error Rate  
 of Trees of Leaves (Train) (OOB) (Train)  
  
 1 16 0.03015 0.0750 0.03015  
 2 35 0.01947 0.0739 0.04523  
 3 53 0.01284 0.0724 0.00754  
 4 66 0.01225 0.0658 0.01005  
 5 80 0.01156 0.0700 0.00754  
 6 92 0.01124 0.0712 0.00754  
 7 106 0.00938 0.0633 0.00251  
 8 122 0.00879 0.0623 0.00000  
 9 139 0.00887 0.0611 0.00000  
 10 157 0.00867 0.0611 0.00000  
 11 171 0.00889 0.0589 0.00251  
 12 188 0.00874 0.0557 0.00000  
 13 203 0.00847 0.0551 0.00000  
 14 223 0.00841 0.0552 0.00000  
 15 241 0.00804 0.0537 0.00251  
 16 253 0.00795 0.0496 0.00251  
 17 268 0.00827 0.0489 0.00503  
 18 283 0.00813 0.0485 0.00251  
 19 300 0.00793 0.0471 0.00251  
 20 315 0.00783 0.0471 0.00251  
 21 329 0.00763 0.0465 0.00251  
 22 345 0.00747 0.0453 0.00000  
 23 361 0.00740 0.0448 0.00000  
 24 375 0.00744 0.0442 0.00000  
 25 392 0.00749 0.0449 0.00251  
 26 406 0.00764 0.0448 0.00251  
 27 420 0.00750 0.0440 0.00251  
 28 437 0.00764 0.0438 0.00000  
 29 451 0.00776 0.0431 0.00000  
 30 466 0.00774 0.0426 0.00000  
 31 484 0.00778 0.0432 0.00251  
 32 502 0.00759 0.0426 0.00000  
 33 518 0.00749 0.0420 0.00251  
 34 535 0.00747 0.0418 0.00000  
 35 550 0.00742 0.0415 0.00000  
 36 562 0.00746 0.0411 0.00000  
 37 578 0.00741 0.0411 0.00000  
 38 594 0.00731 0.0404 0.00000  
 39 609 0.00717 0.0407 0.00000  
 40 623 0.00720 0.0404 0.00000  
 41 642 0.00712 0.0405 0.00000  
 42 661 0.00702 0.0399 0.00000  
 43 679 0.00687 0.0397 0.00000  
 44 692 0.00677 0.0396 0.00000  
 45 710 0.00665 0.0392 0.00000  
 46 731 0.00652 0.0391 0.00000  
 47 741 0.00654 0.0387 0.00000  
 48 754 0.00661 0.0392 0.00000  
 49 769 0.00656 0.0393 0.00000  
 50 780 0.00657 0.0395 0.00000  
 51 795 0.00658 0.0395 0.00000  
 52 812 0.00657 0.0399 0.00000  
 53 829 0.00653 0.0399 0.00000  
 54 843 0.00662 0.0402 0.00000  
 55 856 0.00662 0.0403 0.00000  
 56 869 0.00663 0.0401 0.00000  
 57 883 0.00655 0.0396 0.00000  
 58 898 0.00653 0.0397 0.00000  
 59 914 0.00653 0.0394 0.00000  
 60 929 0.00661 0.0397 0.00000  
 61 946 0.00658 0.0396 0.00000  
 62 959 0.00655 0.0393 0.00000  
 63 975 0.00657 0.0394 0.00000  
 64 988 0.00660 0.0393 0.00000  
 65 1008 0.00662 0.0396 0.00000  
 66 1020 0.00671 0.0397 0.00000  
 67 1036 0.00675 0.0401 0.00000  
 68 1054 0.00672 0.0397 0.00000  
 69 1072 0.00678 0.0401 0.00000  
 70 1088 0.00686 0.0405 0.00000  
 71 1103 0.00692 0.0407 0.00000  
 72 1122 0.00692 0.0410 0.00000  
 73 1137 0.00695 0.0411 0.00000  
 74 1156 0.00682 0.0406 0.00000  
 75 1171 0.00678 0.0406 0.00000  
 76 1188 0.00668 0.0403 0.00000  
 77 1202 0.00665 0.0402 0.00000  
 78 1215 0.00661 0.0402 0.00000  
 79 1229 0.00661 0.0400 0.00000  
 80 1247 0.00658 0.0399 0.00000  
 81 1263 0.00657 0.0395 0.00000  
 82 1276 0.00659 0.0394 0.00000  
 83 1292 0.00659 0.0393 0.00000  
 84 1305 0.00652 0.0388 0.00000  
 85 1322 0.00649 0.0387 0.00000  
 86 1342 0.00644 0.0386 0.00000  
 87 1359 0.00647 0.0387 0.00000  
 88 1373 0.00655 0.0388 0.00000  
 89 1389 0.00655 0.0389 0.00000  
 90 1404 0.00652 0.0385 0.00000  
 91 1418 0.00658 0.0386 0.00000  
 92 1432 0.00652 0.0383 0.00000  
 93 1447 0.00649 0.0381 0.00000  
 94 1460 0.00654 0.0382 0.00000  
 95 1481 0.00657 0.0386 0.00000  
 96 1495 0.00650 0.0383 0.00000  
 97 1509 0.00646 0.0381 0.00000  
 98 1522 0.00651 0.0382 0.00000  
 99 1537 0.00649 0.0382 0.00000  
 100 1554 0.00647 0.0382 0.00000  
  
 Fit Statistics  
   
 Misclassification Log Log  
 Rate Loss Loss  
 (OOB) (Train) (OOB)  
  
 0.0750 0.6942 1.727  
 0.0895 0.1558 1.545  
 0.0952 0.0429 1.358  
 0.0893 0.0453 1.059  
 0.0877 0.0447 1.139  
 0.0871 0.0457 1.054  
 0.0803 0.0417 0.860  
 0.0821 0.0414 0.800  
 0.0842 0.0424 0.742  
 0.0787 0.0429 0.743  
 0.0734 0.0445 0.739  
 0.0732 0.0447 0.626  
 0.0732 0.0443 0.574  
 0.0781 0.0447 0.574  
 0.0756 0.0436 0.571  
 0.0729 0.0433 0.457  
 0.0678 0.0439 0.404  
 0.0603 0.0436 0.404  
 0.0628 0.0430 0.349  
 0.0628 0.0429 0.349  
 0.0628 0.0425 0.348  
 0.0628 0.0420 0.294  
 0.0653 0.0418 0.294  
 0.0628 0.0416 0.292  
 0.0628 0.0420 0.294  
 0.0628 0.0423 0.243  
 0.0603 0.0418 0.241  
 0.0603 0.0429 0.241  
 0.0578 0.0433 0.239  
 0.0578 0.0436 0.239  
 0.0628 0.0437 0.241  
 0.0578 0.0435 0.240  
 0.0553 0.0430 0.238  
 0.0553 0.0431 0.237  
 0.0553 0.0432 0.237  
 0.0528 0.0430 0.236  
 0.0528 0.0431 0.236  
 0.0528 0.0428 0.185  
 0.0553 0.0427 0.186  
 0.0528 0.0426 0.185  
 0.0553 0.0424 0.186  
 0.0553 0.0422 0.184  
 0.0553 0.0418 0.184  
 0.0553 0.0415 0.184  
 0.0578 0.0410 0.183  
 0.0578 0.0410 0.183  
 0.0528 0.0411 0.182  
 0.0578 0.0412 0.182  
 0.0553 0.0412 0.183  
 0.0553 0.0415 0.183  
 0.0528 0.0414 0.183  
 0.0578 0.0417 0.184  
 0.0578 0.0415 0.184  
 0.0578 0.0420 0.186  
 0.0578 0.0420 0.186  
 0.0528 0.0421 0.186  
 0.0528 0.0418 0.185  
 0.0528 0.0418 0.185  
 0.0528 0.0417 0.184  
 0.0553 0.0418 0.184  
 0.0528 0.0417 0.184  
 0.0553 0.0415 0.184  
 0.0578 0.0416 0.184  
 0.0578 0.0416 0.184  
 0.0578 0.0418 0.184  
 0.0578 0.0421 0.185  
 0.0603 0.0422 0.186  
 0.0578 0.0421 0.185  
 0.0553 0.0425 0.186  
 0.0578 0.0428 0.187  
 0.0578 0.0430 0.188  
 0.0578 0.0432 0.189  
 0.0603 0.0431 0.189  
 0.0603 0.0427 0.188  
 0.0578 0.0425 0.188  
 0.0553 0.0423 0.187  
 0.0578 0.0423 0.187  
 0.0578 0.0422 0.187  
 0.0578 0.0421 0.187  
 0.0553 0.0421 0.186  
 0.0578 0.0420 0.185  
 0.0553 0.0420 0.185  
 0.0553 0.0419 0.184  
 0.0553 0.0417 0.183  
 0.0528 0.0416 0.183  
 0.0553 0.0414 0.183  
 0.0528 0.0415 0.183  
 0.0528 0.0416 0.184  
 0.0503 0.0417 0.184  
 0.0477 0.0416 0.183  
 0.0503 0.0417 0.183  
 0.0503 0.0415 0.183  
 0.0528 0.0414 0.134  
 0.0503 0.0417 0.134  
 0.0528 0.0419 0.135  
 0.0503 0.0416 0.135  
 0.0477 0.0415 0.134  
 0.0477 0.0416 0.134  
 0.0477 0.0415 0.134  
 0.0452 0.0416 0.135  
  
 Loss Reduction Variable Importance  
   
 Number OOB OOB  
 Variable of Rules Gini Gini Margin Margin  
  
 \_7 69 0.057751 0.05100 0.115502 0.10851  
 \_27 116 0.057536 0.04812 0.115072 0.10648  
 \_22 66 0.053462 0.04054 0.106925 0.09267  
 \_23 92 0.049798 0.03969 0.099596 0.08961  
 \_20 84 0.045727 0.03686 0.091453 0.08190  
 \_2 43 0.030053 0.02561 0.060105 0.05721  
 \_0 44 0.026259 0.01873 0.052518 0.04483  
 \_13 47 0.018831 0.01425 0.037662 0.03329  
 \_6 55 0.021984 0.01321 0.043968 0.03523  
 \_3 16 0.010751 0.01275 0.021502 0.02310  
 \_26 84 0.017139 0.00693 0.034279 0.02387  
 \_21 73 0.009979 0.00400 0.019958 0.01367  
 \_10 31 0.007944 0.00273 0.015889 0.01089  
 \_12 31 0.007102 0.00217 0.014204 0.00929  
 \_17 31 0.002941 0.00049 0.005882 0.00286  
 \_5 12 0.001882 -0.00010 0.003764 0.00152  
 \_16 17 0.001134 -0.00055 0.002268 0.00089  
 \_11 23 0.001679 -0.00057 0.003358 0.00096  
 \_8 22 0.001543 -0.00077 0.003086 0.00052  
 \_18 22 0.001787 -0.00105 0.003573 0.00081  
 \_9 23 0.001656 -0.00105 0.003312 0.00063  
 \_4 22 0.002237 -0.00114 0.004475 0.00147  
 \_1 58 0.008366 -0.00147 0.016732 0.00648  
 \_24 80 0.010527 -0.00149 0.021054 0.00906  
 \_25 55 0.005040 -0.00151 0.010081 0.00449  
 \_28 70 0.008423 -0.00168 0.016846 0.00617  
 \_15 16 0.001345 -0.00203 0.002690 -0.00059  
 \_14 29 0.001679 -0.00282 0.003357 -0.00110  
 \_19 49 0.003804 -0.00413 0.007609 -0.00028  
 \_29 74 0.005801 -0.00418 0.011603 0.00225

#### ii) Assess the model against the testing data.

/\* Prediction on testing data \*/  
proc hp4score data = test seed = 29;  
 score file = 'hpbreastcancer2.bin' out = scored;  
run;  
  
/\* Determine how many were correctly classified \*/  
data scored;  
 set scored;  
 if (I\_Target = Target) then Result = "Correct";  
 else Result = "Wrong";  
run;  
  
proc freq data = scored;  
 tables Result;  
run;   
NA

The HP4SCORE Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TEST V9 Input On Client  
 WORK.SCORED V9 Output On Client  
  
 Number of Observations  
   
 Type N  
  
 Number of Observations Read 171  
 Number of Observations Used 171  
 Sum of Frequencies Used 171  
   
   
   
 The FREQ Procedure  
  
 Cumulative Cumulative  
 Result Frequency Percent Frequency Percent  
 ------------------------------------------------------------  
 Correct 166 97.08 166 97.08   
 Wrong 5 2.92 171 100.00

### b) Fit a random forest regression model.

#### i) Fit a random forest regression model on training data and determine variable importance.

/\* Notice we are re-using data sets but it is good to re-read the original   
 versions back into the environment \*/  
proc import out = train  
 datafile = 'C:/Users/boston\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/boston\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
proc hpforest data = train;  
 input \_0-\_12 / level = interval;  
 target Target / level = interval;  
 save file = 'hpboston2.bin';  
run;

The HPFOREST Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TRAIN V9 Input On Client  
  
 Model Information  
   
 Parameter Value  
  
 Variables to Try 4 (Default)   
 Maximum Trees 100 (Default)   
 Inbag Fraction 0.6 (Default)   
 Prune Fraction 0 (Default)   
 Prune Threshold 0.1 (Default)   
 Leaf Fraction 0.00001 (Default)   
 Leaf Size Setting 1 (Default)   
 Leaf Size Used 1   
 Category Bins 30 (Default)   
 Interval Bins 100   
 Minimum Category Size 5 (Default)   
 Node Size 100000 (Default)   
 Maximum Depth 20 (Default)   
 Alpha 1 (Default)   
 Exhaustive 5000 (Default)   
 Rows of Sequence to Skip 5 (Default)   
 Split Criterion . Variance   
 Preselection Method . BinnedSearch  
 Missing Value Handling . Valid value   
  
 Number of Observations  
   
 Type N  
  
 Number of Observations Read 354  
 Number of Observations Used 354  
  
 Baseline Fit Statistics  
   
 Statistic Value  
  
 Average Square Error 86.450  
  
 Fit Statistics  
   
 Average Average  
 Square Square  
 Number Number Error Error  
 of Trees of Leaves (Train) (OOB)  
  
 1 187 19.2696 47.7098  
 2 375 11.0807 42.1586  
 3 576 6.4927 30.3271  
 4 771 4.7796 24.0581  
 5 959 4.5159 23.4567  
 6 1155 4.9110 22.6319  
 7 1347 4.1583 23.0376  
 8 1547 3.7435 21.2464  
 9 1748 3.4531 21.2850  
 10 1946 3.1073 20.2032  
 11 2136 3.2121 19.2041  
 12 2332 3.1237 18.8289  
 13 2525 3.3520 18.5622  
 14 2725 3.3682 18.2174  
 15 2923 3.2042 18.1507  
 16 3116 3.0559 17.8623  
 17 3313 3.0496 17.9996  
 18 3507 2.9018 17.6530  
 19 3701 2.7845 17.7495  
 20 3901 2.9128 17.7119  
 21 4094 2.9422 17.0785  
 22 4285 2.8756 16.2444  
 23 4480 2.8453 16.4826  
 24 4672 2.8007 16.3981  
 25 4866 2.8716 16.3951  
 26 5064 2.7868 15.7487  
 27 5256 2.6759 15.3203  
 28 5447 2.6251 14.9469  
 29 5638 2.6434 15.0160  
 30 5827 2.5882 14.7211  
 31 6019 2.5907 14.9239  
 32 6213 2.5328 14.9865  
 33 6406 2.4965 14.8766  
 34 6601 2.4116 14.6827  
 35 6794 2.3845 14.7050  
 36 6990 2.3678 14.5682  
 37 7180 2.4587 14.8108  
 38 7377 2.4158 14.7808  
 39 7568 2.3817 14.7619  
 40 7755 2.3894 14.7697  
 41 7939 2.3840 14.8463  
 42 8135 2.3803 14.9734  
 43 8328 2.3553 14.6860  
 44 8526 2.3624 14.7508  
 45 8727 2.2821 14.3780  
 46 8916 2.3876 14.6701  
 47 9107 2.3905 14.6762  
 48 9305 2.3957 14.7101  
 49 9497 2.4048 14.5986  
 50 9696 2.3662 14.5174  
 51 9895 2.3188 14.4484  
 52 10090 2.3388 14.5882  
 53 10290 2.3816 14.7050  
 54 10495 2.3743 14.6290  
 55 10695 2.3830 14.6148  
 56 10887 2.3311 14.3839  
 57 11078 2.3451 14.2691  
 58 11265 2.3477 14.3125  
 59 11467 2.3386 14.3373  
 60 11657 2.3173 14.3444  
 61 11846 2.3003 14.2456  
 62 12045 2.2826 14.2803  
 63 12234 2.2707 14.3694  
 64 12429 2.2625 14.2487  
 65 12618 2.2413 14.2810  
 66 12812 2.2777 14.3964  
 67 13006 2.3028 14.4682  
 68 13201 2.3037 14.4159  
 69 13394 2.2756 14.3095  
 70 13588 2.2794 14.2381  
 71 13781 2.2749 14.2143  
 72 13978 2.2731 14.2176  
 73 14177 2.2565 14.1390  
 74 14366 2.2477 14.0939  
 75 14562 2.2606 14.0682  
 76 14759 2.2488 14.0231  
 77 14958 2.2422 13.8466  
 78 15147 2.2732 13.8690  
 79 15339 2.2576 13.9034  
 80 15537 2.2376 13.8661  
 81 15733 2.2299 13.8930  
 82 15927 2.2177 13.8899  
 83 16123 2.2365 13.9875  
 84 16311 2.2395 14.0420  
 85 16505 2.2386 14.0029  
 86 16697 2.2201 13.9355  
 87 16891 2.2257 13.9783  
 88 17088 2.2116 13.9920  
 89 17276 2.1882 13.8726  
 90 17466 2.1849 13.8545  
 91 17661 2.1733 13.8756  
 92 17851 2.2094 13.9935  
 93 18049 2.1899 13.9606  
 94 18244 2.1923 13.8830  
 95 18446 2.2126 13.7955  
 96 18641 2.2280 13.8094  
 97 18843 2.2039 13.5802  
 98 19031 2.2005 13.5466  
 99 19227 2.2097 13.5654  
 100 19422 2.2082 13.4970  
  
 Loss Reduction Variable Importance  
   
 Number OOB Absolute OOB Absolute  
 Variable of Rules MSE MSE Error Error  
  
 \_5 1642 24.64748 22.39019 1.626730 1.268681  
 \_12 4406 26.71010 21.49366 1.776141 1.033850  
 \_2 905 7.90343 5.42904 0.532152 0.272568  
 \_10 978 3.94471 2.33507 0.298425 0.109679  
 \_4 1104 5.14102 2.21447 0.456087 0.217719  
 \_9 1283 2.46981 1.33514 0.266256 0.095041  
 \_0 405 2.12306 0.34192 0.167649 0.065257  
 \_1 137 0.06909 -0.01591 0.016111 -0.003479  
 \_7 2375 6.58607 -0.20475 0.606985 0.096708  
 \_3 183 0.50025 -0.23840 0.037301 -0.015978  
 \_8 781 0.58187 -0.33349 0.092398 -0.026270  
 \_11 3567 2.79869 -0.98748 0.473352 -0.018384  
 \_6 1556 1.72934 -1.05805 0.265803 -0.027071

#### ii) Assess the model against the testing data.

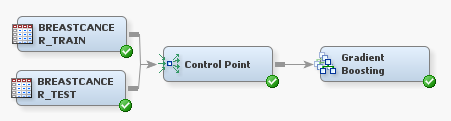
/\* Prediction on testing data \*/  
proc hp4score data = test seed = 29;  
 score file = 'hpboston2.bin' out = scored;  
run;  
   
/\* Compute the squared differences between predicted and target \*/  
data scored;  
 set scored;  
 sq\_error = (P\_Target - Target)\*\*2;  
run;  
   
/\* Compute the mean of the squared differences (mean squared error) as an   
 assessment of the model \*/  
proc means data = scored mean;  
 var sq\_error;  
run;

The HP4SCORE Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TEST V9 Input On Client  
 WORK.SCORED V9 Output On Client  
  
 Number of Observations  
   
 Type N  
  
 Number of Observations Read 152  
 Number of Observations Used 152  
 Sum of Frequencies Used 152  
   
   
   
 The MEANS Procedure  
  
 Analysis Variable : sq\_error   
   
 Mean  
 ------------  
 9.0696172  
 ------------

## 6.5 Fit a gradient boosting model on training data and assess against testing data.

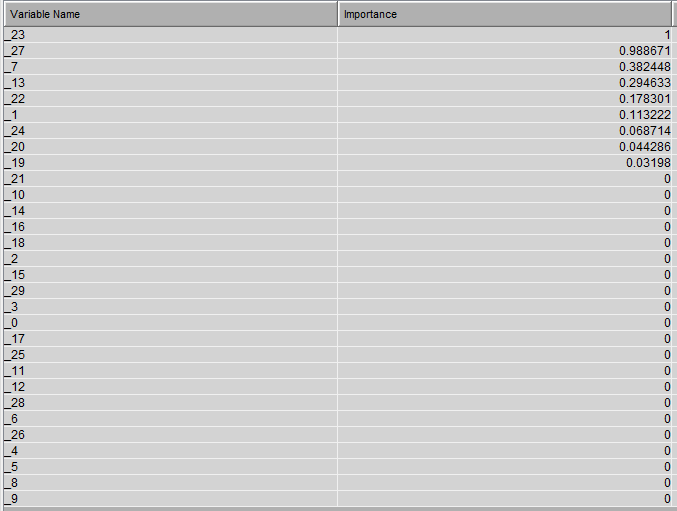
### a) Fit a gradient boosting classification model.

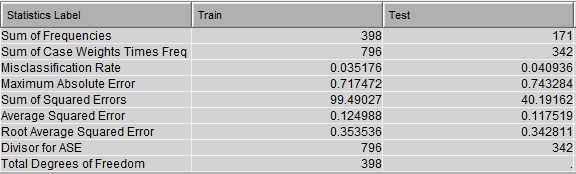
Currently, there is not a gradient boosting procedure available in Base SAS Therefore, the best method to create a gradient boosting model currently is using SAS Enterprise Miner. Create the following diagram in SAS Enterprise Miner:

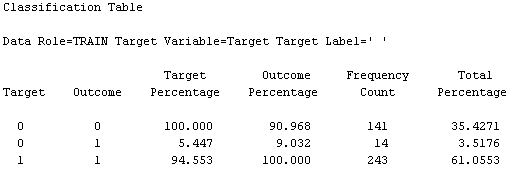
Output: 

For the Gradient Boosting node, set Seed = 29, set Shrinkage = 0.01, and set Train Proportion = 100.

This diagram results in the following variable importance and misclassification against training & testing data:

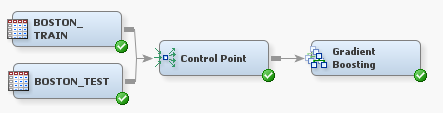
Output: 

Output: 

Output: 

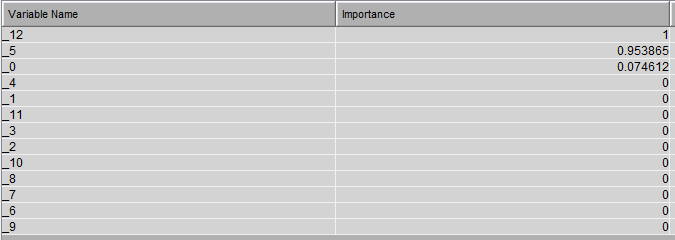
### b) Fit a gradient boosting regression model.

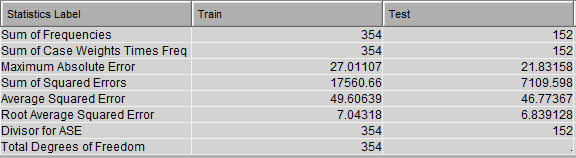
Again, there is not a gradient boosting procedure available in Base SAS, currently. Create the following diagram in SAS Enterprise Miner:

Output: 

For the Gradient Boosting node, set Seed = 29, set Shrinkage = 0.01, and set Train Proportion = 100.

This diagram results in the following variable importance and root mean squared error against training & testing data:

Output: 

Output: 

## 6.6 Fit an extreme gradient boosting model on taining data and assess against testing data.

### a) Fit an extreme gradient boosting classification model.

#### Fit an extreme gradient boosting classification model on training data and assess the model against the testing data.

proc iml;  
 submit / R;  
 train = read.csv('C:/Users/breastcancer\_train.csv')  
 test = read.csv('C:/Users/breastcancer\_test.csv')  
   
 library(xgboost)  
 set.seed(29)  
   
 xgbMod <- xgboost(data.matrix(subset(train, select = -c(Target))),   
 data.matrix(train$Target), max\_depth = 3, nrounds = 2,   
 objective = "binary:logistic", n\_estimators = 2500,  
 shrinkage = .01)  
 # Prediction on testing data  
 predictions <- predict(xgbMod, data.matrix(subset(test, select = - c(Target))))  
 pred.response <- ifelse(predictions < 0.5, 0, 1)  
   
 # Determine how many were correctly classified  
 Results <- ifelse(test$Target == pred.response, "Correct", "Wrong")   
 table(Results)  
 endsubmit;  
quit;

[1] train-error:0.037688  
[2] train-error:0.020101  
Results  
Correct Wrong  
 165 6

#### Fit an extreme gradient boosting regression model on training data and assess the model against the testing data.

proc iml;  
 submit / R;  
 train = read.csv('C:/Users/boston\_train.csv')  
 test = read.csv('C:/Users/boston\_test.csv')  
   
 library(xgboost)  
 set.seed(29)  
   
 xgbMod <- xgboost(data.matrix(subset(train, select = -c(Target))),   
 data.matrix(train$Target / 50), max\_depth = 3,   
 nrounds = 2, n\_estimators = 2500, shrinkage = .01)  
   
 # Predict the target in the testing data, remembering to  
 # multiply by 50  
 prediction = data.frame(matrix(ncol = 0, nrow = nrow(test)))   
 prediction$target\_hat <- predict(xgbMod,   
 data.matrix(subset(test,   
 select = - c(Target))))\*50  
   
 # Compute the squared difference between predicted tip and actual tip  
 prediction$sq\_diff <- (prediction$target\_hat - test$Target)\*\*2  
   
 # Compute the mean of the squared differences (mean squared error) # as an assessment of the model  
 mean\_sq\_error <- mean(prediction$sq\_diff)  
 print(mean\_sq\_error)  
 endsubmit;  
quit;

[1] train-rmse:0.146609  
[2] train-rmse:0.114851  
[1] 36.13079

## 6.7 Fit a support vector model on training data and assess against testing data.

### a) Fit a support vector classification model.

#### i) Fit a support vector classification model on training data.

/\* Notice we are re-using data sets but it is good to re-read the original   
 versions back into the environment \*/  
proc import out = train  
 datafile = 'C:/Users/breastcancer\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/breastcancer\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
/\* Fit a support vector classification model \*/  
proc hpsvm data = train;  
 input \_0-\_29 / level = interval;  
 target Target / level = nominal;  
 code file='hpbreastcancer3.sas';  
run;

The HPSVM Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TRAIN V9 Input On Client  
  
 Model Information  
  
 Task Type C\_CLAS   
 Optimization Technique Interior Point  
 Scale YES   
 Kernel Function Linear   
 Penalty Method C   
 Penalty Parameter 1   
 Maximum Iterations 25   
 Tolerance 1e-06   
  
 Number of Observations Read 398  
 Number of Observations Used 398  
  
 Training Results  
  
 Inner Product of Weights 35.2508001  
 Bias -6.375275  
 Total Slack (Constraint Violations) 34.3511008  
 Norm of Longest Vector 3.79226578  
 Number of Support Vectors 71  
 Number of Support Vectors on Margin 63  
 Maximum F 11.4630802  
 Minimum F -4.7061491  
 Number of Effects 30  
 Columns in Data Matrix 30  
  
 Iteration History  
   
 Iteration Complementarity Feasibility  
  
 1 1002265.3132 88067.240896  
 2 1411.2168312 80.210592636  
 3 210.36307705 8.0210592E-7  
 4 5.5675772656 1.2652961E-8  
 5 0.8865572275 1.544403E-10  
 6 0.2947605635 3.866263E-11  
 7 0.1606295757 1.766043E-11  
 8 0.0981078445 8.719581E-12  
 9 0.0603316585 4.770961E-12  
 10 0.0258720492 1.4998E-12  
 11 0.0171466879 5.151435E-13  
 12 0.0090859249 1.514344E-13  
 13 0.0023785349 3.508305E-14  
 14 0.0001072635 3.552714E-15  
 15 4.813479E-7 5.617035E-15  
  
 Classification Matrix  
   
 Training Prediction  
 Observed 1 0 Total  
  
 1 243 0 243  
 0 7 148 155  
 Total 250 148 398  
  
 Fit Statistics  
   
 Statistic Training  
  
 Accuracy 0.9824  
 Error 0.0176  
 Sensitivity 1.0000  
 Specificity 0.9548

#### ii) Assess the model against the testing data.

/\* Prediction on testing data \*/  
data scored;  
 set test;  
 %include 'hpbreastcancer3.sas';  
run;  
   
/\* Determine how many were correctly classified \*/  
data scored;  
 set scored;  
 if (I\_Target = Target) then Result = "Correct";  
 else Result = "Wrong";  
run;  
  
proc freq data = scored;  
 tables Result;  
run;

The FREQ Procedure  
  
 Cumulative Cumulative  
 Result Frequency Percent Frequency Percent  
 ------------------------------------------------------------  
 Correct 167 97.66 167 97.66   
 Wrong 4 2.34 171 100.00

[HPSVM Procedure](#svm), [%include](#include) & [if-then/else](#if) statements, [FREQ Procedure](#freq)

### b) Fit a support vector regression model.

Not available in this current release.

## 6.8 Fit a neural network model on training data and assess against testing data.

### a) Fit a neural network classification model.

#### i) Fit a neural network classification model on training data.

/\* Notice we are using new data sets that need to be read into the   
 environment \*/  
proc import out = train  
 datafile = 'C:/Users/digits\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/digits\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
  
/\* In order to use the NEURAL Procedure we first need to create a data  
 mining database (DMDB) that reflects the original data \*/  
proc dmdb batch data = train  
 out = dmtrain  
 dmdbcat = digits;  
 var \_0 - \_63;  
 class Target;  
 target Target;  
run;  
proc dmdb batch data = test  
 out = dmtest  
 dmdbcat = digits;  
 var \_0 - \_63;  
 class Target;  
 target Target;  
run;  
  
/\* Now we can fit the neural network model \*/  
/\* Neural network produces a lot of output which is why here  
 "nloptions noprint" is specified \*/  
proc neural data = train dmdbcat = digits random = 29;  
 nloptions noprint;  
 input \_0 - \_63 / level = interval;  
 target Target / level = nominal;  
 archi MLP hidden=100;  
 train maxiter = 200;  
 score out = out outfit = fit;  
 score data = test out = gridout;  
run;

#### ii) Assess the model against the testing data.

run;   
/\* Prediction on testing data \*/  
data scored;  
 set gridout;  
 rename I\_Target = Prediction;  
run;  
   
/\* This produces a confusion matrix \*/  
proc freq data = scored;  
 tables Target\*Prediction / nopercent norow nocol;

The FREQ Procedure  
  
 Table of Target by Prediction  
  
 Target Prediction(Into: Target)  
  
 Frequency|0 |1 |2 |3 |4 | Total  
 ---------+--------+--------+--------+--------+--------+  
 0 | 58 | 0 | 0 | 0 | 0 | 58  
 ---------+--------+--------+--------+--------+--------+  
 1 | 1 | 56 | 0 | 0 | 0 | 58  
 ---------+--------+--------+--------+--------+--------+  
 2 | 0 | 0 | 58 | 0 | 0 | 58  
 ---------+--------+--------+--------+--------+--------+  
 3 | 0 | 0 | 0 | 58 | 0 | 59  
 ---------+--------+--------+--------+--------+--------+  
 4 | 0 | 0 | 0 | 0 | 51 | 54  
 ---------+--------+--------+--------+--------+--------+  
 5 | 0 | 0 | 0 | 0 | 0 | 59  
 ---------+--------+--------+--------+--------+--------+  
 6 | 0 | 0 | 0 | 0 | 0 | 41  
 ---------+--------+--------+--------+--------+--------+  
 7 | 0 | 0 | 0 | 0 | 0 | 51  
 ---------+--------+--------+--------+--------+--------+  
 8 | 0 | 4 | 0 | 0 | 0 | 45  
 ---------+--------+--------+--------+--------+--------+  
 9 | 0 | 0 | 0 | 0 | 0 | 57  
 ---------+--------+--------+--------+--------+--------+  
 Total 59 60 58 58 51 540  
 (Continued)  
  
 Table of Target by Prediction  
  
 Target Prediction(Into: Target)  
  
 Frequency|5 |6 |7 |8 |9 | Total  
 ---------+--------+--------+--------+--------+--------+  
 0 | 0 | 0 | 0 | 0 | 0 | 58  
 ---------+--------+--------+--------+--------+--------+  
 1 | 0 | 1 | 0 | 0 | 0 | 58  
 ---------+--------+--------+--------+--------+--------+  
 2 | 0 | 0 | 0 | 0 | 0 | 58  
 ---------+--------+--------+--------+--------+--------+  
 3 | 1 | 0 | 0 | 0 | 0 | 59  
 ---------+--------+--------+--------+--------+--------+  
 4 | 1 | 1 | 0 | 1 | 0 | 54  
 ---------+--------+--------+--------+--------+--------+  
 5 | 58 | 0 | 0 | 0 | 1 | 59  
 ---------+--------+--------+--------+--------+--------+  
 6 | 0 | 41 | 0 | 0 | 0 | 41  
 ---------+--------+--------+--------+--------+--------+  
 7 | 1 | 0 | 50 | 0 | 0 | 51  
 ---------+--------+--------+--------+--------+--------+  
 8 | 0 | 0 | 0 | 39 | 2 | 45  
 ---------+--------+--------+--------+--------+--------+  
 9 | 2 | 0 | 0 | 2 | 53 | 57  
 ---------+--------+--------+--------+--------+--------+  
 Total 63 43 50 42 56 540

### b) Fit a neural network regression model.

#### i) Fit a neural network regression model on training data.

/\* Notice we are re-using data sets but it is good to re-read the original   
 versions back into the environment \*/  
proc import out = train  
 datafile = 'C:/Users/boston\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/boston\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
  
/\* In order to use the NEURAL Procedure we first need to create a data  
 mining database (DMDB) that reflects the original data \*/  
proc dmdb batch data = train  
 out = dmtrain  
 dmdbcat = boston;  
 var \_0 - \_12 Target;  
 target Target;  
run;  
proc dmdb batch data = test  
 out = dmtest  
 dmdbcat = boston;  
 var \_0 - \_12 Target;  
 target Target;  
run;  
  
/\* Now we can fit the neural network model \*/  
/\* Neural network produces a lot of output which is why here  
 "nloptions noprint" is specified \*/  
proc neural data = train dmdbcat = boston random = 29;  
 nloptions noprint;  
 archi MLP hidden=100;  
 input \_0 - \_12 / level = interval;  
 target Target / level = interval;  
 train maxiter = 250;  
 score data = test outfit = netfit out = gridout;  
run;

#### ii) Assess the model against the testing data.

ods select all;  
/\* Prediction on testing data \*/  
data scored(keep = sq\_error P\_Target Target);  
 set gridout;  
 sq\_error = (P\_Target - Target)\*\*2;  
run;  
  
/\* Determine mean squared error \*/  
proc means data = scored mean;  
var sq\_error;

The MEANS Procedure  
  
 Analysis Variable : sq\_error   
   
 Mean  
 ------------  
 16.1149499  
 ------------

# 7 Unsupervised Machine Learning

## 7.1 KMeans Clustering

proc import out = iris  
 datafile = 'C:/Users/iris.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
data iris;  
 length Species $ 20;  
 set iris;  
 if (Target = 0) then Species = "Setosa";  
 if (Target = 1) then Species = "Versicolor";  
 if (Target = 2) then Species = "Virginica";  
run;  
  
proc fastclus data=iris maxclusters=3 out=kmeans random = 29; \* noprint;  
 var PetalLength PetalWidth SepalLength SepalWidth;  
run;  
  
proc freq data = kmeans;  
 tables Species\*Cluster;  
run;

The FASTCLUS Procedure  
 Replace=FULL Radius=0 Maxclusters=3 Maxiter=1  
  
 Initial Seeds  
   
Cluster PetalLength PetalWidth SepalLength SepalWidth  
---------------------------------------------------------------------------  
 1 6.700000000 2.200000000 7.700000000 3.800000000  
 2 1.500000000 0.400000000 5.700000000 4.400000000  
 3 4.500000000 1.700000000 4.900000000 2.500000000  
  
 Criterion Based on Final Seeds = 0.3712  
  
 Cluster Summary  
   
 Maximum Distance  
 RMS Std from Seed Radius Nearest  
 Cluster Frequency Deviation to Observation Exceeded Cluster  
 ------------------------------------------------------------------------  
 1 33 0.3883 1.2923 3  
 2 50 0.2788 1.2394 3  
 3 67 0.4180 1.8532 1  
  
 Cluster Summary  
   
 Distance Between  
 Cluster Cluster Centroids  
 ----------------------------  
 1 1.8341  
 2 3.4222  
 3 1.8341  
  
 Statistics for Variables  
   
 Variable Total STD Within STD R-Square RSQ/(1-RSQ)  
 ---------------------------------------------------------------------  
 PetalLength 1.76442 0.42974 0.941475 16.086593  
 PetalWidth 0.76316 0.23898 0.903258 9.336801  
 SepalLength 0.82807 0.44824 0.710915 2.459187  
 SepalWidth 0.43359 0.32558 0.443729 0.797684  
 OVER-ALL 1.06880 0.37038 0.881525 7.440564  
  
 Pseudo F Statistic = 546.88  
  
 Approximate Expected Over-All R-Squared = 0.62721  
  
 Cubic Clustering Criterion = 24.526  
  
 WARNING: The two values above are invalid for correlated variables.  
  
 Cluster Means  
   
Cluster PetalLength PetalWidth SepalLength SepalWidth  
---------------------------------------------------------------------------  
 1 5.827272727 2.127272727 6.900000000 3.096969697  
 2 1.464000000 0.244000000 5.006000000 3.418000000  
 3 4.452238806 1.453731343 5.947761194 2.761194030  
  
 Cluster Standard Deviations  
   
Cluster PetalLength PetalWidth SepalLength SepalWidth  
---------------------------------------------------------------------------  
 1 0.4577613511 0.2401467354 0.5012484414 0.2909948974  
 2 0.1735111594 0.1072095031 0.3524896872 0.3810243980  
 3 0.5360795421 0.3011736428 0.4831582365 0.2953966126  
   
   
   
 The FREQ Procedure  
  
 Table of Species by CLUSTER  
  
 Species CLUSTER(Cluster)  
  
 Frequency |  
 Percent |  
 Row Pct |  
 Col Pct | 1| 2| 3| Total  
 -----------+--------+--------+--------+  
 Setosa | 0 | 50 | 0 | 50  
 | 0.00 | 33.33 | 0.00 | 33.33  
 | 0.00 | 100.00 | 0.00 |  
 | 0.00 | 100.00 | 0.00 |  
 -----------+--------+--------+--------+  
 Versicolor | 0 | 0 | 50 | 50  
 | 0.00 | 0.00 | 33.33 | 33.33  
 | 0.00 | 0.00 | 100.00 |  
 | 0.00 | 0.00 | 74.63 |  
 -----------+--------+--------+--------+  
 Virginica | 33 | 0 | 17 | 50  
 | 22.00 | 0.00 | 11.33 | 33.33  
 | 66.00 | 0.00 | 34.00 |  
 | 100.00 | 0.00 | 25.37 |  
 -----------+--------+--------+--------+  
 Total 33 50 67 150  
 22.00 33.33 44.67 100.00

## 7.2 Spectral Clustering

proc iml;  
 submit / R;  
 iris = read.csv('C:/Users/iris.csv')  
 iris$Species = ifelse(iris$Target == 0, "Setosa",   
 ifelse(iris$Target == 1, "Versicolor", "Virginica"))  
 features <- as.matrix(subset(iris, select = c(PetalLength,   
 PetalWidth, SepalLength, SepalWidth)))  
 library(kernlab)   
 set.seed(29)  
 spectral <- specc(features, centers = 3, iterations = 10, nystrom.red = TRUE)  
 labels <- as.data.frame(spectral)   
 table(iris$Species, labels$spectral)  
 endsubmit;  
quit;

1 2 3  
 Setosa 50 0 0  
 Versicolor 0 47 3  
 Virginica 0 3 47

## 7.3 Ward Hierarchical Clustering

proc import out = iris  
 datafile = 'C:/Users/iris.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
  
data iris;  
 length Species $ 20;  
 set iris;  
 if (Target = 0) then Species = "Setosa";  
 if (Target = 1) then Species = "Versicolor";  
 if (Target = 2) then Species = "Virginica";  
run;  
  
proc cluster data = iris method = ward print=15 ccc pseudo noprint;  
 var petal: sepal:;  
 copy species;  
run;  
  
proc tree noprint ncl=3 out=out;  
 copy petal: sepal: species;  
run;  
  
proc freq data = out;  
 tables Species\*Cluster;  
run;

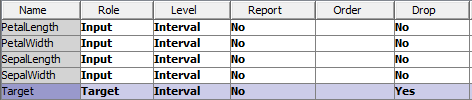
## 7.4 DBSCAN

proc iml;  
 submit / R;  
 iris = read.csv('C:/Users/iris.csv')  
 iris$Species = ifelse(iris$Target == 0, "Setosa",   
 ifelse(iris$Target == 1, "Versicolor", "Virginica"))  
 features <- as.matrix(subset(iris, select = c(PetalLength,   
 PetalWidth, SepalLength, SepalWidth)))  
 library(dbscan)  
 set.seed(29)  
 dbscan <- dbscan(features, eps = 0.5)  
 labels <- dbscan$cluster   
 table(iris$Species, labels)  
 endsubmit;  
quit;

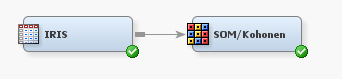
labels  
 0 1 2  
 Setosa 1 49 0  
 Versicolor 6 0 44  
 Virginica 10 0 40

## 7.5 Self-organizing map

Currently, there is not a self-organizing map procedure available in Base SAS. Therefore, the best method to create a self-organizing map currently is using SAS Enterprise Miner. First, you need to read in the Iris data set, setting the Species/Target variable to be dropped before investigation.

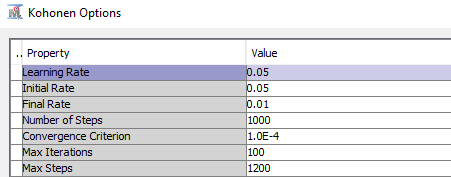
Output: 

Then create the following diagram in SAS Enterprise Miner:

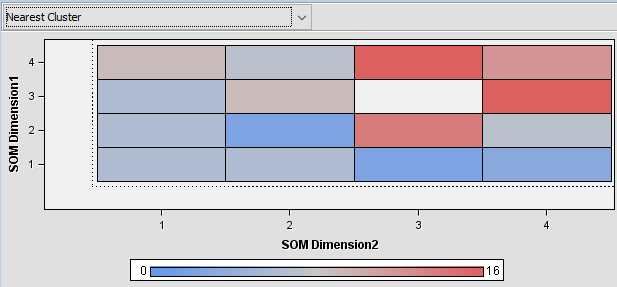
Output: 

For the SOM/Kohonen node set the following options:

1. Choose the Kohonen SOM method.
2. Set row and column to both be 4.
3. Under the "Kohonen" options section, set "Use Defaults" to "No", and open the Kohonen Options window by clicking the ... box.
4. Set the following options in the popup window:

Output: 

This model produces the following output which is similar to the output of R and Python:

Output: 

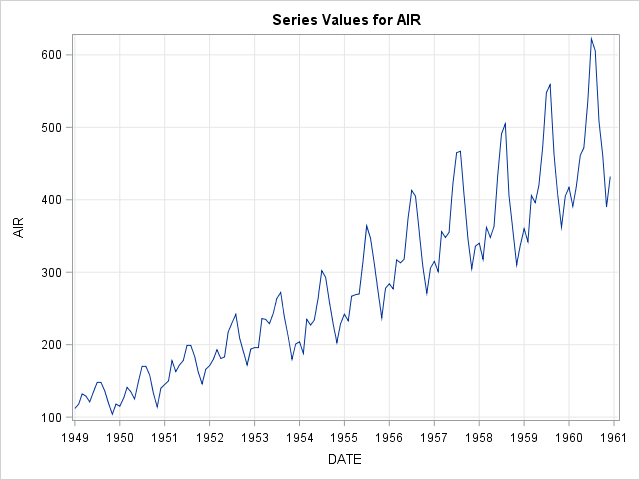
# 8 Forecasting

## 8.1 Fit an ARIMA model to a timeseries.

### a) Plot the timeseries.

proc import out = air  
 datafile = 'C:/Users/air.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
proc timeseries data = air plot = series;  
 id date interval = month;  
 var air;  
run;

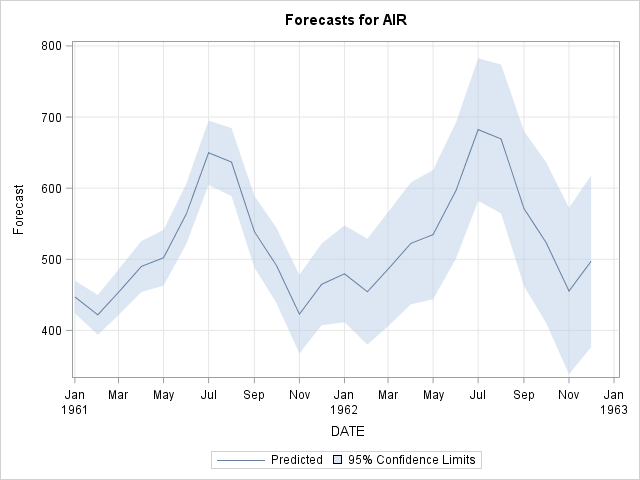
The TIMESERIES Procedure  
  
 Input Data Set  
  
 Name WORK.AIR  
 Label   
 Time ID Variable DATE  
 Time Interval MONTH  
 Length of Seasonal Cycle 12

Output: 

### b) Fit an ARIMA model, predict 2 years (24 months) out, and plot predictions.

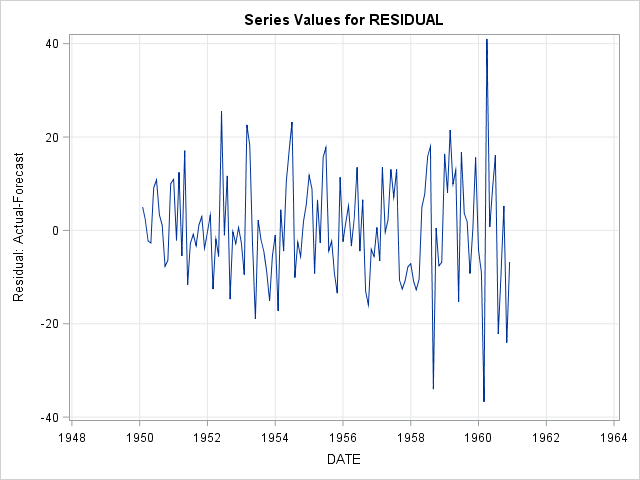
proc import out = air  
 datafile = 'C:/Users/air.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
proc arima data = air;  
 identify var = air(1,12);  
 estimate q=(1)(12) noint method=ml;  
 forecast id=date interval=month out=forecast;  
run;  
  
/\* SAS automatically predicts 2 years out and plots the predictions \*/

The ARIMA Procedure  
  
 Name of Variable = AIR  
  
 Period(s) of Differencing 1,12  
 Mean of Working Series 0.183206  
 Standard Deviation 12.3095  
 Number of Observations 131  
 Observation(s) eliminated by differencing 13  
  
 Autocorrelations  
   
Lag Covariance Correlation -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1  
  
 0 151.524 1.00000 | |\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|   
 1 -46.944257 -.30981 | \*\*\*\*\*\*| . |   
 2 14.448005 0.09535 | . |\*\* . |   
 3 -14.681265 -.09689 | . \*\*| . |   
 4 -15.000093 -.09900 | . \*\*| . |   
 5 9.243052 0.06100 | . |\* . |   
 6 -0.043609 -.00029 | . | . |   
 7 -8.501764 -.05611 | . \*| . |   
 8 -9.237759 -.06097 | . \*| . |   
 9 26.655581 0.17592 | . |\*\*\*\* |   
 10 -21.255578 -.14028 | .\*\*\*| . |   
 11 10.566554 0.06974 | . |\* . |   
 12 -20.254692 -.13367 | .\*\*\*| . |   
 13 13.209419 0.08718 | . |\*\* . |   
 14 0.377976 0.00249 | . | . |   
 15 9.899368 0.06533 | . |\* . |   
 16 -16.540607 -.10916 | . \*\*| . |   
 17 -0.051148 -.00034 | . | . |   
 18 6.671223 0.04403 | . |\* . |   
 19 -17.265401 -.11395 | . \*\*| . |   
 20 -13.829785 -.09127 | . \*\*| . |   
 21 6.355321 0.04194 | . |\* . |   
 22 -23.834143 -.15730 | .\*\*\*| . |   
 23 39.033032 0.25760 | . |\*\*\*\*\* |   
 24 8.005913 0.05284 | . |\* . |   
  
 "." marks two standard errors  
  
 Inverse Autocorrelations  
   
 Lag Correlation -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1  
  
 1 0.32010 | . |\*\*\*\*\*\* |  
 2 0.12065 | . |\*\*. |  
 3 0.20842 | . |\*\*\*\* |  
 4 0.22218 | . |\*\*\*\* |  
 5 0.02341 | . | . |  
 6 0.07164 | . |\* . |  
 7 0.13300 | . |\*\*\* |  
 8 0.13465 | . |\*\*\* |  
 9 -0.00310 | . | . |  
 10 0.12561 | . |\*\*\* |  
 11 0.09764 | . |\*\*. |  
 12 0.14768 | . |\*\*\* |  
 13 -0.00757 | . | . |  
 14 0.06215 | . |\* . |  
 15 0.11303 | . |\*\*. |  
 16 0.14325 | . |\*\*\* |  
 17 0.06835 | . |\* . |  
 18 0.12971 | . |\*\*\* |  
 19 0.19013 | . |\*\*\*\* |  
 20 0.13470 | . |\*\*\* |  
 21 0.05669 | . |\* . |  
 22 0.10188 | . |\*\*. |  
 23 -0.11120 | .\*\*| . |  
 24 -0.08229 | .\*\*| . |  
  
 Partial Autocorrelations  
   
 Lag Correlation -1 9 8 7 6 5 4 3 2 1 0 1 2 3 4 5 6 7 8 9 1  
  
 1 -0.30981 | \*\*\*\*\*\*| . |  
 2 -0.00070 | . | . |  
 3 -0.07472 | . \*| . |  
 4 -0.16676 | \*\*\*| . |  
 5 -0.01515 | . | . |  
 6 0.01829 | . | . |  
 7 -0.08809 | .\*\*| . |  
 8 -0.13389 | \*\*\*| . |  
 9 0.15641 | . |\*\*\* |  
 10 -0.05875 | . \*| . |  
 11 -0.05243 | . \*| . |  
 12 -0.11501 | .\*\*| . |  
 13 0.05996 | . |\* . |  
 14 0.00432 | . | . |  
 15 0.03659 | . |\* . |  
 16 -0.08660 | .\*\*| . |  
 17 -0.02783 | . \*| . |  
 18 0.01912 | . | . |  
 19 -0.11379 | .\*\*| . |  
 20 -0.25688 | \*\*\*\*\*| . |  
 21 0.00684 | . | . |  
 22 -0.22457 | \*\*\*\*| . |  
 23 0.08453 | . |\*\*. |  
 24 0.11749 | . |\*\*. |  
  
 Autocorrelation Check for White Noise  
   
 To Chi- Pr >  
Lag Square DF ChiSq ---------------Autocorrelations---------------  
  
 6 17.23 6 0.0085 -0.310 0.095 -0.097 -0.099 0.061 -0.000  
 12 28.77 12 0.0043 -0.056 -0.061 0.176 -0.140 0.070 -0.134  
 18 32.64 18 0.0184 0.087 0.002 0.065 -0.109 -0.000 0.044  
 24 51.36 24 0.0009 -0.114 -0.091 0.042 -0.157 0.258 0.053  
  
 Maximum Likelihood Estimation  
   
 Standard Approx  
 Parameter Estimate Error t Value Pr > |t| Lag  
  
 MA1,1 0.30876 0.08430 3.66 0.0002 1  
 MA2,1 0.10738 0.10189 1.05 0.2919 12  
  
 Variance Estimate 137.5235  
 Std Error Estimate 11.72704  
 AIC 1019.003  
 SBC 1024.753  
 Number of Residuals 131  
  
 Correlations of Parameter  
 Estimates  
   
 Parameter MA1,1 MA2,1  
  
 MA1,1 1.000 -0.046  
 MA2,1 -0.046 1.000  
  
 Autocorrelation Check of Residuals  
   
 To Chi- Pr >  
Lag Square DF ChiSq ---------------Autocorrelations---------------  
  
 6 4.99 4 0.2886 -0.003 0.049 -0.103 -0.151 0.012 -0.021  
 12 10.95 10 0.3617 -0.088 -0.063 0.127 -0.107 0.045 -0.022  
 18 14.76 16 0.5423 0.078 0.048 0.022 -0.121 -0.038 -0.018  
 24 40.63 22 0.0091 -0.176 -0.141 -0.023 -0.087 0.288 0.141  
  
 Model for variable AIR  
  
 Period(s) of Differencing 1,12  
  
 No mean term in this model.  
  
 Moving Average Factors  
  
 Factor 1: 1 - 0.30876 B\*\*(1)   
 Factor 2: 1 - 0.10738 B\*\*(12)  
  
 Forecasts for variable AIR  
   
 Obs Forecast Std Error 95% Confidence Limits  
  
 145 447.0542 11.7270 424.0697 470.0388  
 146 421.8779 14.2560 393.9366 449.8192  
 147 453.5244 16.3995 421.3819 485.6669  
 148 489.9028 18.2936 454.0480 525.7576  
 149 502.1847 20.0092 462.9675 541.4019  
 150 564.2263 21.5888 521.9130 606.5396  
 151 649.7981 23.0605 604.6003 694.9959  
 152 636.7155 24.4438 588.8066 684.6244  
 153 538.9216 25.7528 488.4470 589.3962  
 154 491.0684 26.9985 438.1523 543.9845  
 155 422.8237 28.1892 367.5739 478.0735  
 156 464.7521 29.3316 407.2633 522.2409  
 157 479.5823 34.7179 411.5364 547.6283  
 158 454.4060 37.9567 380.0122 528.7998  
 159 486.0525 40.9401 405.8115 566.2936  
 160 522.4309 43.7203 436.7406 608.1212  
 161 534.7128 46.3341 443.8997 625.5259  
 162 596.7544 48.8080 501.0925 692.4164  
 163 682.3262 51.1625 582.0496 782.6028  
 164 669.2436 53.4133 564.5556 773.9317  
 165 571.4497 55.5730 462.5287 680.3707  
 166 523.5965 57.6518 410.6011 636.5920  
 167 455.3518 59.6583 338.4237 572.2798  
 168 497.2802 61.5994 376.5476 618.0128

Output: 

### c) Plot residuals of predictions and known values.

proc import out = air  
 datafile = 'C:/Users/air.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc timeseries data = air plot = series;  
 id date interval = month;  
 var air;  
run;  
proc arima data = air;  
 identify var = air(1,12);  
 estimate q=(1)(12) noint method=ml;  
 forecast id=date interval=month out=forecast;  
run;  
proc timeseries data = forecast plot = series;  
 id date interval = month;  
 var residual;  
run;

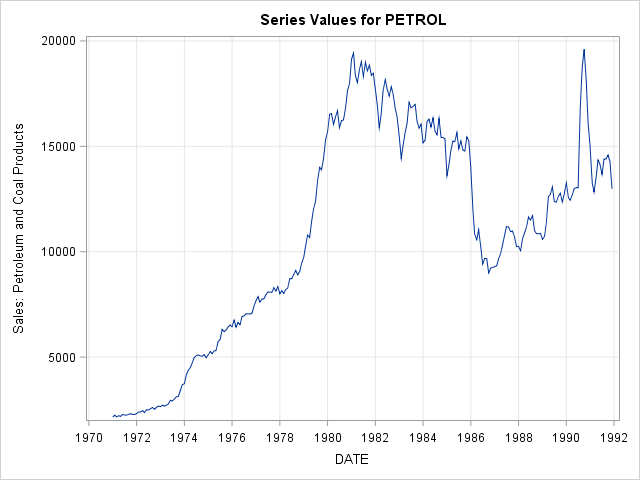
Output: 

## 8.2 Fit a Simple Exponential Smoothing model to a timeseries.

### a) Plot the timeseries.

proc import out = usecon  
 datafile = 'C:/Users/usecon.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc timeseries data = usecon plot = series;  
 id date interval = month;  
 var petrol;  
run;

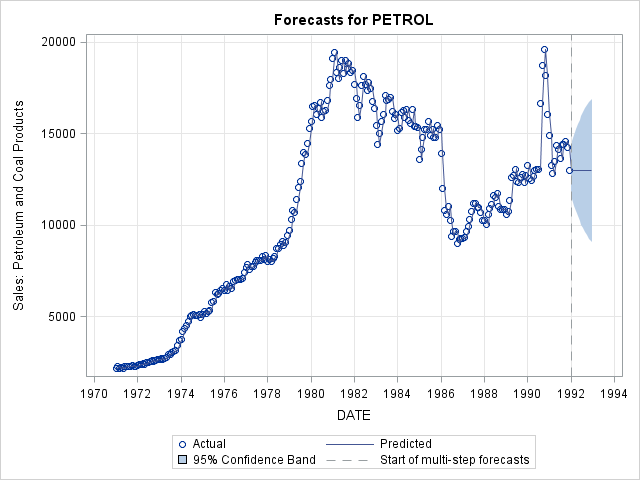
The TIMESERIES Procedure  
  
 Input Data Set  
  
 Name WORK.USECON  
 Label   
 Time ID Variable DATE  
 Time Interval MONTH  
 Length of Seasonal Cycle 12

Output: 

### b) Fit a Simple Exponential Smoothing model, predict 2 years (24 months) out and plot predictions.

proc import out = usecon  
 datafile = 'C:/Users/usecon.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc esm data = usecon out = forecast lead = 24 plot = forecasts;  
 id date interval = month;  
 forecast petrol / model = simple;  
run;

The ESM Procedure  
  
 Input Data Set  
  
 Name WORK.USECON  
 Label   
 Time ID Variable DATE  
 Time Interval MONTH  
 Length of Seasonal Cycle 12  
 Forecast Horizon 24  
  
 Variable Information  
  
 Name PETROL  
 Label   
 First JAN1971  
 Last DEC1991  
 Number of Observations Read 252

Output: 

## 8.3 Fit a Holt-Winters model to a timeseries.

### a) Plot the timeseries.

proc import out = usecon  
 datafile = 'C:/Users/usecon.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc timeseries data = usecon plot = series;  
 id date interval = month;  
 var vehicle;  
run;

Output: output

### b) Fit a Holt-Winters additive model, predict 2 years (24 months) out and plot predictions.

proc esm data = usecon out = forecast lead = 24 plot = forecasts;  
 id date interval = month;  
 forecast vehicle / model = addwinters;  
run;

Output: output

# 9 Model Evaluation & Selection

## 9.1 Evaluate the accuracy of regression models.

### a) Evaluation on training data.

/\* Notice we are re-using data sets but it is good to re-read the original   
 version back into the environment \*/  
proc import out = train  
 datafile = 'C:/Users/boston\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/boston\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
  
/\* Random Forest Regression Model \*/  
ods select none;  
proc hpforest data = train ;  
 input \_0-\_12 / level = interval;  
 target Target / level = interval;  
 save file = 'rfMod.bin';  
run;  
ods select all;  
  
/\* Evaluation on training data \*/  
proc hp4score data = train;  
 score file = 'rfMod.bin' out = scored\_train;  
run;  
  
/\* Determine coefficient of determination score \*/  
proc iml;  
 use scored\_train;  
 read all var \_ALL\_ into data;  
 close scored\_train;  
 tip = data[,1];  
 pred\_rf = data[,2];  
 r2\_rf = 1 - ( (sum((tip - pred\_rf)##2)) / (sum((tip - mean(tip))##2)) );  
 print(r2\_rf);  
quit;

The HP4SCORE Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TRAIN V9 Input On Client  
 WORK.SCORED\_TRAIN V9 Output On Client  
  
 Number of Observations  
   
 Type N  
  
 Number of Observations Read 354  
 Number of Observations Used 354  
 Sum of Frequencies Used 354  
   
   
   
 r2\_rf  
  
 0.9751727

### b) Evaluation on testing data.

/\* Random Forest Regression Model (rfMod) \*/  
  
/\* Evaluation on testing data \*/  
proc hp4score data = test;  
 score file = 'rfMod.bin' out = scored\_test;  
run;  
  
/\* Determine coefficient of determination score \*/  
proc iml;  
 use scored\_test;  
 read all var \_ALL\_ into data;  
 close scored\_test;  
 tip = data[,1];  
 pred\_rf = data[,2];  
 r2\_rf = 1 - ( (sum((tip - pred\_rf)##2)) / (sum((tip - mean(tip))##2)) );  
 print(r2\_rf);  
quit;

The HP4SCORE Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TEST V9 Input On Client  
 WORK.SCORED\_TEST V9 Output On Client  
  
 Number of Observations  
   
 Type N  
  
 Number of Observations Read 152  
 Number of Observations Used 152  
 Sum of Frequencies Used 152  
   
   
   
 r2\_rf  
  
 0.8890058

The formula used here for the coefficient score is based off the Python skearn formula for [r2\_score](http://scikit-learn.org/stable/modules/model_evaluation.html#r2-score-the-coefficient-of-determination).

[REG Procedure](#reg), [SCORE Procedure](#score), [IML Procedure](#iml), [HPFOREST Procedure](#hpforest), [HP4SCORE Procedure](#hp4score)

## 9.2 Evaluate the accuracy of classification models.

### a) Evaluation on training data.

/\* Notice we are re-using data sets but it is good to re-read the original   
 versions back into the environment \*/  
proc import out = train  
 datafile = 'C:/Users/digits\_train.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
proc import out = test  
 datafile = 'C:/Users/digits\_test.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
  
/\* Random Forest Classification Model \*/  
ods select none;  
proc hpforest data = train;  
 input \_0-\_63 / level = interval;  
 target Target / level = nominal;  
 save file = 'rfMod.bin';  
run;  
ods select all;  
  
/\* Evaluation on training data \*/  
proc hp4score data = train;  
 score file = 'rfMod.bin' out = scored;  
run;  
  
data scored(keep = Target I\_Target correct);  
 set scored;  
 correct = (I\_Target = Target);  
run;  
  
/\* Determine accuracy score \*/  
proc iml;  
 use scored;  
 read all var \_ALL\_ into data;  
 close scored;  
   
 accuracy\_forest = (1/nrow(data)) \* sum(data[,2]);  
   
 print(accuracy\_forest);  
quit;

The HP4SCORE Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TRAIN V9 Input On Client  
 WORK.SCORED V9 Output On Client  
  
 Number of Observations  
   
 Type N  
  
 Number of Observations Read 1257  
 Number of Observations Used 1257  
 Sum of Frequencies Used 1257  
   
   
   
 accuracy\_forest  
  
 1

### b) Evaluation on testing data.

/\* Random Forest Classification Model (rfMod) \*/  
  
/\* Evaluation on testing data \*/  
proc hp4score data = test;  
 score file = 'rfMod.bin' out = scored;  
run;  
  
data scored(keep = Target I\_Target correct);  
 set scored;  
 correct = (I\_Target = Target);  
run;  
  
/\* Determine accuracy score \*/  
proc iml;  
 use scored;  
 read all var \_ALL\_ into data;  
 close scored;  
   
 accuracy\_forest = (1/nrow(data)) \* sum(data[,2]);  
   
 print(accuracy\_forest);  
quit;

The HP4SCORE Procedure  
  
 Performance Information  
  
 Execution Mode Single-Machine  
 Number of Threads 4   
  
 Data Access Information  
   
 Data Engine Role Path  
  
 WORK.TEST V9 Input On Client  
 WORK.SCORED V9 Output On Client  
  
 Number of Observations  
   
 Type N  
  
 Number of Observations Read 540  
 Number of Observations Used 540  
 Sum of Frequencies Used 540  
   
   
   
 accuracy\_forest  
  
 0.9703704

## 9.3 Evaluation with cross validation.

### a) KFold

The HPFOREST output for the 5 models has been removed from the output in this tutorial for the sake of space.

proc import out = breastcancer  
 datafile = 'C:/Users/breastcancer.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
data folds;  
 set breastcancer;  
 \*randomly assign observation to one of K groups;  
 call streaminit(29);  
 rand=ceil(5\*rand('UNIFORM'));  
 output;  
run;  
  
%macro hp\_KFolds();  
   
data train1 test1 train2 test2 train3 test3   
 train4 test4 train5 test5;  
 set folds;  
 %do i = 1 %to 5;  
 %do j = 1 %to 5;  
 if (rand = &j) then do;  
 if (&i ^= &j) then output train&i;  
 else output test&i;  
 end;  
 %end;  
 %end;  
 drop rand;  
run;  
  
%do i = 1 %to 5;  
   
ods select none;  
proc hpforest data = train&i;  
 input \_0-\_29 / level = interval;  
 target Target / level = nominal;  
 save file = 'hpbreastcancer&i.bin';  
run;  
   
proc hp4score data = test&i;  
 score file = 'hpbreastcancer&i.bin' out = scored\_&i;  
run;  
ods select all;  
   
data scored\_&i;  
 set scored\_&i;  
 correct = (I\_Target = Target);  
run;  
   
proc freq data = scored\_&i noprint;  
 tables correct / out=FreqCount&i;  
run;   
  
%end;  
   
%mend;  
  
%hp\_KFolds()  
  
data FreqCount;  
 set FreqCount1 FreqCount2 FreqCount3 FreqCount4 FreqCount5;  
 if (correct = 1);  
run;  
  
proc means data = FreqCount mean std;  
 var PERCENT;  
run;

The MEANS Procedure  
  
 Analysis Variable : PERCENT Percent of Total Frequency  
   
 Mean Std Dev  
 ----------------------------  
 96.0918078 1.8699234  
 ----------------------------

### b) ShuffleSplit

The HPFOREST output for the 5 models has been removed from the output in this tutorial for the sake of space.

proc import out = breastcancer  
 datafile = 'C:/Users/breastcancer.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
proc surveyselect data = breastcancer out = cv seed = 29 samprate = 0.7   
 outall reps = 5;  
run;  
   
data train1 train2 train3 train4 train5 test1 test2 test3 test4 test5;  
 set cv;  
 if (replicate = 1) then do;  
 if (selected = 1) then output train1;  
 else output test1;  
 end;  
 if (replicate = 2) then do;  
 if (selected = 1) then output train2;  
 else output test2;  
 end;  
 if (replicate = 3) then do;  
 if (selected = 1) then output train3;  
 else output test3;  
 end;  
 if (replicate = 4) then do;  
 if (selected = 1) then output train4;  
 else output test4;  
 end;  
 if (replicate = 5) then do;  
 if (selected = 1) then output train5;  
 else output test5;  
 end;  
run;  
   
%macro hp\_replicate();  
  
%do i = 1 %to 5;  
   
ods select none;  
proc hpforest data = train&i;  
 input \_0-\_29 / level = interval;  
 target Target / level = nominal;  
 save file = 'hpbreastcancer&i.bin';  
run;  
  
proc hp4score data = test&i;  
 score file = 'hpbreastcancer&i.bin' out = scored\_&i;  
run;  
ods select all;  
   
data scored\_&i;  
 set scored\_&i;  
 correct = (I\_Target = Target);  
run;  
   
proc freq data = scored\_&i noprint;  
 tables correct / out=FreqCount&i;  
run;   
  
%end;  
  
%mend;  
   
%hp\_replicate()  
   
data FreqCount;  
 set FreqCount1 FreqCount2 FreqCount3 FreqCount4 FreqCount5;  
 if (correct = 1);  
run;  
  
proc means data = FreqCount mean std;  
 var PERCENT;  
run;

The SURVEYSELECT Procedure  
  
 Selection Method Simple Random Sampling  
  
 Input Data Set BREASTCANCER  
 Random Number Seed 29  
 Sampling Rate 0.7  
 Sample Size 399  
 Selection Probability 0.70123  
 Sampling Weight 0  
 Number of Replicates 5  
 Total Sample Size 1995  
 Output Data Set CV  
   
   
   
 The MEANS Procedure  
  
 Analysis Variable : PERCENT Percent of Total Frequency  
   
 Mean Std Dev  
 ----------------------------  
 95.7647059 0.6443795  
 ----------------------------

# 10 Text Analytics

# 11 Deep Learning

# Appendix

## 1 Built-in SAS Data Types

* [CHAR](#char)
* [DOUBLE](#double)

## 2 SAS Procedures

### [COMPARE Procedure](http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm" \l "n0c1y14wyd3u7yn1dmfcpaejllsn.htm)

### [CONTENTS Procedure](http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm" \l "n1hqa4dk5tay0an15nrys1iwr5o2.htm)

### [CORR Procedure](http://support.sas.com/documentation/cdl/en/procstat/66703/HTML/default/viewer.htm" \l "procstat_corr_overview.htm)

### [FCMP Procedure](http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm" \l "n1aozmc89vjkpzn1q6a54nleh56o.htm)

### [EXPORT Procedure](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/viewer.htm" \l "p09k160vk93xxhn171zz1z6551w2.htm)

### [FREQ Procedure](https://support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm" \l "statug_freq_sect006.htm)

### [GENMOD Procedure](https://support.sas.com/documentation/cdl/en/statuggenmod/61787/PDF/default/statuggenmod.pdf)

### [HP4SCORE Procedure](https://support.sas.com/documentation/onlinedoc/miner/em14/emhpprcref.pdf)

### [HPFOREST Procedure](https://support.sas.com/documentation/onlinedoc/miner/em14/emhpprcref.pdf)

### [HPSPLIT Procedure](http://support.sas.com/documentation/cdl/en/stathpug/66410/HTML/default/viewer.htm" \l "stathpug_hpsplit_overview.htm)

### [HPSVM Procedure](http://documentation.sas.com/?docsetId=emhpprcref&docsetVersion=14.2&docsetTarget=emhpprcref_hpsvm_overview.htm&locale=en)

### [IML Procedure](https://support.sas.com/documentation/cdl/en/imlug/63541/PDF/default/imlug.pdf)

### [IMPORT Procedure](http://support.sas.com/documentation/cdl/en/acpcref/69731/HTML/default/viewer.htm" \l "p0jf3o1i67m044n1j0kz51ifhpvs.htm)

### [MEANS Procedure](http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm" \l "p0f0fjpjeuco4gn1ri963f683mi4.htm)

### [PRINCOMP Procedure](https://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm" \l "princomp_toc.htm)

### [PRINT Procedure](http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm" \l "p10qiuo2yicr4qn17rav8kptnjpu.htm)

### [REG Procedure](https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm" \l "statug_reg_sect007.htm)

### [SCORE Procedure](https://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm" \l "statug_score_sect001.htm)

### [SGPLOT Procedure](http://support.sas.com/documentation/cdl/en/grstatproc/69716/HTML/default/viewer.htm" \l "p1t32i8511t1gfn17sw07yxtazad.htm)

* [histogram](http://support.sas.com/documentation/cdl/en/grstatproc/69716/HTML/default/viewer.htm#n17xrpcduau1f8n1c1nhe477pv18.htm)
* [inset](http://support.sas.com/documentation/cdl/en/grstatproc/69716/HTML/default/viewer.htm#p0hz27ehuzdd6pn0zaic6x52pkav.htm)
* [reg](http://support.sas.com/documentation/cdl/en/grstatproc/69716/HTML/default/viewer.htm#p0mn6vl6clqbgyn1ivs69lezdxhf.htm)
* [scatter](http://support.sas.com/documentation/cdl/en/grstatproc/69716/HTML/default/viewer.htm#p1lcbd3lhs3t3bn1jk6d8sjt2yqx.htm)
* [vbox](http://support.sas.com/documentation/cdl/en/grstatproc/69716/HTML/default/viewer.htm#n1waawwbez01ppn15dn9ehmxzihf.htm)

### [SGSCATTER Procedure](http://support.sas.com/documentation/cdl/en/grstatproc/62603/HTML/default/viewer.htm" \l "sgscatter-syn.htm)

### [SORT Procedure](http://support.sas.com/documentation/cdl/en/proc/70377/HTML/default/viewer.htm" \l "p02bhn81rn4u64n1b6l00ftdnxge.htm)

### [SQL Procedure](https://support.sas.com/documentation/cdl/en/sqlproc/69822/HTML/default/viewer.htm" \l "n1oihmdy7om5rmn1aorxui3kxizl.htm)

### [SURVEYSELECT Procedure](https://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm" \l "statug_surveyselect_sect001.htm)

## 3 [SAS DATA step](http://support.sas.com/documentation/cdl/en/basess/68381/HTML/default/viewer.htm#n053a58fwk57v7n14h8x7y7u34y4.htm)

Statements:

### [%include](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm" \l "a000214504.htm)

### [if-then/else](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm" \l "a000202239.htm)

### [infile](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm" \l "a000146932.htm)

### [input](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm" \l "a000146292.htm)

### [merge](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm" \l "a000202970.htm)

### [output](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm" \l "a000194540.htm)

### [set](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm" \l "a000173782.htm)

### [where](http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm" \l "a000202951.htm)

# Alphabetical Index

## [CHAR](http://support.sas.com/documentation/cdl/en/fedsqlref/67364/HTML/default/viewer.htm" \l "n19bf2z7e9p646n0z224cokuj567.htm)

The SAS implementation of a string as a fixed-length character string of length *n*.

## [Data Frame](http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm" \l "a001005709.htm)

A two-dimensional tabular structure with labeled axes (rows and columns), where data observations are represented by rows and data variables are represented by columns.

## [Dictionary](https://www.google.com/url?q=http://support.sas.com/resources/papers/proceedings12/147-2012.pdf&ust=1496196720000000&usg=AFQjCNGcqLlU2Ur5Qv_62TM4zJEQY6LjTA&hl=en-US)

An associative array which is indexed by keys which map to values. Therefore, a dictionary is an unordered set of key:value pairs where each key is unique. In SAS, a dictionary can be implemented using a hash table. Please see the following example.

/\* Results will be displayed in the log \*/  
data class\_dict;  
declare hash mydict();  
mydict.defineKey("Name");  
mydict.defineData("Age");  
mydict.defineDone();  
do while (not eof);  
 set sashelp.class end = eof;  
 rc = mydict.add();  
 output;  
end;  
Name = 'James';  
rc = mydict.find();  
put rc= Name= Age=;

Output: output

## [DOUBLE](http://support.sas.com/documentation/cdl/en/fedsqlref/67364/HTML/default/viewer.htm" \l "n19bf2z7e9p646n0z224cokuj567.htm)

A decimal point number implemented as a 64-bit double precision, floating-point number.

## [Series](http://support.sas.com/documentation/cdl/en/lestmtsref/69738/HTML/default/viewer.htm" \l "p08do6szetrxe2n136ush727sbuo.htm)

A series is a one-dimension data frame, which is also called an array in SAS. Please see the following example.

array my\_array{4} a1-a4 (1 3 5 9);

For more information on SAS packages and functions, along with helpful examples, please see [SAS](https://support.sas.com/en/support-home.html).