SAS Tutorial

Table of Contents

[1 Reading in Data and Basic Statistical Functions 5](#_Toc486249184)

[1.1 Read in the data. 5](#_Toc486249185)

[a) Read the data in as a .csv file. 5](#_Toc486249186)

[b) Read the data in as a .xls file. 5](#_Toc486249187)

[c) Read the data in as a .json file. 5](#_Toc486249188)

[1.2 Find the dimensions of the data set. 5](#_Toc486249189)

[1.3 Find basic information about the data set. 6](#_Toc486249190)

[1.4 Look at the first 5 observations. 7](#_Toc486249191)

[1.5 Calculate mean of numeric variables. 7](#_Toc486249192)

[1.6 Compute summary statistics of the data set. 8](#_Toc486249193)

[1.7 Descriptive statistics functions applied to columns of the data set. 8](#_Toc486249194)

[1.8 Produce a one-way table to describe the frequency of a variable. 9](#_Toc486249195)

[a) Produce a one-way table of a discrete variable. 9](#_Toc486249196)

[b) Produce a one-way table of a categorical variable. 9](#_Toc486249197)

[1.9 Produce a two-way table to visualize the frequency of two categorical (or discrete) variables. 9](#_Toc486249198)

[1.10 Select a subset of the data that meets a certain criterion. 10](#_Toc486249199)

[1.11 Determine the correlation between two continuous variables. 11](#_Toc486249200)

[2 Basic Graphing and Plotting Functions 11](#_Toc486249201)

[2.1 Visualize a single continuous variable by producing a histogram. 11](#_Toc486249202)

[2.2 Visualize a single continuous variable by producing a boxplot. 12](#_Toc486249203)

[2.3 Visualize two continuous variables by producing a scatterplot. 13](#_Toc486249204)

[2.4 Visualize a relationship between two continuous variables by producing a scatterplot and a plotted line of best fit. 14](#_Toc486249205)

[2.5 Visualize a categorical variable by producing a bar chart. 15](#_Toc486249206)

[2.6 Visualize a continuous variable, grouped by a categorical variable, using side-by-side boxplots. 16](#_Toc486249207)

[More advanced side-by-side boxplot with color. 16](#_Toc486249208)

[3 Basic Data Wrangling and Manipulation 17](#_Toc486249209)

[3.1 Create a new variable in a data set as a function of existing variables in the data set. 17](#_Toc486249210)

[3.2 Create a new variable in a data set using if/else logic of existing variables in the data set. 18](#_Toc486249211)

[3.3 Create a new variable in a data set using mathemtical functions applied to existing variables in the data set. 18](#_Toc486249212)

[3.4 Drop variables from a data set. 19](#_Toc486249213)

[3.5 Sort a data set by a variable. 19](#_Toc486249214)

[a) Sort data set by a continuous variable. 19](#_Toc486249215)

[b) Sort data set by a categorical variable. 20](#_Toc486249216)

[3.6 Compute descriptive statistics of continuous variables, grouped by a categorical variable. 20](#_Toc486249217)

[3.7 Add a new row to the bottom of a data set. 21](#_Toc486249218)

[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. 21](#_Toc486249219)

[4 More Advanced Data Wrangling 22](#_Toc486249220)

[4.1 Drop observations with missing information. 22](#_Toc486249221)

[4.2 Merge two data sets together on a common variable. 24](#_Toc486249222)

[a) First, select specific columns of a data set to create two smaller data sets. 24](#_Toc486249223)

[b) Second, we want to merge the two smaller data sets on the common variable. 24](#_Toc486249224)

[c) Finally, we want to check to see if the merged data set is the same as the original data set. 25](#_Toc486249225)

[4.3 Merge two data sets together by index number only. 25](#_Toc486249226)

[a) First, select specific columns of a data set to create two smaller data sets. 25](#_Toc486249227)

[b) Second, we want to join the two smaller data sets. 26](#_Toc486249228)

[c) Finally, we want to check to see if the joined data set is the same as the original data set. 26](#_Toc486249229)

[4.4 Create a pivot table to summarize information about a data set. 26](#_Toc486249230)

[4.5 Return all unique values from a text variable. 27](#_Toc486249231)

[5 Preparation & Basic Regression 28](#_Toc486249232)

[5.1 Pre-process a data set using principal component analysis. 28](#_Toc486249233)

[5.2 Split data into training and testing data and export as a .csv file. 28](#_Toc486249234)

[5.3 Fit a logistic regression model. 29](#_Toc486249235)

[5.4 Fit a linear regression model. 31](#_Toc486249236)

[6 Regression & Machine Learning: Modeling & Prediction 31](#_Toc486249237)

[6.1 Fit a logistic regression model on training data and assess against testing data. 31](#_Toc486249238)

[a) Fit a logistic regression model on training data. 31](#_Toc486249239)

[b) Assess the model against the testing data. 33](#_Toc486249240)

[6.2 Fit a linear regression model on training data and assess against testing data. 34](#_Toc486249241)

[a) Fit a linear regression model on training data. 34](#_Toc486249242)

[b) Assess the model against the testing data. 35](#_Toc486249243)

[6.3 Fit a decision tree model on training data and assess against testing data. 36](#_Toc486249244)

[a) Fit a decision tree classification model. 36](#_Toc486249245)

[b) Fit a decision tree regression model. 39](#_Toc486249246)

[6.4 Fit a random forest model on training data and assess against testing data. 41](#_Toc486249247)

[a) Fit a random forest classification model. 41](#_Toc486249248)

[b) Fit a random forest regression model. 48](#_Toc486249249)

[6.5 Fit a gradient boosting model on training data and assess against testing data. 53](#_Toc486249250)

[a) Fit a gradient boosting classification model. 53](#_Toc486249251)

[b) Fit a gradient boosting regression model. 55](#_Toc486249252)

[6.6 Fit an extreme gradient boosting model on taining data and assess against testing data. 56](#_Toc486249253)

[a) Fit an extreme gradient boosting classification model. 56](#_Toc486249254)

[6.7 Fit a support vector model on training data and assess against testing data. 57](#_Toc486249255)

[a) Fit a support vector classification model. 57](#_Toc486249256)

[b) Fit a support vector regression model. 60](#_Toc486249257)

[6.8 Fit a neural network model on training data and assess against testing data. 60](#_Toc486249258)

[a) Fit a neural network classification model. 60](#_Toc486249259)

[b) Fit a neural network regression model. 62](#_Toc486249260)

[7 Unsupervised Machine Learning 64](#_Toc486249261)

[7.1 KMeans Clustering 64](#_Toc486249262)

[7.2 Spectral Clustering 66](#_Toc486249263)

[7.3 Ward Hierarchical Clustering 67](#_Toc486249264)

[7.4 DBSCAN 67](#_Toc486249265)

[7.5 Self-organizing map 68](#_Toc486249266)

[8 Forecasting 69](#_Toc486249267)

[8.1 Fit an ARIMA model to a timeseries. 69](#_Toc486249268)

[a) Plot the timeseries. 69](#_Toc486249269)

[b) Fit an ARIMA model, predict 2 years (24 months) out, and plot predictions. 70](#_Toc486249270)

[c) Plot residuals of predictions and known values. 71](#_Toc486249271)

[8.2 Fit a Simple Exponential Smoothing model to a timeseries. 72](#_Toc486249272)

[a) Plot the timeseries. 72](#_Toc486249273)

[b) Fit a Simple Exponential Smoothing model, predict 2 years (24 months) out and plot predictions. 73](#_Toc486249274)

[8.3 Fit a Holt-Winters model to a timeseries. 74](#_Toc486249275)

[a) Plot the timeseries. 74](#_Toc486249276)

[b) Fit a Holt-Winters additive model, predict 2 years (24 months) out and plot predictions. 75](#_Toc486249277)

[9 Model Evaluation & Selection 76](#_Toc486249278)

[9.1 Evaluate the accuracy of regression models. 76](#_Toc486249279)

[a) Evaluation on training data. 76](#_Toc486249280)

[b) Evaluation on testing data. 77](#_Toc486249281)

[9.2 Evaluate the accuracy of classification models. 78](#_Toc486249282)

[a) Evaluation on training data. 78](#_Toc486249283)

[b) Evaluation on testing data. 79](#_Toc486249284)

[9.3 Evaluation with cross validation. 79](#_Toc486249285)

[a) KFold 79](#_Toc486249286)

[b) ShuffleSplit 81](#_Toc486249287)

[10 Text Analytics 83](#_Toc486249288)

[11 Deep Learning 83](#_Toc486249289)

[Appendix 83](#_Toc486249290)

[1 Built-in SAS Data Types 83](#_Toc486249291)

[2 SAS Procedures 84](#_Toc486249292)

[3 SAS DATA step 85](#_Toc486249293)

[Alphabetical Index 85](#_Toc486249294)

[Data Frame 85](#_Toc486249295)

[Dictionary 85](#_Toc486249296)

[Series 86](#_Toc486249297)

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.

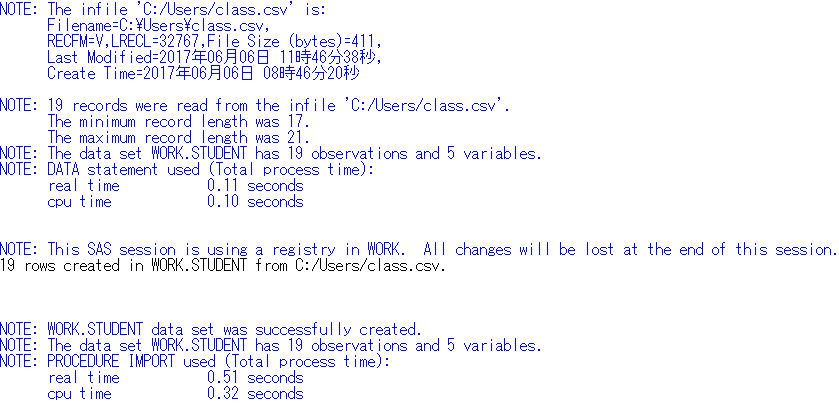
data student\_json;   
 INFILE 'C:/Users/class.json' LRECL = 3456677 TRUNCOVER SCANOVER   
 dsd   
 dlm=",}";   
 INPUT   
 @'"Name":' Name : $12.   
 @'"Sex":' Sex : $2.   
 @'"Age":' Age :   
 @'"Height":' Height :   
 @'"Weight":' Weight :   
 @@;   
run;

[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 14:00:08 Observation Length 32  
Last Modified 06/26/2017 14:00:08 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\\_TD6744\_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)

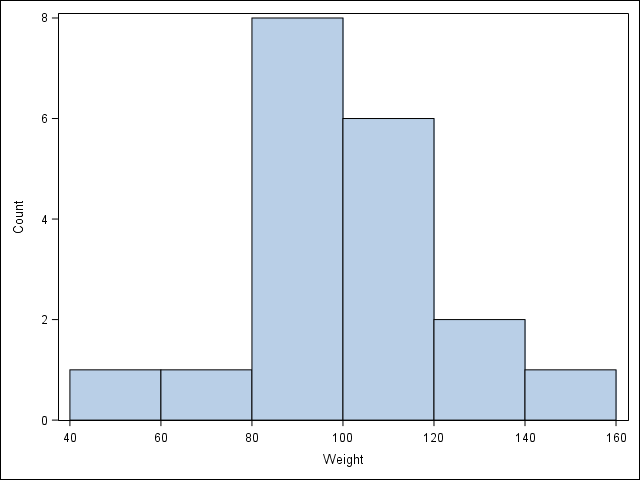
**TRY THIS AT HOME**: Run this procedure and do not reduce the output.

# 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.

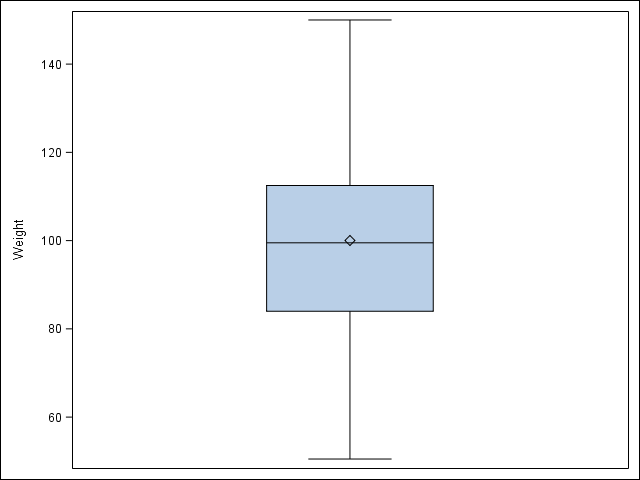
## 2.1 Visualize a single continuous variable by producing a histogram.

/\* 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: 

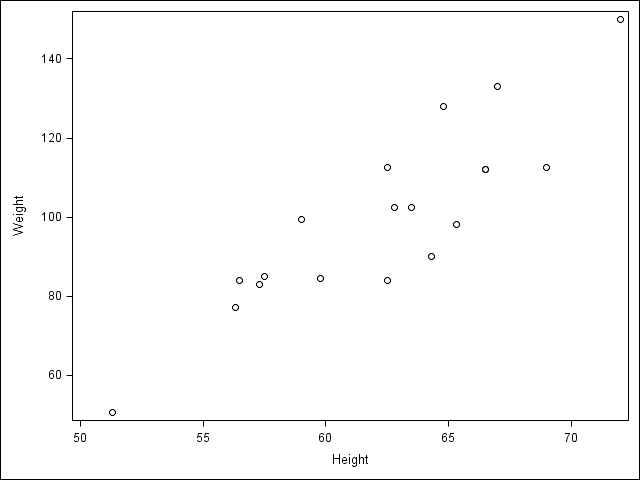
## 2.2 Visualize a single continuous variable by producing a boxplot.

/\* 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.

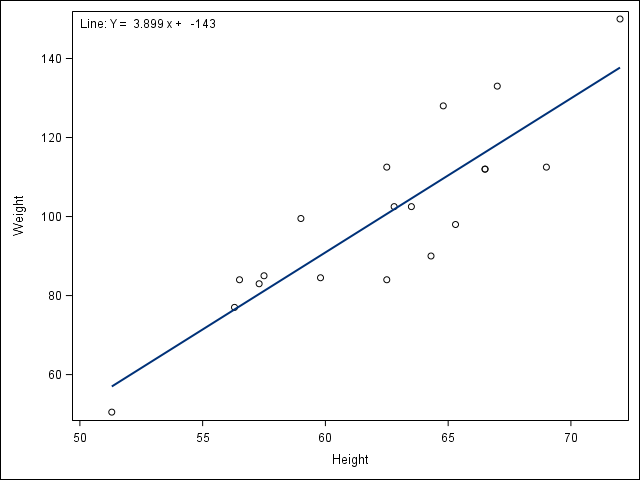
/\* Notice here you specify the y variable followed by the x variable \*/  
proc sgscatter data = student;   
 plot Weight \* Height;   
run;

Output: 

[SGSCATTER Procedure](#sgscatter)

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

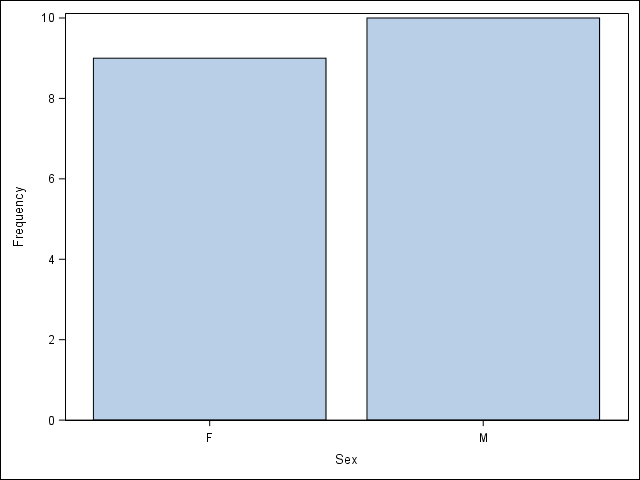
/\* Use proc reg to get the parameter estimates for the line of best fit,   
 but don't print the graph (ods graphics off) \*/  
ods graphics off;   
proc reg data = student;   
 /\* Syntax indicates Weight as a function of Height \*/  
 model Weight = Height;   
 ods output ParameterEstimates=PE;   
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;

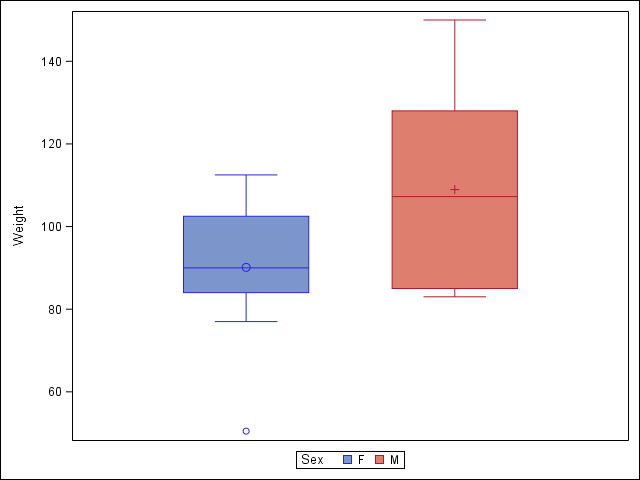
Output: 

[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 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  
 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 6  
 Number of Leaves Before Pruning 15  
 Number of Leaves After Pruning 9  
 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 4 151 0.0258  
  
 Model-Based Fit Statistics for Selected Tree  
   
 N Mis-  
 Leaves ASE class Sensitivity Specificity Entropy Gini RSS  
  
 9 0.0121 0.0126 0.9959 0.9742 0.0841 0.0242 9.6349  
  
 Model-Based Fit Statistics for Selected Tree  
   
 AUC  
  
 0.9881  
  
 Variable Importance  
   
 Training  
 Variable Relative Importance Count  
  
 \_23 1.0000 11.3559 2  
 \_27 0.4047 4.5962 1  
 \_1 0.3466 3.9356 2  
 \_6 0.2341 2.6581 1  
 \_8 0.1664 1.8898 1  
 \_0 0.1631 1.8516 1

[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 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  
 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 61  
  
 Number of Observations Read 354  
 Number of Observations Used 354  
   
   
   
 The HPSPLIT Procedure  
  
 Model-Based Fit Statistics for Selected Tree  
   
 N  
 Leaves ASE RSS  
  
 61 2.0825 737.2  
  
 Variable Importance  
   
 Training  
 Variable Relative Importance Count  
  
 \_5 1.0000 132.6 8  
 \_12 0.5997 79.5242 8  
 \_7 0.3952 52.4071 6  
 \_4 0.2640 35.0099 9  
 \_0 0.2273 30.1348 3  
 \_9 0.1569 20.8019 7  
 \_6 0.1108 14.6883 6  
 \_10 0.1064 14.1112 4  
 \_11 0.0797 10.5698 5  
 \_8 0.0679 8.9986 2  
 \_2 0.0476 6.3100 2

/\* 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  
 ------------  
 27.2078085  
 ------------

## 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 \*/  
ods select none;  
proc hp4score data = test seed = 29;  
 score file = 'hpbreastcancer2.bin' out = scored;  
run;  
ods select all;  
  
/\* 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 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 11.3683 43.2860  
 3 576 6.6232 31.3383  
 4 773 4.8837 24.8313  
 5 958 4.0583 21.5074  
 6 1155 3.7023 18.2075  
 7 1355 3.2854 20.0734  
 8 1551 2.8333 16.4209  
 9 1745 2.9038 17.4034  
 10 1942 2.9196 17.6812  
 11 2122 2.7632 16.6404  
 12 2313 2.6170 16.7454  
 13 2509 2.6299 16.6065  
 14 2706 2.5418 16.4241  
 15 2901 2.4397 15.5039  
 16 3091 2.2983 15.2623  
 17 3292 2.3907 15.3176  
 18 3478 2.2574 14.6200  
 19 3677 2.1693 14.2939  
 20 3873 2.2105 14.2776  
 21 4066 2.1068 14.0666  
 22 4261 2.1419 13.9742  
 23 4457 2.1239 14.1427  
 24 4641 2.0986 13.8644  
 25 4838 2.1718 13.9556  
 26 5042 2.2465 14.1226  
 27 5236 2.1678 13.7884  
 28 5415 2.2207 13.8642  
 29 5610 2.1705 13.6529  
 30 5800 2.1763 13.4050  
 31 5994 2.3066 13.7989  
 32 6176 2.3564 13.6090  
 33 6360 2.2905 13.4957  
 34 6553 2.2505 13.0359  
 35 6745 2.1593 12.4862  
 36 6941 2.1614 12.6334  
 37 7125 2.1541 12.5703  
 38 7319 2.1364 12.7408  
 39 7522 2.1109 12.6118  
 40 7712 2.1543 12.6576  
 41 7898 2.1385 12.6186  
 42 8080 2.1784 12.6728  
 43 8270 2.2877 12.9858  
 44 8463 2.2850 12.8440  
 45 8658 2.2836 12.8810  
 46 8857 2.2888 13.1012  
 47 9045 2.2843 13.0514  
 48 9233 2.3148 13.0745  
 49 9431 2.2789 13.0634  
 50 9623 2.2752 12.9776  
 51 9831 2.2498 12.9791  
 52 10026 2.2526 12.9777  
 53 10221 2.2672 12.9902  
 54 10408 2.2593 13.0558  
 55 10596 2.2957 13.2262  
 56 10788 2.2959 13.1870  
 57 10977 2.3256 13.2589  
 58 11173 2.3208 13.2695  
 59 11364 2.2901 13.1079  
 60 11552 2.2612 13.1308  
 61 11742 2.2491 13.0531  
 62 11938 2.2204 12.9735  
 63 12136 2.2213 13.0562  
 64 12333 2.2066 13.0283  
 65 12525 2.2162 13.0132  
 66 12718 2.2031 12.9627  
 67 12911 2.1974 13.0353  
 68 13108 2.2049 13.1707  
 69 13289 2.2056 13.0791  
 70 13484 2.1924 12.9408  
 71 13674 2.1686 12.8484  
 72 13866 2.2003 13.0394  
 73 14057 2.1860 13.0218  
 74 14242 2.1825 12.9853  
 75 14439 2.1844 12.9505  
 76 14621 2.1770 12.9093  
 77 14815 2.1590 12.9136  
 78 15000 2.1914 12.9766  
 79 15186 2.2442 13.1019  
 80 15376 2.2233 13.1474  
 81 15566 2.2257 13.1030  
 82 15762 2.2025 13.0779  
 83 15953 2.2114 13.0018  
 84 16145 2.2128 13.0747  
 85 16321 2.2574 13.2711  
 86 16510 2.2372 13.2143  
 87 16706 2.2161 13.1554  
 88 16890 2.1980 13.1078  
 89 17085 2.1905 12.9783  
 90 17284 2.1902 12.9538  
 91 17477 2.1743 12.9489  
 92 17668 2.1681 12.9888  
 93 17858 2.1504 12.9055  
 94 18052 2.1521 12.9346  
 95 18240 2.1390 12.8186  
 96 18429 2.1394 12.8283  
 97 18619 2.1219 12.6850  
 98 18821 2.1138 12.5864  
 99 19016 2.1106 12.5023  
 100 19210 2.1227 12.5373  
  
 Loss Reduction Variable Importance  
   
 Number OOB Absolute OOB Absolute  
 Variable of Rules MSE MSE Error Error  
  
 \_5 1607 25.96700 21.44497 1.674146 1.271060  
 \_12 4298 26.08398 21.20143 1.747167 1.062741  
 \_2 862 7.51862 4.32709 0.481668 0.225365  
 \_10 1008 4.43794 2.90609 0.311778 0.121390  
 \_4 1080 4.85033 2.54304 0.447890 0.205106  
 \_9 1323 2.91508 1.36073 0.295646 0.085797  
 \_0 410 2.31937 0.03880 0.191909 0.033169  
 \_1 162 0.10618 -0.03221 0.019840 -0.004146  
 \_8 781 0.72275 -0.26446 0.099445 -0.021106  
 \_3 183 0.54563 -0.30985 0.039416 -0.008242  
 \_7 2419 6.78473 -0.42270 0.617601 0.110761  
 \_11 3469 3.02608 -0.76432 0.470083 -0.028214  
 \_6 1508 1.95177 -0.90383 0.285909 0.000426

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

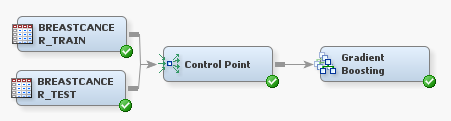
/\* Prediction on testing data \*/  
ods select none;  
proc hp4score data = test seed = 29;  
 score file = 'hpboston2.bin' out = scored;  
run;  
ods select all;  
   
/\* Compute the squared differences between predicted and target \*/  
data scored;  
 set scored;  
 sq\_error = (P\_Target - Target)\*\*2;  
run;  
   
/\* Compute the mean of the squared differences (mean squared error) as an   
 assessment of the model \*/  
proc means data = scored mean;

The MEANS Procedure  
  
 Analysis Variable : sq\_error   
   
 Mean  
 ------------  
 9.1427096  
 ------------

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

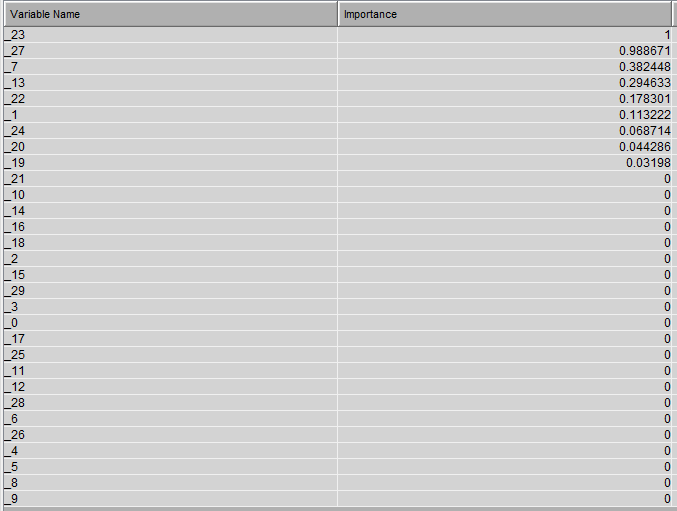
### a) Fit a gradient boosting classification model.

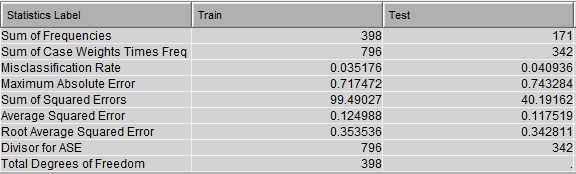
Currently, there is not a gradient boosting procedure available in Base SAS Therefore, the best method to create a gradient boosting model currently is using SAS Enterprise Miner. Create the following diagram in SAS Enterprise Miner:

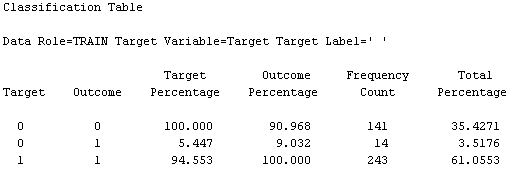
Output: 

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:

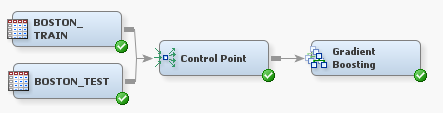
Output: 

Output: 

Output: 

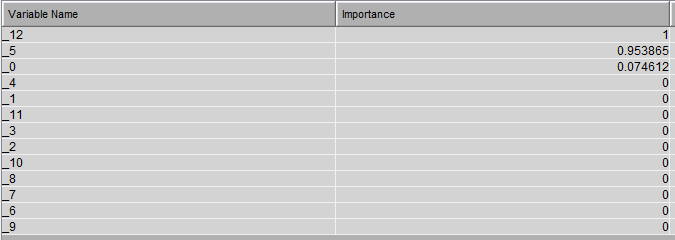
### b) Fit a gradient boosting regression model.

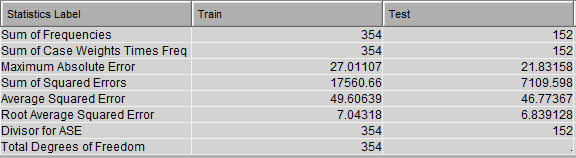
Again, there is not a gradient boosting procedure available in Base SAS, currently. Create the following diagram in SAS Enterprise Miner:

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.

/\* 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;  
run;

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.

/\* 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;  
run;

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;

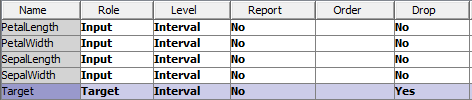
## 7.4 DBSCAN

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;

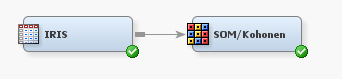
labels  
 0 1 2  
 Setosa 1 49 0  
 Versicolor 6 0 44  
 Virginica 10 0 40

## 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.

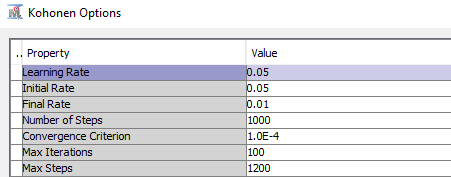
Output: 

Then create the following diagram in SAS Enterprise Miner:

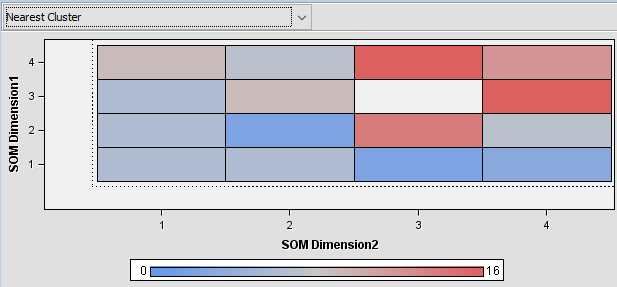
Output: 

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:

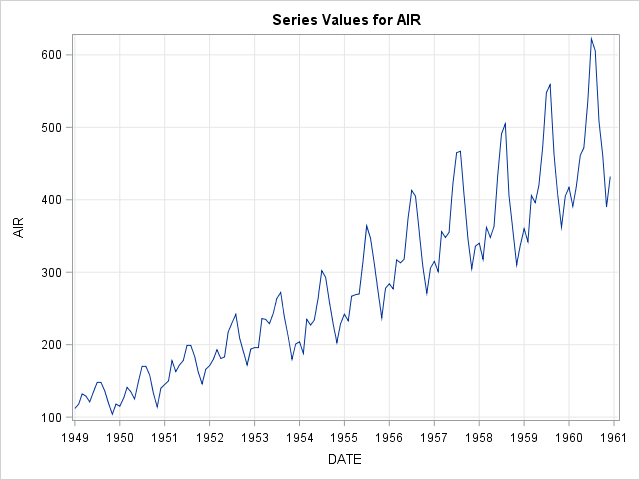
Output: 

# 8 Forecasting

## 8.1 Fit an ARIMA model to a timeseries.

### a) Plot the timeseries.

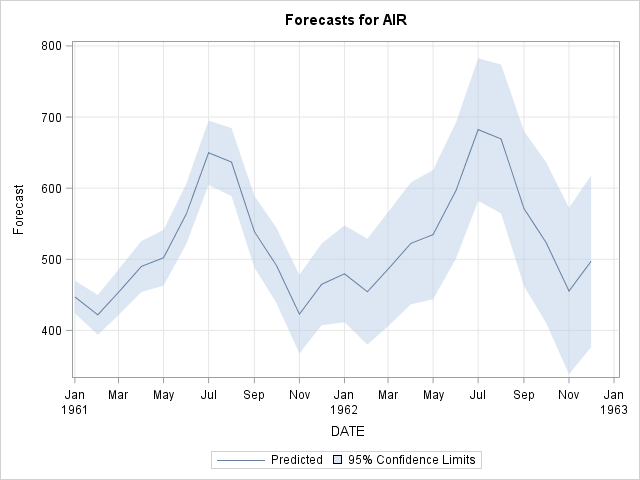
proc import out = air  
 datafile = 'C:/Users/air.csv'  
 dbms = csv replace;  
 getnames = yes;  
run;  
   
proc timeseries data = air plot = series;  
 id date interval = month;  
 var air;  
run;

Output: 

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

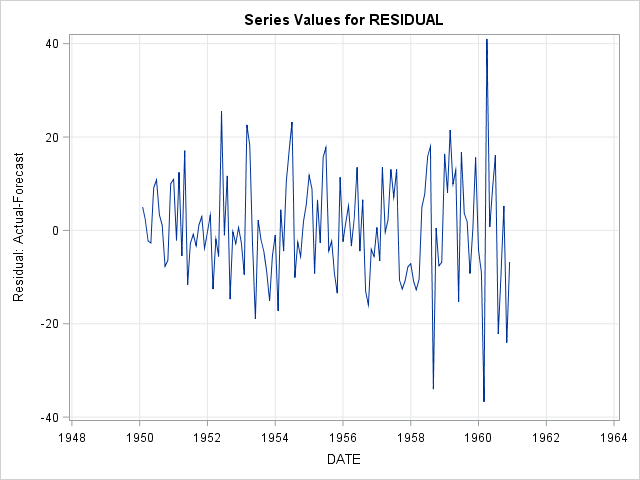
The output of this code has been limited for space reasons.

proc arima data = air;  
 identify var = air(1,12) noprint;  
 estimate q=(1)(12) noint method=ml noprint;  
 forecast id=date interval=month out=forecast;  
run;  
  
/\* SAS automatically predicts 2 years out and plots the predictions \*/

Output: 

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

proc timeseries data = forecast plot = series;  
 id date interval = month;  
 var residual;  
run;

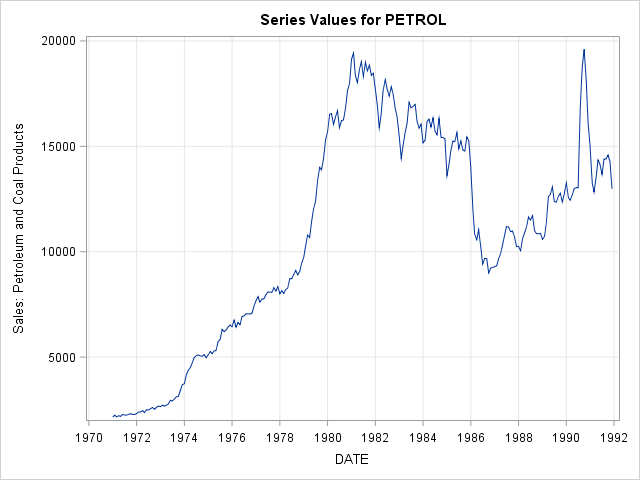
Output: 

## 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;

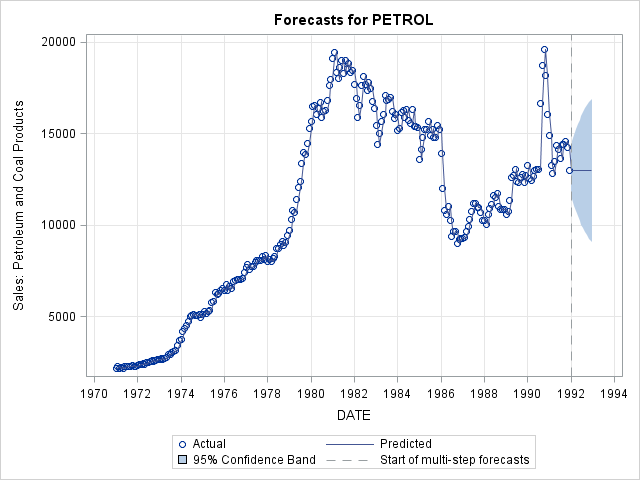
The TIMESERIES Procedure  
  
 Input Data Set  
  
 Name WORK.USECON  
 Label   
 Time ID Variable DATE  
 Time Interval MONTH  
 Length of Seasonal Cycle 12

Output: 

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

proc 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

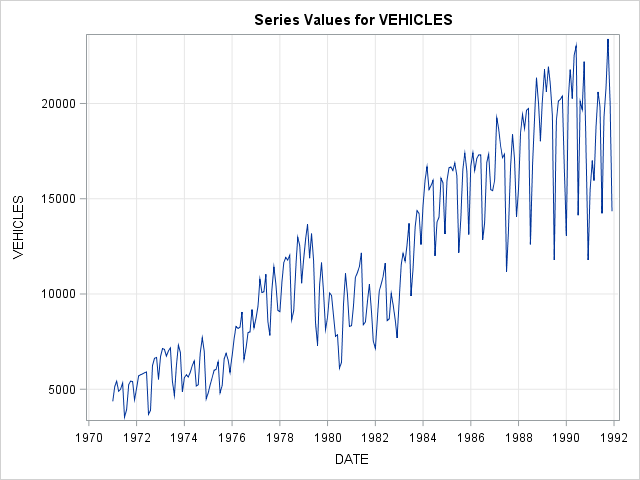
Output: 

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

### a) Plot the timeseries.

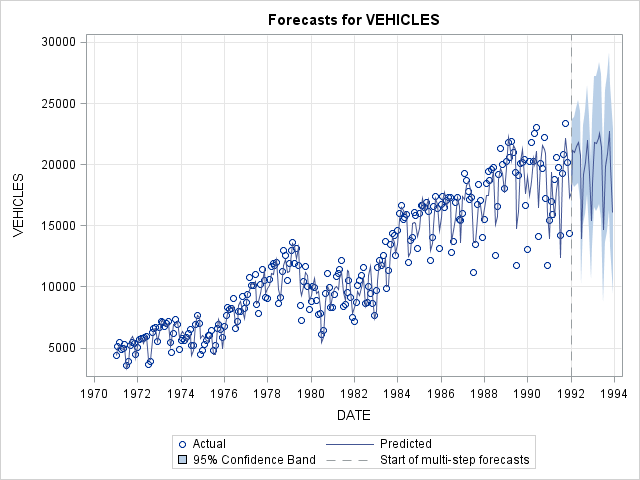
proc timeseries data = usecon plot = series;  
 id date interval = month;  
 var vehicles;  
run;

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 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 vehicles / model = addwinters;  
run;

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 \*/  
ods select none;  
proc hp4score data = train;  
 score file = 'rfMod.bin' out = scored\_train;  
run;  
ods select all;  
  
/\* 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;

r2\_rf  
  
 0.9743865

### b) Evaluation on testing data.

/\* Random Forest Regression Model (rfMod) \*/  
  
/\* Evaluation on testing data \*/  
ods select none;  
proc hp4score data = test;  
 score file = 'rfMod.bin' out = scored\_test;  
run;  
ods select all;  
  
/\* 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;

r2\_rf  
  
 0.8864763

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 \*/  
ods select none;  
proc hp4score data = train;  
 score file = 'rfMod.bin' out = scored;  
run;  
ods select all;  
  
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;

accuracy\_forest  
  
 1

### b) Evaluation on testing data.

/\* Random Forest Classification Model (rfMod) \*/  
  
/\* Evaluation on testing data \*/  
ods select none;  
proc hp4score data = test;  
 score file = 'rfMod.bin' out = scored;  
run;  
ods select all;  
  
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;

accuracy\_forest  
  
 0.9685185

## 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](http://support.sas.com/documentation/cdl/en/fedsqlref/67364/HTML/default/viewer.htm#n19bf2z7e9p646n0z224cokuj567.htm) The SAS implementation of a string as a fixed-length character string of length *n*.
* [DOUBLE](http://support.sas.com/documentation/cdl/en/fedsqlref/67364/HTML/default/viewer.htm#n19bf2z7e9p646n0z224cokuj567.htm) A decimal point number implemented as a 64-bit double precision, floating-point number.

## 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

## [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

## [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).