Paper ###-2008

Classification of Breast Cancer Cells Using JMP

Marie Gaudard, North Haven Group, Hernando, FL

ABSTRACT

This paper illustrates some of the features of JMP that support classification and data mining. We will utilize the Wisconsin Breast Cancer Diagnostic Data Set, a set of data used to classify breast lumps as malignant or benign based on the values of thirty potential predictors, obtained by measuring the nuclei of fluid removed using a fine needle aspirate. We begin by illustrating some visualization techniques that help build an understanding of the data set. After partitioning our data into a training set, a validation set, and a test set, we fit four models to the training data. These include a logistic model, a partition model, and two neural net models. We then compare the performance of these four models on the validation data set to choose one. The test set is used to assess the performance of this final model.

INTRODUCTION

The purpose of this paper is to illustrate a number of features of JMP that support classification and data mining. We acknowledge at the outset that JMP is not intended to be a complete data mining package. However, it contains a number of modeling techniques that can be profitably used in data mining. It is our goal to show how a classification model can be fit relatively efficiently using JMP capabilities.

To illustrate these techniques, we will use a published data set, the Wisconsin Breast Cancer Diagnostic data, which is described in detail in the next section. For now, we only mention that the response of interest is whether a tumor is malignant or benign, and we will attempt to classify into these two categories using 30 potential predictors. After we describe the Wisconsin study and the data set, we will show ways to visualize this data, and then we will fit four classification models using three techniques: logistic modeling, recursive partitioning, and neural nets. We conclude with a comparison of the four classification models, choosing the best one based on performance on a validation subset of our data set.

THE WISCONSIN BREAST CANCER DIAGNOSTIC DATA SET

The Wisconsin Breast Cancer Data Set arises in connection with diagnosing breast tumors based on a fine needle aspirate (Mangasarian, OL, et al, 1994). In this study, a small-gauge needle is used to remove fluid directly from the lump or mass. The fluid is placed on a glass slide and is stained, so as to reveal the nuclei of the cells. A software program is used to determine the boundaries of the nuclei. A typical image consists of 10 to 40 nuclei. The software computes ten characteristics for each nucleus: radius, perimeter, area, texture, smoothness, compactness, number of concave regions and size of concavities (a ***concavity*** is an indentation in the cell nucleus), symmetry, and fractal dimension of the boundary (a higher value means a less regular contour). (For more detail on these characteristics, see Street, WN, et al, 1993.)

A set of 569 images was processed as described above. Since a typical image can contain from 10 to 40 nuclei, the data were summarized. For each variable, the mean, max, and standard error of the mean, were computed. These are the 30 variables in our data set. The model developed by the researchers utilized a linear programming method that identified separating hyperplanes. Using all 569 observations, a classification accuracy of 97.3% was achieved. Even more remarkably, the next 131 cases that were analyzed were classified with 100% accuracy. Their final model utilized only three variables: Mean Texture, Max Area, and Max Smoothness.

We will utilize this data set in illustrating some of JMP's capabilities in the area of classification analysis. The data set can be downloaded from http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic). Alternatively, JMP versions of this data set can be obtained at the link below or on request by writing to the contact email address at the end of this paper.

Breast Cancer Data Files:

`BreastCancerClassification_Raw.jmp` – This consists of the raw data and only two columns defined by us, namely the `Data Set Indicator` column (to ensure the same analysis data sets) and the `Random Unif` column on which the previous column is based. The user can utilize this data file to recreate, from scratch, the analyses in this paper.

`BreastCancerClassification_Scripts.jmp` – This data file contains scripts for most analyses, but the user must run these, save columns, and add formula columns to complete the analyses.

`BreastCancerClassification.jmp` – This data file contains the scripts that are in the file above as well as all of the added columns.

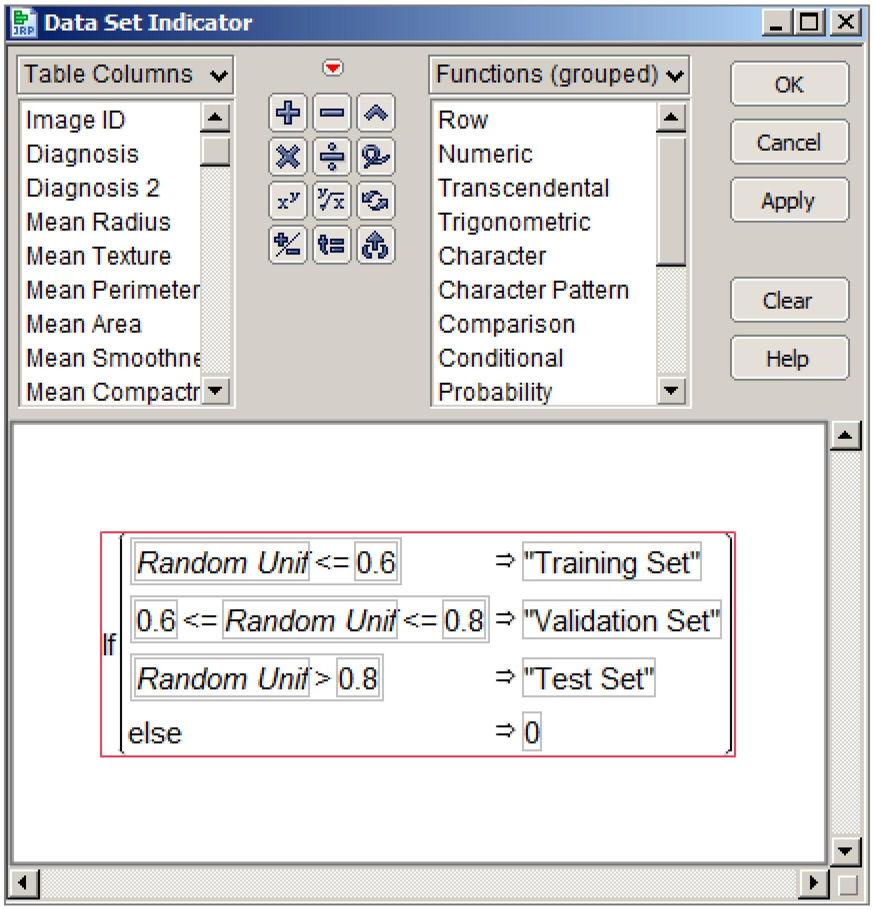
TRAINING, VALIDATION, AND TEST SETS

Our plan will be to construct models using three approaches: logistic regression, recursive partitioning, and neural nets. These will be constructed using a portion of our data called the ***training set***. Since bias (resulting from a tendency to overfit) may be built into these models by virtue of being fit using the training data, we will then assess these models relative to their performance on a hold-out portion of our data, an independent data set called the ***validation set***. The best model will be chosen, and, again because choosing a model based on the validation set can lead to overfitting, we will confirm the performance of the chosen model on another independent data set, called the ***test set***. (See Bishop, CM, 1995, and Ripley, BD, 1996.)

Once again, if you wish to recreate our analysis on your own, use the data table `BreastCancerClassifiction_Raw.jmp.` If you want to do part of the work on your own, use `BreastCancerClassification_Scripts.jmp .` Otherwise, you will find all of the following work included in our working data table, `BreastCancerClassifiction.jmp.`

We begin by splitting our data set of 569 rows into three portions: a training set consisting of about 60% of the data, a validation set consisting of about 20% of the data, and a test set consisting of the remaining 20% of the data. We accomplish this in JMP by defining a new column, called `Random Unif`, containing the formula `Random Uniform().` We then define another new column, called `Data Set Indicator`, using a formula (shown in Figure 1) that assigns rows to one of the three data sets based on the value of the random uniform value assigned in the column `Random Unif`. Note that we have hidden and excluded the column `Random Unif`, since it is not needed in what follows. To `Hide` and `Exclude` a column, right-click on the column name in the `Columns` panel, and choose `Hide` and `Exclude`. Two little icons will appear to the right of the column name to indicate that it is hidden and excluded.

Figure 1. Formula Defining Assignment to Data Sets



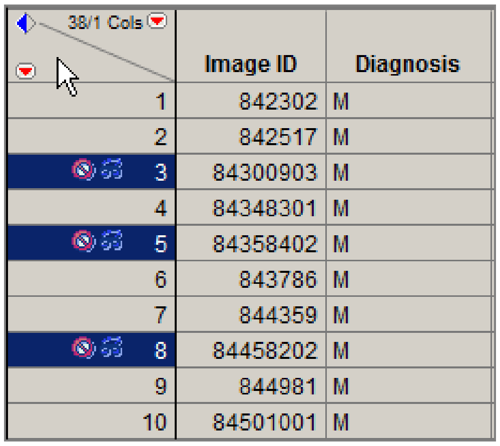
```
If | Random Unif <= 0.6               ⇒ "Training Set"
   | 0.6 <= Random Unif <= 0.8        ⇒ "Validation Set"
   | Random Unif > 0.8                ⇒ "Test Set"
   | else                             ⇒ 0
```

Now we need to make these three data set assignments convenient for use as we explore the data and fit models. We do this through the use of ***row state variables***. A row state variable is a column whose contents are ***row states***, namely attributes, such as `Exclude`, `Hide`, and `Select`, that are applied to certain rows.

To create a row state variable, we start by defining a new column. In the `Column Information` window, under `Data Type`, choose `Row State`. Then click `OK`. This defines the new column as a row state column. We have defined three new columns in this fashion, and they are called `Training Set`, `Validation Set`, and `Test Set`.

We would like the column `Training Set`, when applied to the data table, to exclude and hide all rows that are not part of the training set. To accomplish this, we first apply this configuration to the data table, namely, we exclude and hide all rows that are not part of the training set. We go to `Rows > Row Selection > Select Where`, and select all rows where `Training Set Indicator does not equal Training Set`. We then choose `Rows > Exclude` and `Rows > Hide`. This inserts `Exclude` and `Hide` icons next to the selected rows. Finally, deselect the rows by left-clicking in the lower left triangular region in the upper left of the data grid, as shown in Figure 2. This deselects the rows, but retains the `Exclude` and `Hide` row states.

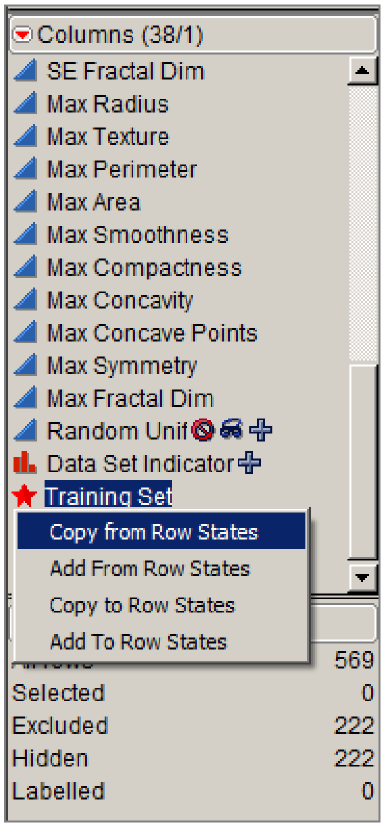
Figure 2. Lower Triangular Region of the Data Grid Corresponding to Rows



38/1 Cols	Image ID	Diagnosis
1	842302	M
2	842517	M
3	84300903	M
4	84348301	M
5	84358402	M
6	843786	M
7	844359	M
8	84458202	M
9	844981	M
10	84501001	M

Finally, to insert these row states as values into the row state variable `Training Set`, one goes to the `Columns` panel in the data table window, shown in Figure 3. Click on the star next to the row state variable, and choose `Copy from Row States`. We define three columns using this procedure: `Training Set`, `Validation Set`, and `Test Set`.

Figure 3. Columns Panel Showing Choice for Copying Row States into the Column Training Set



Columns (38/1)
- SE Fractal Dim
- Max Radius
- Max Texture
- Max Perimeter
- Max Area
- Max Smoothness
- Max Compactness
- Max Concavity
- Max Concave Points
- Max Symmetry
- Max Fractal Dim
- Random Unif
- Data Set Indicator
- Training Set

All rows	569
Selected	0
Excluded	222
Hidden	222
Labelled	0

To further enhance our understanding of the data, it will be useful to color and/or mark the points in our plots according to the diagnosis. We will want to add this information to the row state variables that we have just created. Begin by going to `Rows > Clear Row States`. We will add colors and markers, and this will ensure that these are the only row states available at this time. Once row states are cleared, go to `Rows > Color or Mark by Column`, and click on `Diagnosis`. If you are following along on a computer, you can simply click `Make Legend with Window`, and JMP will color the points red and blue, based on whether the tumor is malignant or benign. For the purposes of this paper, which prints in gray-scale, we also choose `Standard Markers`. The markers for malignant masses are circles and the markers for benign masses are plus signs. When we click `OK`, the markers appear in the row margins. Now, for each row state variable in turn, we click the star to the left of that variable in the `Columns` panel and we choose `Add From Row States`. (Do this with care – it is easy to make mistakes!)

Just in case our colors and markers disappear, we create another row state column called `Diagnosis Colors`. We `Copy from Row States` so that this new variable contains the colors (and markers) that we have just defined. If you are creating these columns on your own, now is a good time to save your work!

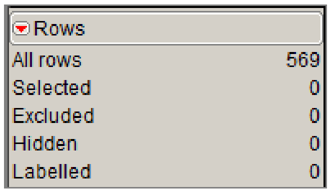
To apply these row states as we need them, we will now simply be able to click on the star corresponding to the appropriate data set column, and choose `Copy to Row States`.

DATA VISUALIZATION

ONE VARIABLE AT A TIME

At this point, no rows should be Excluded or Hidden. This can be verified by checking the `Rows` panel, shown in Figure 4, and noting that no rows are `Excluded` or `Hidden`. If there are excluded or hidden rows, go to `Rows > Clear Row States`. This removes all row states. Then go to the row state variable `Diagnosis Colors` in the `Columns` panel, and select `Copy to Row States` to reinstate the colors.

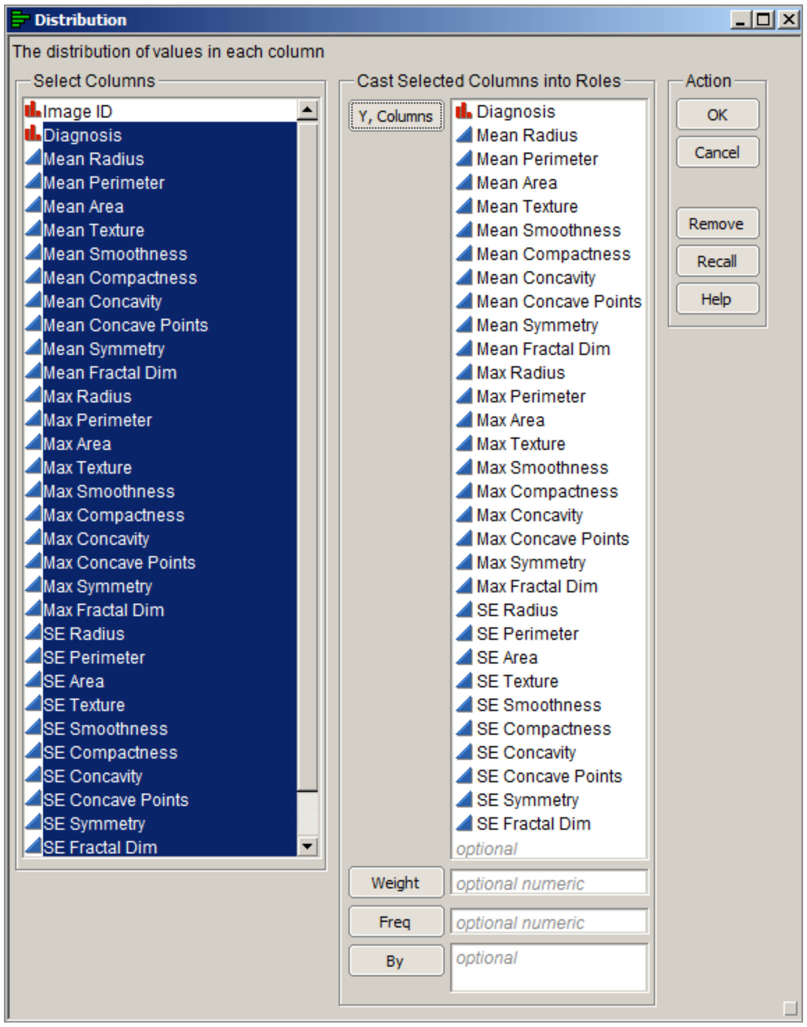
Figure 4. Rows Panel



Rows	
All rows	569
Selected	0
Excluded	0
Hidden	0
Labelled	0

We want to see distribution reports for all of our variables. To do this, go to `Analyze > Distribution` and populate the launch window as shown in Figure 5.

Figure 5. Launch Window for Distribution Platform

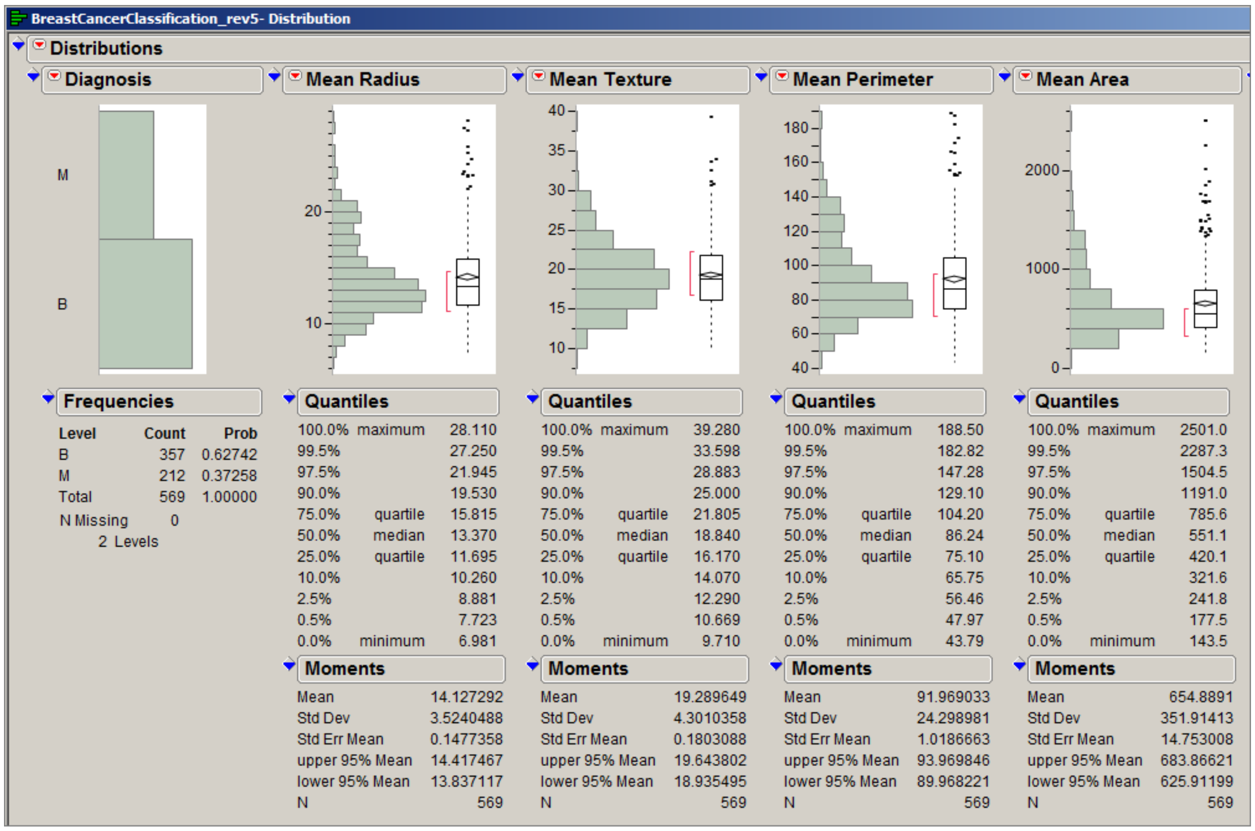


Select Columns: Image ID, Diagnosis, Mean Radius, Mean Perimeter, Mean Area, Mean Texture, Mean Smoothness, Mean Compactness, Mean Concavity, Mean Concave Points, Mean Symmetry, Mean Fractal Dim, Max Radius, Max Perimeter, Max Area, Max Texture, Max Smoothness, Max Compactness, Max Concavity, Max Concave Points, Max Symmetry, Max Fractal Dim, SE Radius, SE Perimeter, SE Area, SE Texture, SE Smoothness, SE Compactness, SE Concavity, SE Concave Points, SE Symmetry, SE Fractal Dim

Y, Columns: Diagnosis, Mean Radius, Mean Perimeter, Mean Area, Mean Texture, Mean Smoothness, Mean Compactness, Mean Concavity, Mean Concave Points, Mean Symmetry, Mean Fractal Dim, Max Radius, Max Perimeter, Max Area, Max Texture, Max Smoothness, Max Compactness, Max Concavity, Max Concave Points, Max Symmetry, Max Fractal Dim, SE Radius, SE Perimeter, SE Area, SE Texture, SE Smoothness, SE Compactness, SE Concavity, SE Concave Points, SE Symmetry, SE Fractal Dim

Clicking `OK` results in 31 distribution reports, the first five of which are shown in Figure 6. The vertical layout for the graphs is the JMP default. However, this can be changed under `File > Preferences`. The bar graph corresponding to `Diagnosis` indicates that 212, or 37.258%, of the tumors included in the study were malignant. Scrolling through the plots for the 30 predictors, one can assess the shape of the distributions and the presence of outliers. We note that most distributions are skewed to the right and that there may be some outliers (for example, there may be two outliers for `SE Concavity`). One can also determine, by looking at `N` under `Moments`, that there are no missing values for any of the variables.

Figure 6. First Five of 31 Distribution Reports



Frequencies

Level	Count	Prob
B	357	0.62742
M	212	0.37258
Total	569	1.00000
N Missing	0	
2 Levels		

Quantiles

		Mean Radius	Mean Texture	Mean Perimeter	Mean Area
100.0%	maximum	28.110	39.280	188.50	2501.0
99.5%		27.250	33.598	182.82	2287.3
97.5%		21.945	28.883	147.28	1504.5
90.0%		19.530	25.000	129.10	1191.0
75.0%	quartile	15.815	21.805	104.20	785.6
50.0%	median	13.370	18.840	86.24	551.1
25.0%	quartile	11.695	16.170	75.10	420.1
10.0%		10.260	14.070	65.75	321.6
2.5%		8.881	12.290	56.46	241.8
0.5%		7.723	10.669	47.97	177.5
0.0%	minimum	6.981	9.710	43.79	143.5

Moments

	Mean Radius	Mean Texture	Mean Perimeter	Mean Area
Mean	14.127292	19.289649	91.969033	654.8891
Std Dev	3.5240488	4.3010358	24.298981	351.91413
Std Err Mean	0.1477358	0.1803088	1.0186663	14.753008
upper 95% Mean	14.417467	19.643802	93.969846	683.86621
lower 95% Mean	13.837117	18.935495	89.968221	625.91199
N	569	569	569	569

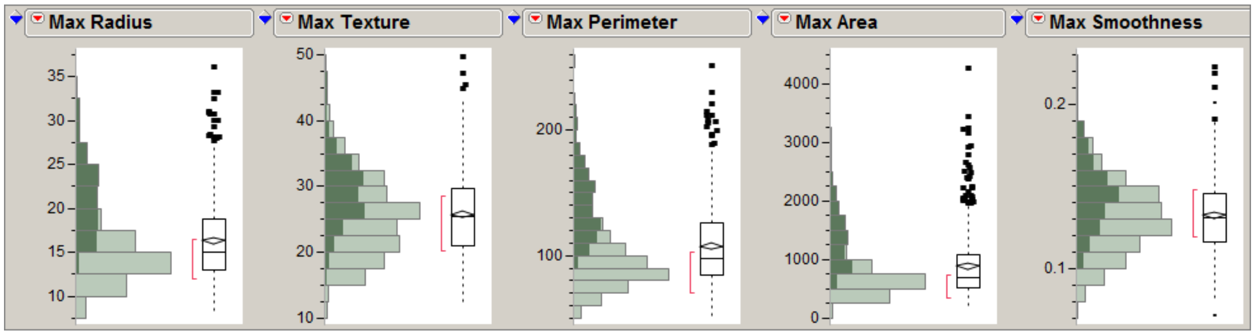
The script that generates the analysis in Figure 6 is saved to the data table as `Distribution – 31 Variables`. It can be found in the `Table` panel at the upper left of the data table. To run a script, simply click on the red triangle and choose `Run Script`.

TWO VARIABLES AT A TIME

In this section, we consider the issue of bivariate relationships among the variables. Of special interest is whether the predictors are useful in predicting `Diagnosis`.

For initial insight on this issue, we can utilize our `Distribution` output. Go to the bar graph for `Diagnosis`, and click on the bar corresponding to M. This has the effect of selecting all rows in the table for which `Diagnosis` has the value M. These rows are dynamically linked to all open plots, and so, in the 30 histograms corresponding to predictors, areas that correspond to the rows where `Diagnosis` is M are highlighted. We show five of the histograms corresponding to Max values in Figure 7. Note that masses that are malignant tend to have high values for these five variables. By scrolling through, one can detect relationships with most of the other variables as well. For example, malignant masses tend to have smaller `SE Texture` values than do benign masses. One can click on the bar for `Diagnosis` equal to B for additional insight.

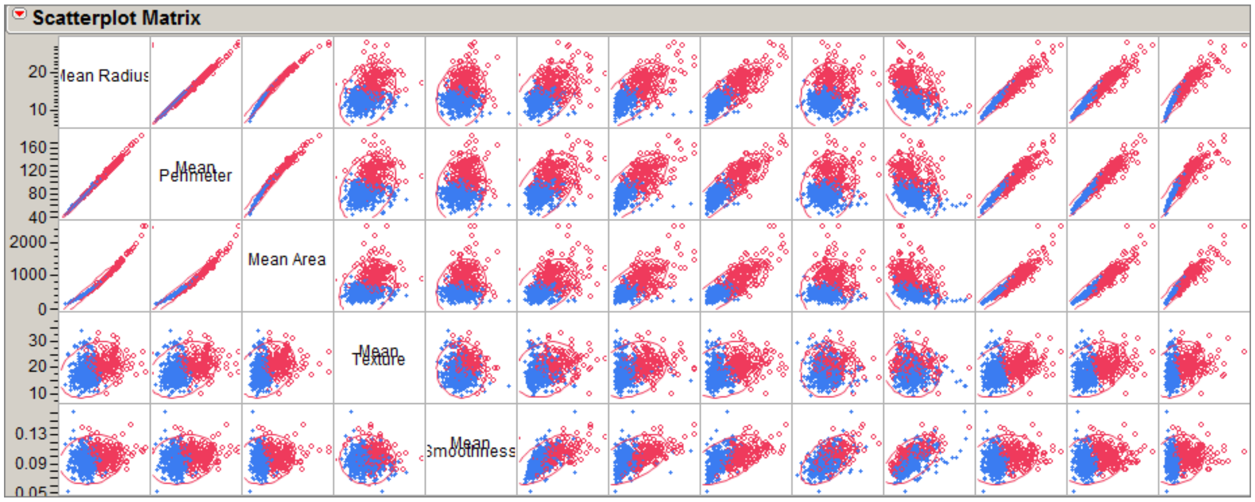
Figure 7. Five Histograms for Max Variables, with Areas for Malignant Diagnoses Highlighted



We are also interested in how the 30 predictors relate to each other. To see bivariate relationships among these 30 predictors, we will look at correlations and scatterplots. Go to `Analyze > Multivariate Methods > Multivariate`. In the launch window, enter all 30 predictors, from `Mean Radius` to `Max Fractal Dimension`, as `Y, Columns`. Clicking `OK` results in a report showing a correlation matrix. Go to the red triangle and choose `Scatterplot Matrix` from the drop-down menu. This gives a 30 by 30 matrix showing all bivariate scatterplots for our 30 predictors. The script that generates this output is saved to the data table as `Scatterplots and Correlations`.

We can think of radius, perimeter, and area as variables that describe the size of the nuclei. From the 3 by 3 square at the top left of the scatterplot matrix (see Figure 8), we see that `Mean Radius` and `Mean Perimeter` are highly correlated (see `Correlations` panel, which gives r = 0.9979), not an unexpected result. We also see that `Mean Area` is highly correlated with both `Mean Radius` and `Mean Perimeter`, with an expected quadratic relationship.

Figure 8. Portion of 30 by 30 Scatterplot Matrix



We note that the Max size variables are also highly correlated among themselves (all correlations greater than 0.9776), as are the SEs of the size variables (all correlations greater than 0.9377). These last details are easy to see if one clicks the red triangle next to `Multivariate`, and asks for `Pairwise Correlations`. (If you have run the script, this panel is already open.) Once in the `Pairwise Correlations` panel of the report, right-click, choose `Sort by Column`, and sort by `Correlation`.

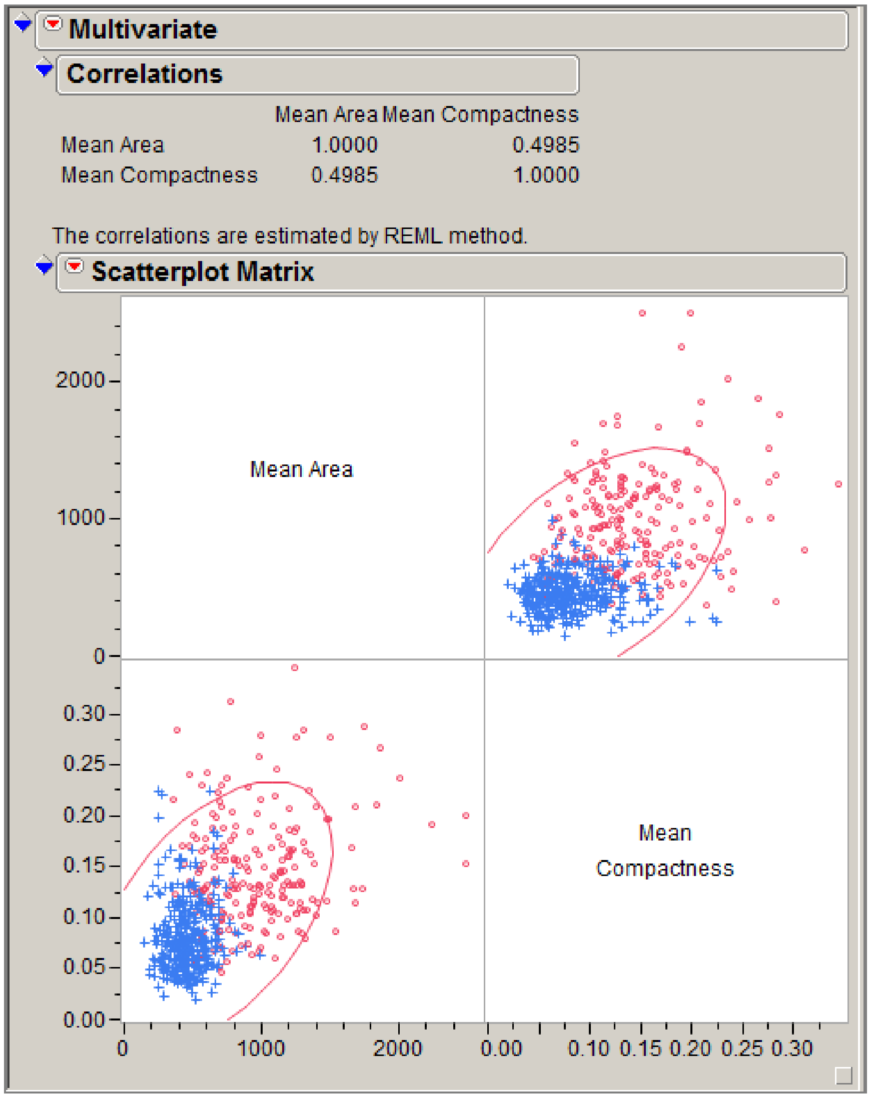
It is also of interest to note that the Max size variables are fairly highly correlated with the Mean size variables. Again, this is not an unexpected result.

The effects of multicollinearity for explanatory models are well-known. For predictive models, such are the ones that we will be constructing, the impact of multicollinearity is minimal, so long as future observations come from within the range of the multicollinear relationships. In other words, if multicollinear relationships are present, we can interpolate, but we should not extrapolate.

Further examination of the scatterplot matrix suggests that there may be a few bivariate outliers. We choose not to pursue these at this point. However, one might want to consider their inclusion in model-building since they have the potential to be influential.

Incidentally, this is a nice opportunity to remind the reader of one of the shortfalls of interpreting correlations of grouped data. Consider the two variables `Mean Area` and `Mean Compactness`. From the correlation matrix, we see that their correlation is 0.4985, which seems somewhat substantial. To see the relationship between these two variables more clearly, we go to `Analyze > Multivariate Methods > Mutlivariate`, and enter only `Mean Area` and `Mean Compactness` as `Y, Columns`. Clicking `OK` gives the output in Figure 9. There does appear to be some correlation.

Figure 9. Correlation and Scatterplot Matrix for Mean Area and Mean Compactness

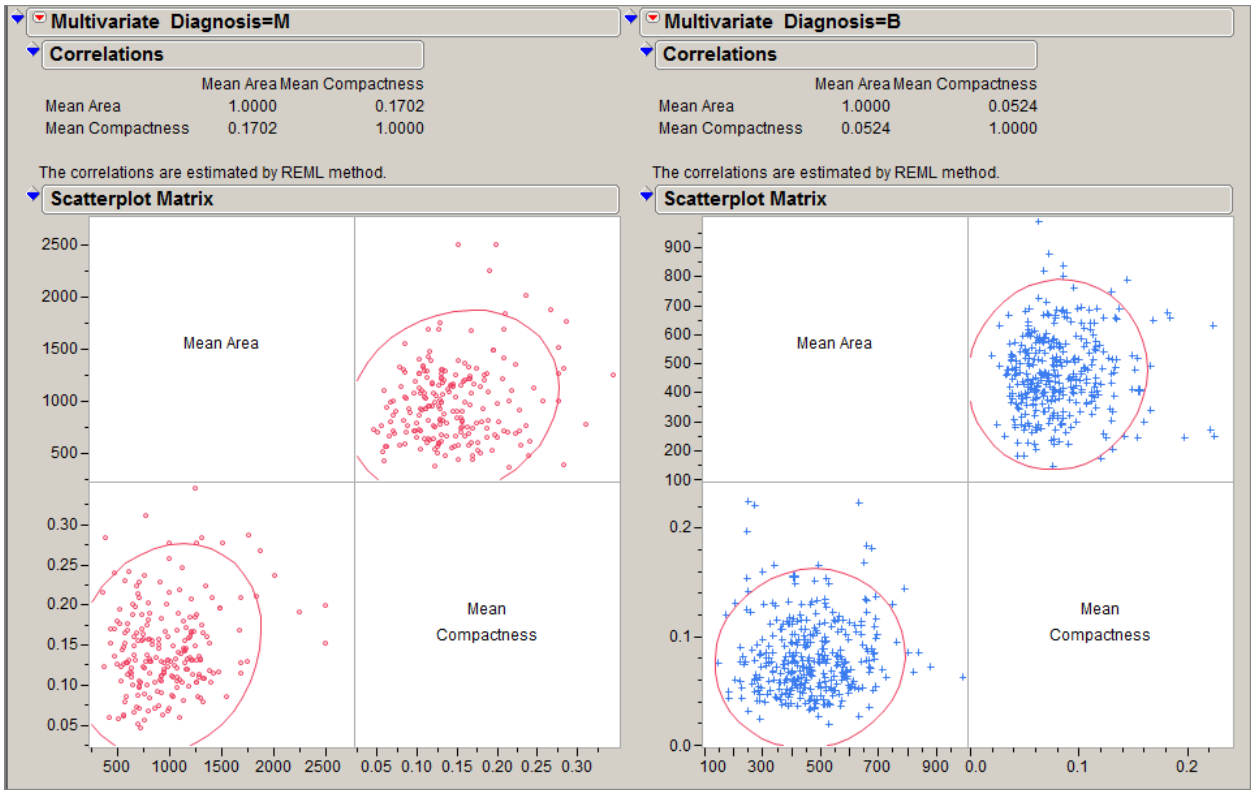


	Mean Area	Mean Compactness
Mean Area	1.0000	0.4985
Mean Compactness	0.4985	1.0000

The correlations are estimated by REML method.

Now, we go back `to Analyze > Multivariate Methods > Mutlivariate`, click on `Recall` to repopulate the menu with the previous entries, and add `Diagnosis` as a `By` variable. The resulting output, shown in Figure 10, shows very little correlation between these two variables, based on `Diagnosis` grouping. The apparent correlation when the data are aggregated is a function of how the two Diagnosis groups differ relative to the magnitudes of the two predictors.

Figure 10. Correlations and Scatterplots for Mean Area and Mean Compactness by Diagnosis



Multivariate Diagnosis=M

Correlations

```
                   Mean Area  Mean Compactness
Mean Area            1.0000            0.1702
Mean Compactness     0.1702            1.0000
```

The correlations are estimated by REML method.

Scatterplot Matrix

Multivariate Diagnosis=B

Correlations

```
                   Mean Area  Mean Compactness
Mean Area            1.0000            0.0524
Mean Compactness     0.0524            1.0000
```

The correlations are estimated by REML method.

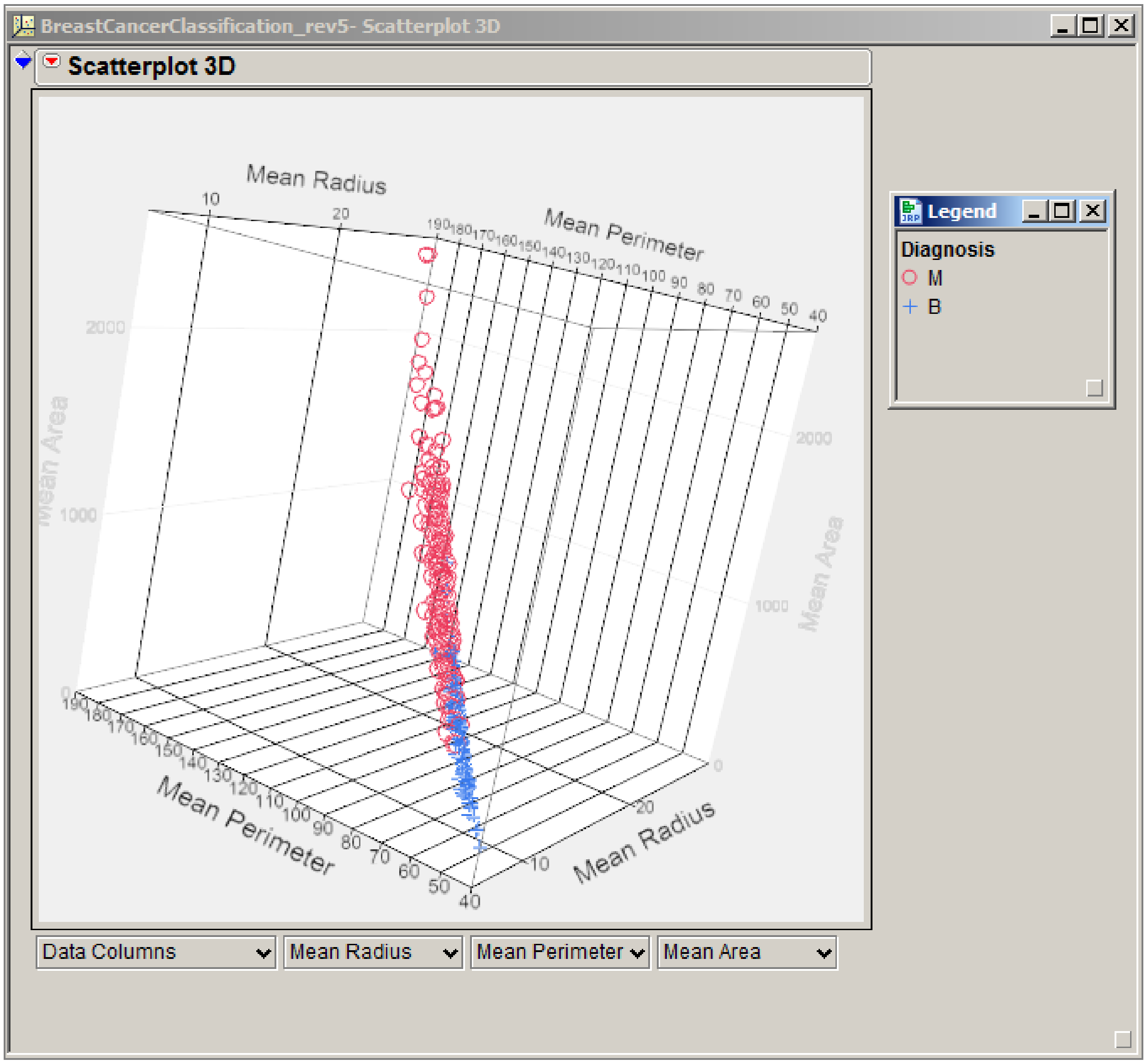
Scatterplot Matrix

MORE THAN TWO VARIABLES AT A TIME

JMP provides several ways to visualize high-dimensional data. We will only look at a few of these.

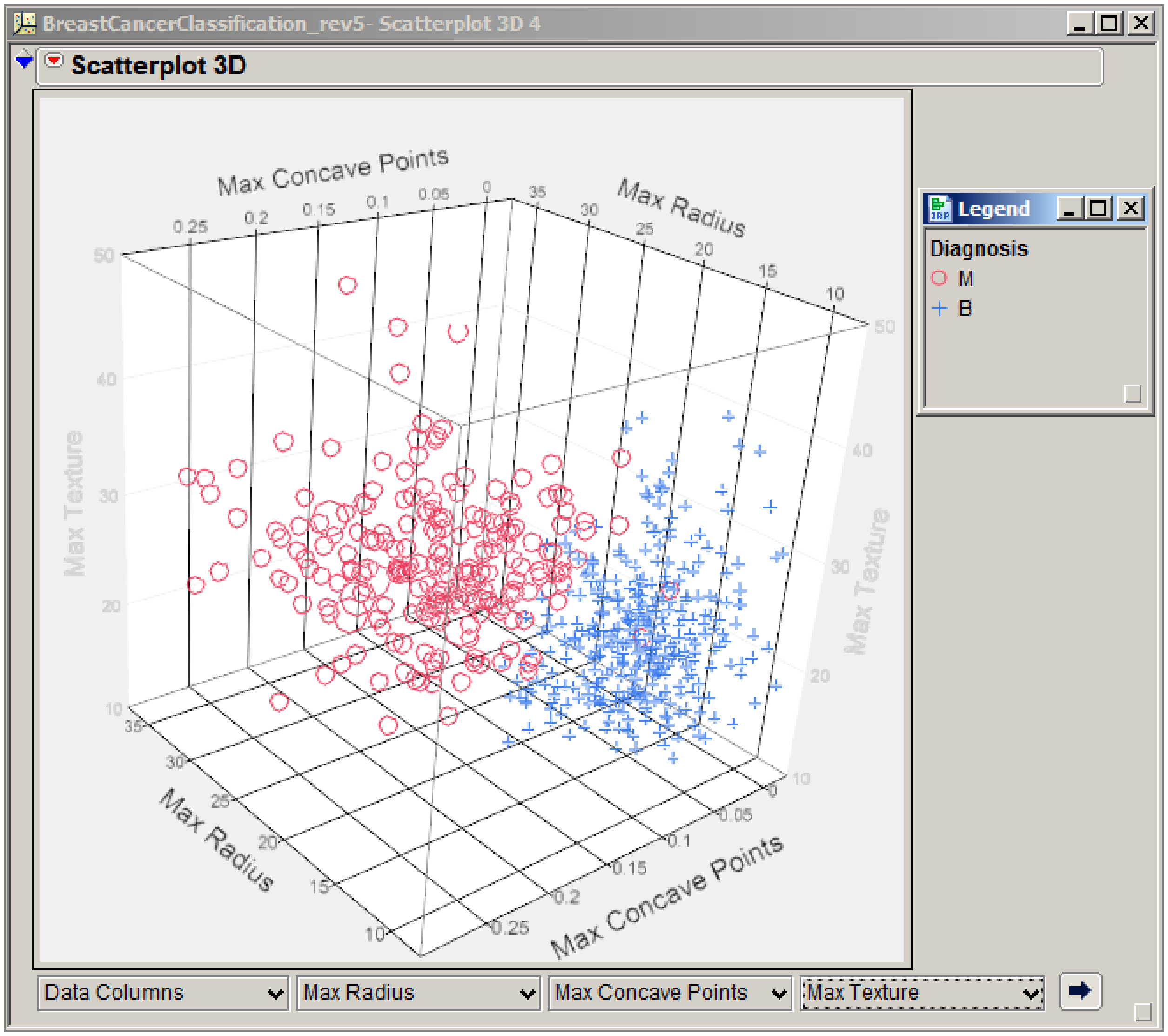
To see the three-dimensional relationship among the three size variables, `Mean Radius`, `Mean Perimeter`, and `Mean Area`, we will go to `Graph > Scatterplot 3D`. Enter the three size variables as `Y, Columns`, and click `OK`. The resulting plot is a rotatable 3D plot of the data, shown in Figure 11. The script generating this plot is called `Scatterplot 3D`. The dependencies of the three variables are striking. (Note that the markers and colors and the `Legend` window must be previously constructed through the `Rows` menu – this is not captured in the `Scatterplot 3D` script.)

Figure 11. Three Dimensional View of Mean Size Variables, by Diagnosis



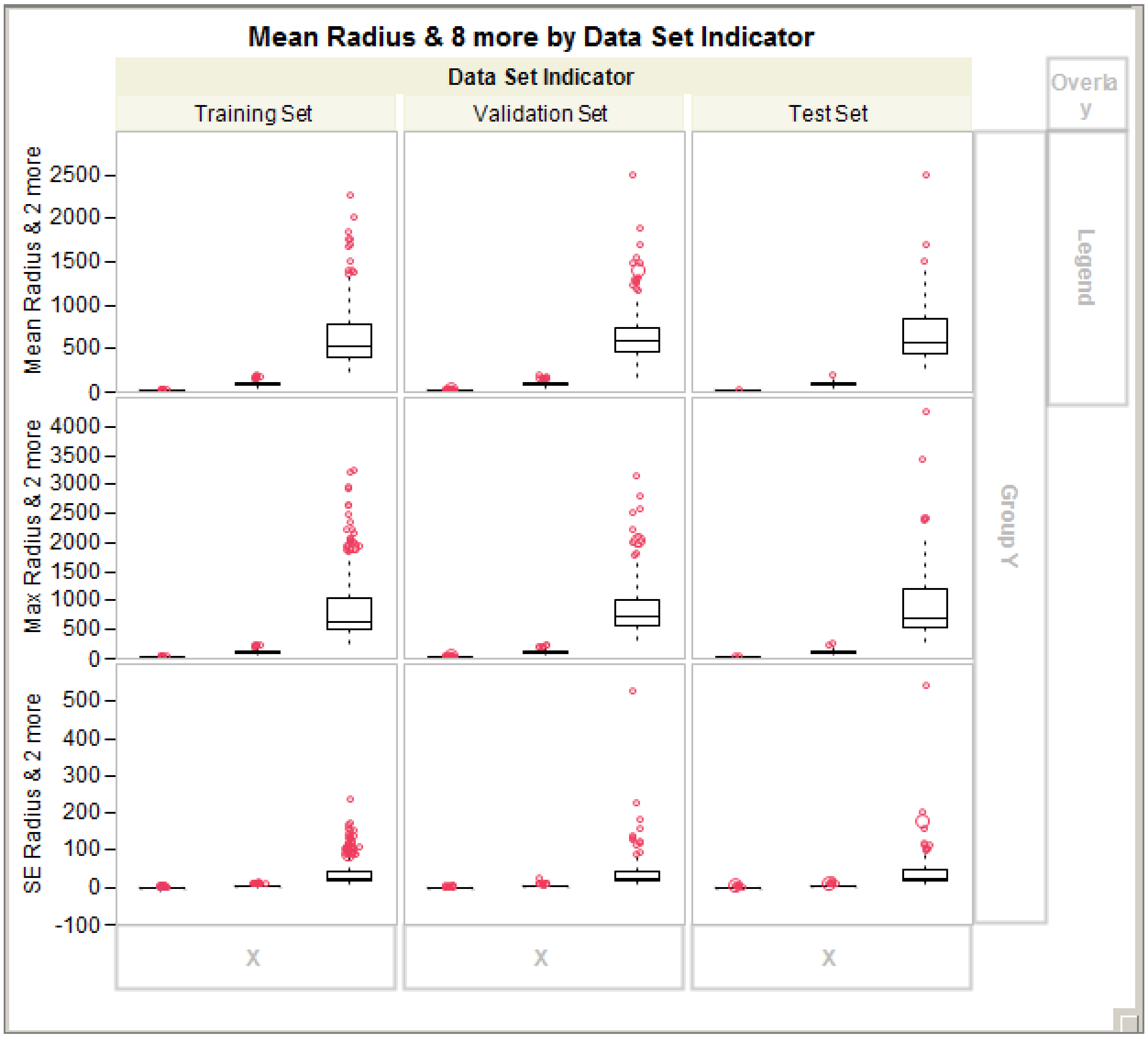
It is possible to enter all 30 predictor variables as `Y, Columns` in the Scatterplot 3D launch window. When that is done, JMP allows the user to select three variables from the drop-down menus at the bottom of the plot to be used for the axes. Figure 12 displays the 3-D scatterplot for three of the Max variables. Note that the two diagnosis classes have a reasonable degree of separation in this three-dimensional space. The script that generates this report is called `Scatterplot 3D 2`, and is saved to the data table.

Figure 12. Scatterplot 3D for Max Radius, Max Concave Points, and Max Texture, by Diagnosis



We might be interested in confirming that our training, validation, and test sets do not differ greatly in terms of the distributions of the predictors. JMP's `Graph Builder` (new in JMP 8), which is found under `Graph`, provides an intuitive interface for comparing the distributions of multiple variables in a variety of ways. The graph in Figure 13 utilizes boxplots to compare the distributions of the size variables (`Radius`, `Perimeter`, and `Area`), for the `Mean`, `Max`, and `SE` groupings, across the training, validation, and test sets. (The script that creates this output is saved as `Graph Builder – Radius, Perimeter, Area`.) We conclude that the distributions across the analysis sets are quite similar, with the exception of two outliers for `SE Area` that appear in the Validation and Test sets. We also note that the shapes of the distributions for the `Mean` and `Max` size values are similar as well.

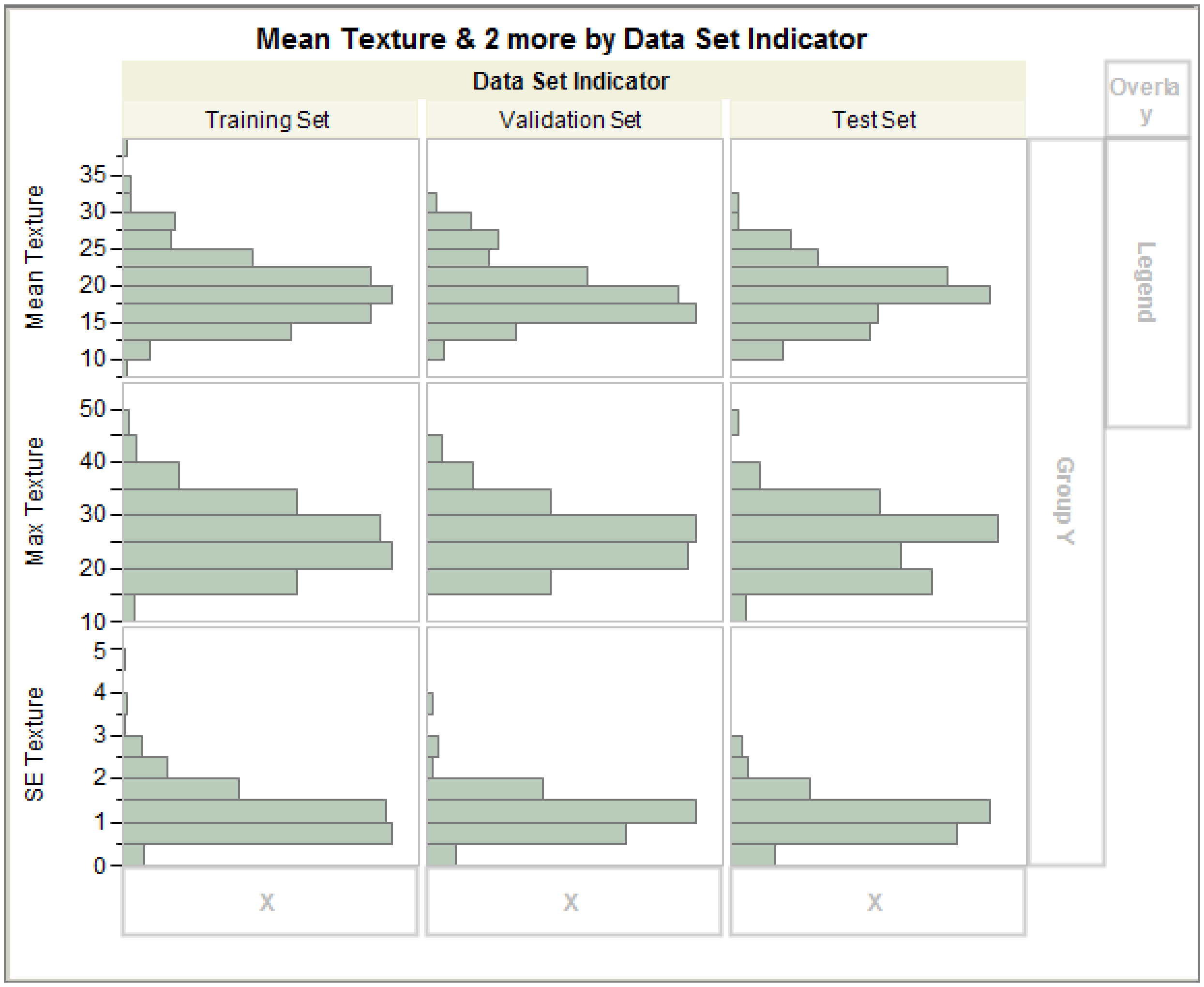
Figure 13. Boxplots of Mean, Max, and SE of Radius, Perimeter, and Area, by Analysis Data Set



Now, we could have obtained boxplots for all ten variables in the format shown in Figure 13. But, because of the differences in scaling, the boxplots for the rest of the variables would have appeared as horizontal lines, showing no detail whatsoever. So, a caveat in using `Graph Builder` is that variables need to be comparable in terms of scale in order for the type of display shown in Figure 13 to be meaningful.

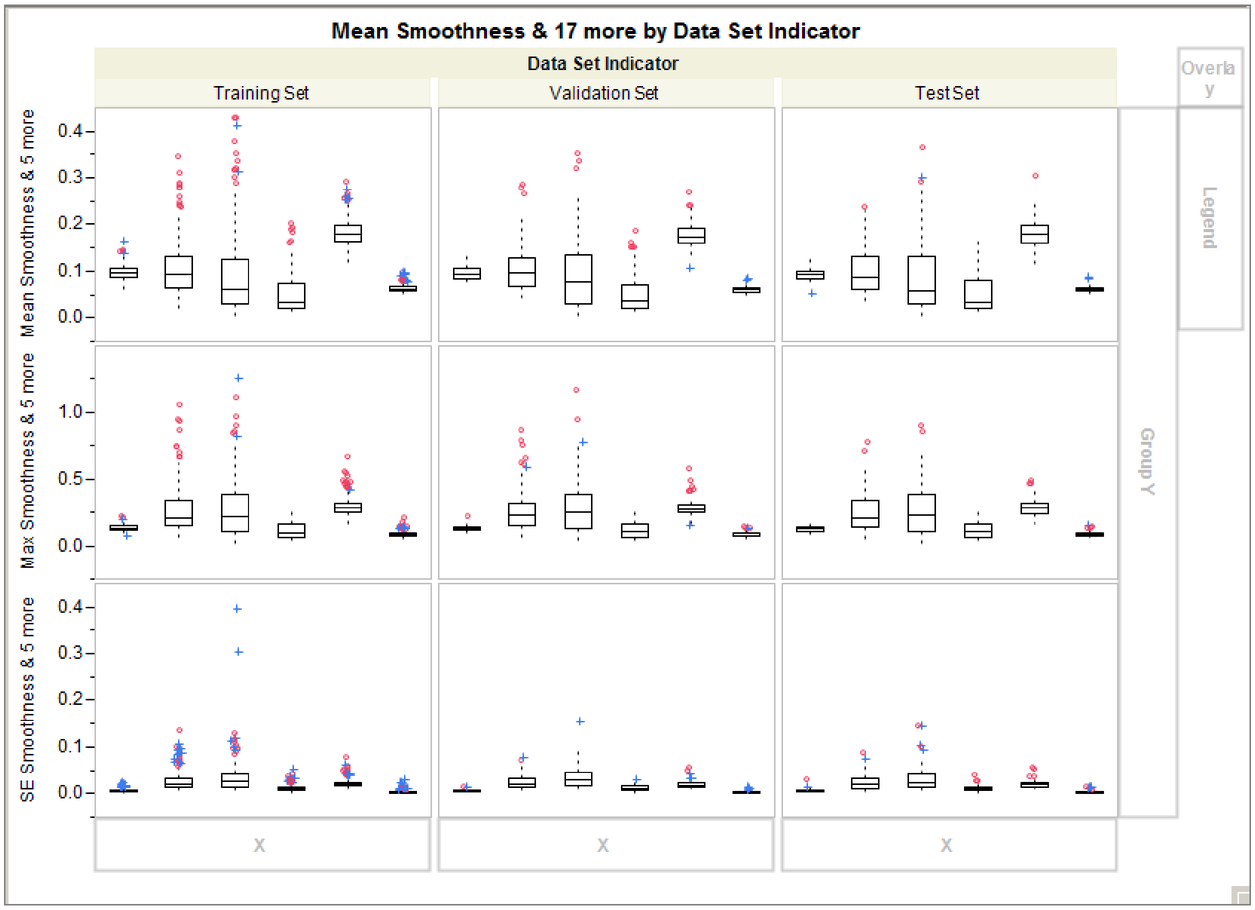
To view the remaining variables, because of the comparability of scaling issue, we separated `Texture` from the remaining six variables. Figure 14 shows a `Graph Builder` view of the `Texture` variable, using histograms instead of boxplots (script is `Graph Builder – Texture`). Again, the distributions appear consistent across analysis sets.

Figure 14. Histograms of Mean, Max, and SE of Texture, by Analysis Data Set



Boxplots for the remaining variables, `Smoothness`, `Compactness`, `Concavity`, `Concave Points`, `Symmetry`, and `Fractal Dim`, are shown in Figure 15. Again, the distributions are roughly comparable across analysis data sets, although there is evidence of two outliers on `SE Concavity` in the Training Set.

Figure 15. Boxplots for Mean, Max, and SE of Smoothness through Fractal Dim, by Analysis Data Set



LOGISITIC MODEL

At this point, we will begin the process of fitting classification models to our data. We will utilize only the training set for fitting models. So, we copy the values in the variable `Training Set` to the row states in our data table (click on the star to the right of `Training Set` in the `Columns` panel and select `Copy to Row States`). In the `Rows` panel, we should see that 222 rows are `Excluded` and `Hidden`.

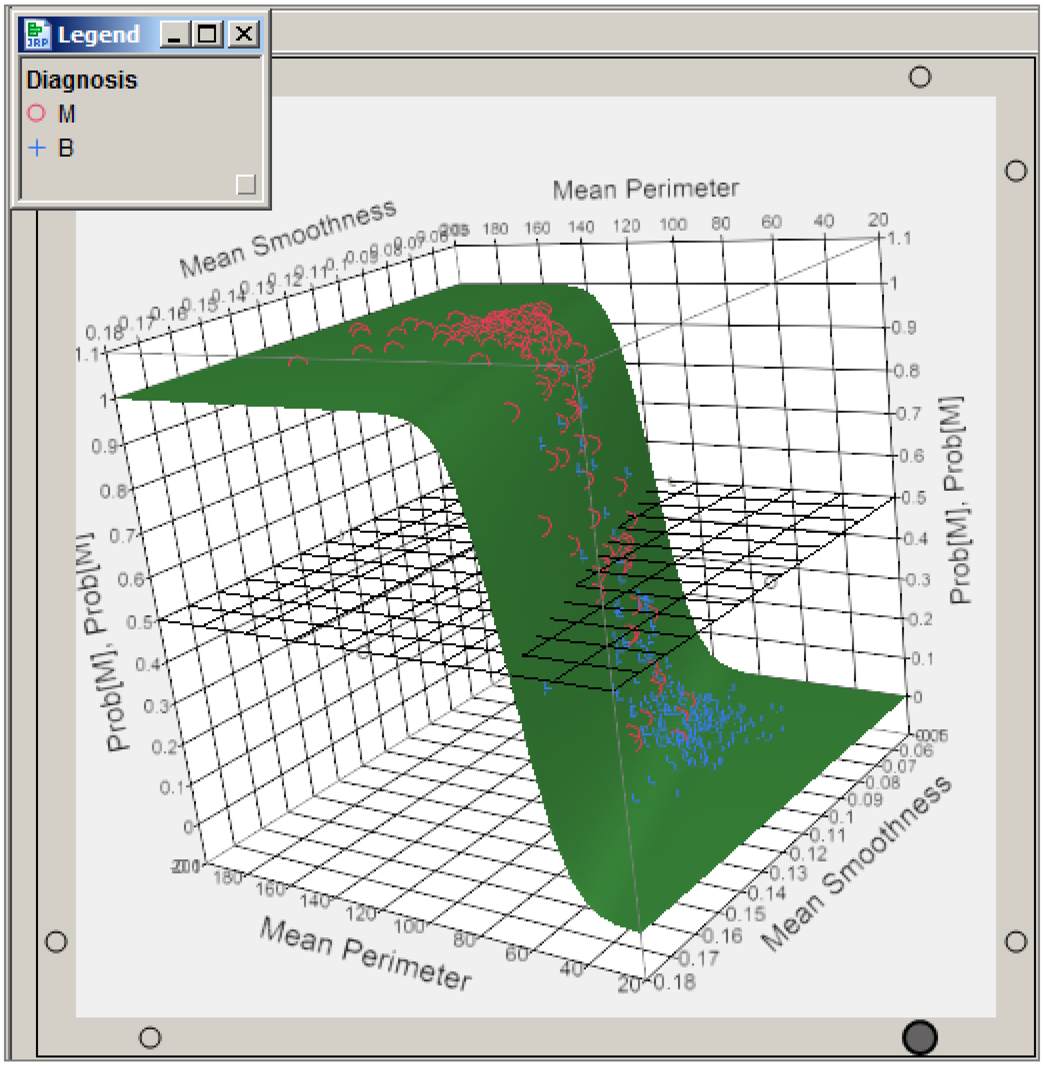
Our first issue relates to variable selection. We are given thirty potential predictors, but we could expand this set if we wished. For example, two-way interactions (such as the interaction of `Mean Perimeter` and `Mean Texture`, which might be a useful predictor) and/or quadratic terms could be explored in fitting classification models. This brings us quickly to an intimate awareness of the well-known "curse of dimensionality". Even with only 30 predictors, there are well over one billion different possible logistic models.

Why not simply use all thirty predictors? There are several reasons. Too many predictors can lead to overfitting. In other words, with enough predictors, it is possible to fit every point in a data set exactly. But such models tend not to generalize very well. So it is important to find a set of predictors that describes the structure of the data, and not the vagaries of individual points. Also, there can be computational issues when too many predictors are present. In particular, multicollinearity can degrade models obtained using certain algorithms (such as least squares regression).

For our breast cancer data, if we fit a logistic model to `Diagnosis` using all thirty predictors, we can perfectly classify every observation in our training set. But, when we apply the classification formula to our validation set, it behaves quite poorly. So it is important that we reduce the number of variables. We will do so using JMP's stepwise variable selection procedure.

But, before we do that, we want to take the opportunity to show what a logistic model, based on only two predictors, might look like. The surface plot in Figure 16 shows the surface obtained when the probability of a malignant diagnosis is modeled by only the two predictors `Mean Perimeter` and `Mean Smoothness` (the script for the model is `Logistic - Two Predictors`). Note the S-shaped logistic surface. `Prob[M]` refers to the probability of a malignant tumor. The values of `Prob[M]` are plotted for the points in our training set; these points are plotted as red circles for malignant tumors and blue plus signs for benign tumors. Note that a grid has been inserted at `Prob[M] = 0.5`. If you were to classify observations with `Prob[M]` values above 0.5 as malignant and below 0.5 as benign, then you can see that the classification rule would correctly classify a fair number of observations (to be exact, 312 of the 347 observations are correctly classified). (This plot was obtained using `Graph > Surface Plot`, using the predicted probabilities from the relevant `Fit Model` analysis.)

Figure 16. Logistic Model Based on Mean Perimeter and Mean Smoothness

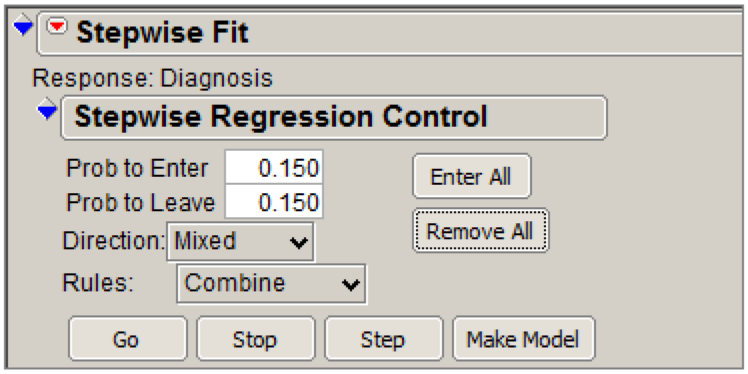


It's important to realize that logistic regression results in an estimate of the probability of class membership, conditional on the values of the predictor variables. So, it makes sense to classify a new observation into the malignant class if and only if `Prob[M]` exceeds (or equals) 0.5. (If you are following along and have saved columns based on this analysis, please delete them now.)

We now proceed to find a "best" logistic model for our training data. We go `to Analyze > Fit Model`, where we enter `Diagnosis` as `Y` and all thirty of our predictors in the `Construct Model Effects` box. Under `Personality`, we choose `Stepwise`. This window can be obtained by running the script `Stepwise Logistic Launch Window`.

Clicking `Run Model` opens the `Stepwise Fit` report. This window allows the user to make stepwise selection choices similar to those available in linear regression analysis. Just as a reminder, ***forward*** and ***backward*** selection procedures consist, respectively, of entering the most desirable variable and removing the least desirable variable. We will do our variable selection using a ***mixed*** procedure, meaning that each `Enter` step will be followed by a `Remove` step. To communicate this to JMP, we set `Direction` equal to `Mixed`. We will also set both the `Prob to Enter` and `Prob to Leave` values at 0.15. This will allow variables with significance level below 0.15 to enter, and variables that have entered at a previous stage, but which now have significance levels exceeding 0.15, will be available to be removed. The `Stepwise Regression Control` panel is shown in Figure 17; it can be obtained by running the script `Stepwise Fit Report`.

Figure 17. Stepwise Fit Panel for Logistic Variable Selection



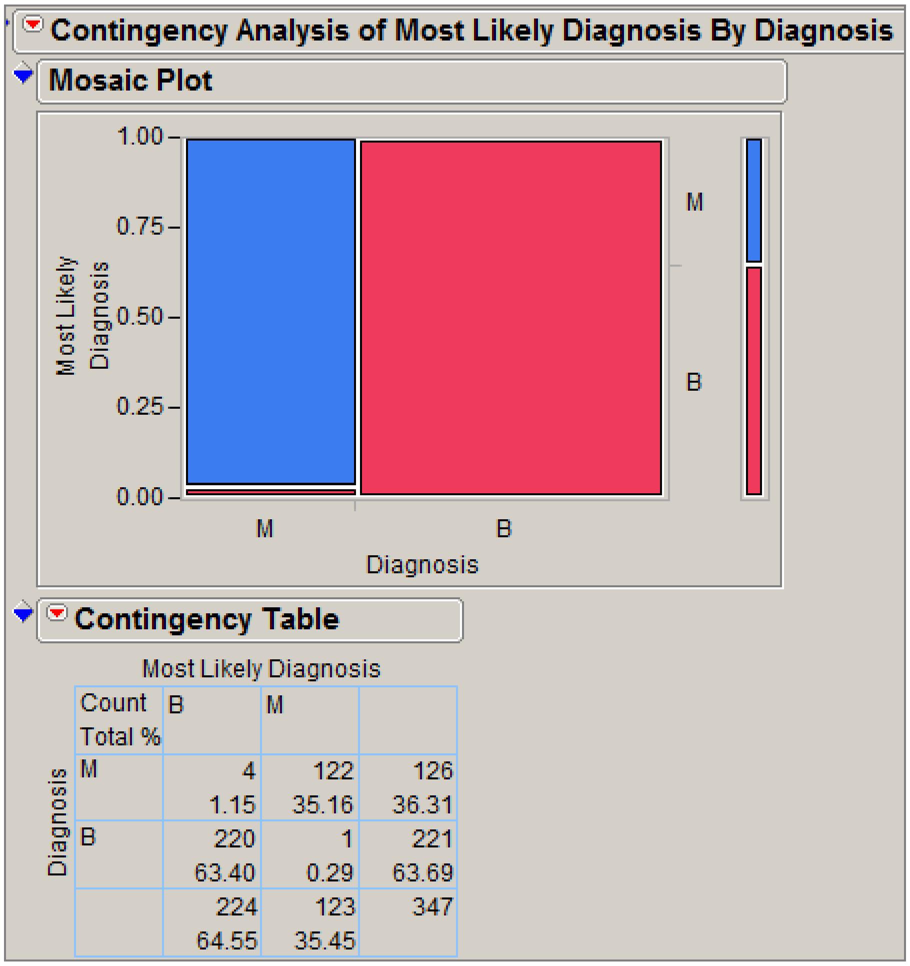
Response: Diagnosis

Prob to Enter 0.150
Prob to Leave 0.150
Direction: Mixed
Rules: Combine

When we click on `Go`, seven variables are selected. Then, we click on `Make Model`. This results in a `Fit Model` window containing the specification for our seven predictor model (the script is called `Stepwise Logistic Model`). The report that opens when we click `Run Model` shows details about the model fit. Under the red triangle, we can ask to view the `Receiver Operating Characteristic (ROC)` curve or the `Lift` curve. We will ask to `Save Probability Formula`. This saves a number of formulas to the data table, including `Prob[M]` and `Prob[B]`, namely, the probability that the tumor is malignant or benign, respectively, as well as a column called `Most Likely Diagnosis`, which gives the `Diagnosis` class with the highest probability, conditional on the values of the predictors.

To see just how well the classification performs on the training data, we can now go to `Analyze > Fit Y by X`. Enter `Most Likely Diagnosis` as `Y, Response`, and `Diagnosis` as `X, Factor`. The resulting mosaic plot and contingency table are shown in Figure 18. Of the 347 rows, only five are misclassified. We remind the reader, though, that there is inherent bias in evaluating a classification rule on the data used in fitting the model. In a following section, we will compare our three models using the validation data set.

Figure 18. Mosaic Plot and Contingency Table for Classification Based on Logistic Model



Most Likely Diagnosis

Diagnosis	Count / Total %	B	M	
M		4 1.15	122 35.16	126 36.31
B		220 63.40	1 0.29	221 63.69
		224 64.55	123 35.45	347

RECURSIVE PARTITIONING

Our next model will be fit using JMP's partition platform, which provides a version of classification and regression tree analysis. The partition platform allows both response and predictors to be either continuous or categorical. Continuous predictors are split into two partitions according to cutting values, while categorical predictors (predictors that are nominal or ordinal) are split into two groups of levels.

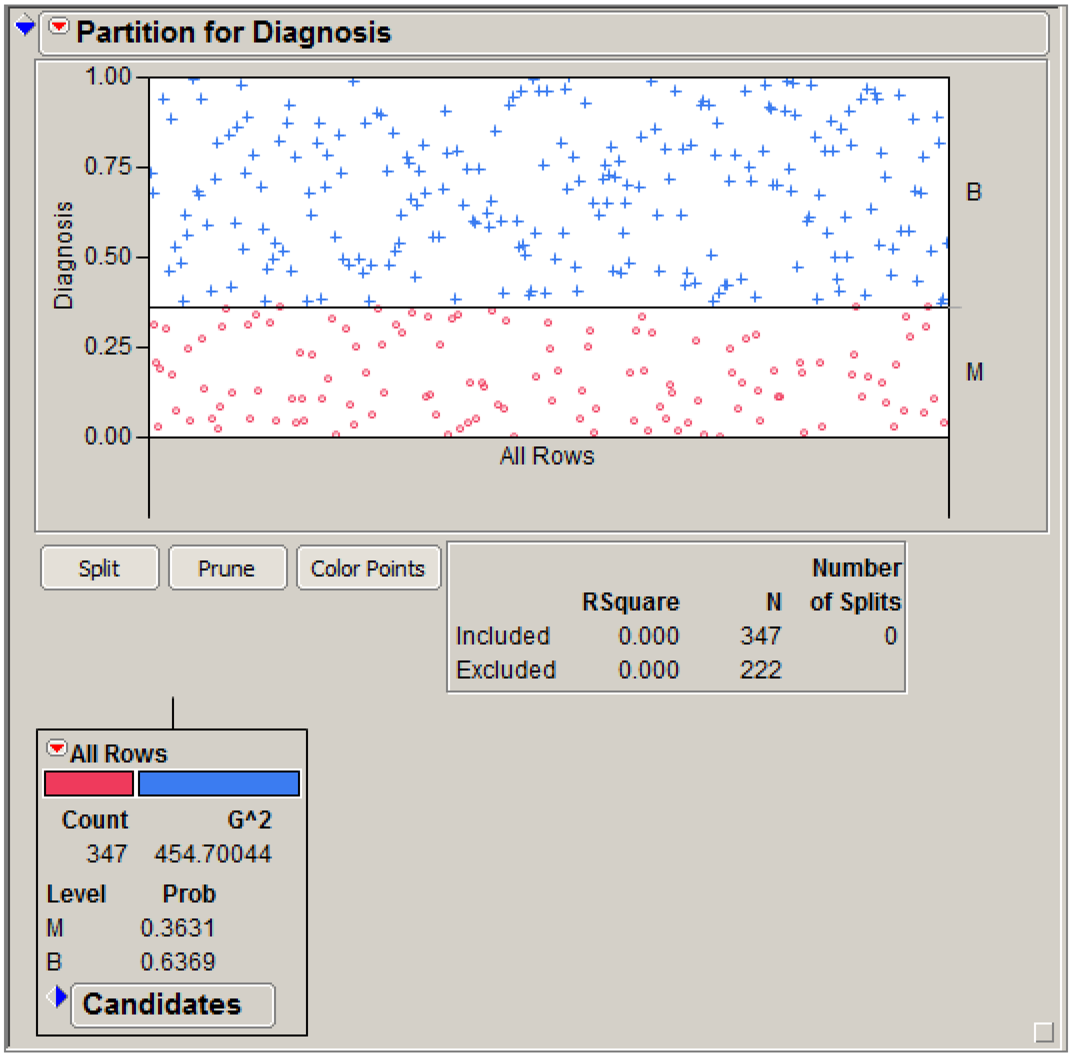
If the response is continuous, the sum of squares due to the differences between means is a measure of the difference in the two groups. Both the variable to be split at a given level and the cutting value for the split are determined by maximizing a quantity, called the *LogWorth*, which is related to the p-value associated with the sum of squares due to the difference in means. In the case of a continuous response, the fitted values are the means within the two groups.

If the response is categorical, as in our case, the splits are determined by maximizing a LogWorth statistic that is related to the likelihood ratio chi-square statistic, reported in the JMP output as "G^2". In this case, the fitted values are the estimated proportions, or response rates, within groups.

The partition platform is useful for both exploring relationships and for modeling. It is very flexible, allowing a user to find not only splits that are optimal in a global sense, but also node-specific splits that satisfy various criteria. The platform provides only a minimal ***stopping rule*** — that is, a criterion to end splitting. This rule is based on a user-defined minimum node size. The platform does not incorporate any other stopping rules; this is advantageous in that it enhances flexibility.

To fit our partition model, we go to `Analyze > Modeling > Partition`. Enter `Diagnosis` as `Y`, `Response`, and enter all thirty predictor variables as `X, Factor`. Clicking `OK` results in an initial report window. Under the red triangle, choose `Display Options >Show Split Prob` – this shows the split proportions in the nodes. This results in the report shown in Figure 19 (the script for this analysis is called `Initial Partition Report Window)`. The graph shows the blue plus signs, indicating benign tumors, and the red circles, indicating malignant tumors, separated by a horizontal line at 0.36, the overall proportion malignant.

Figure 19. Initial Partition Report Window



	RSquare	N	Number of Splits
Included	0.000	347	0
Excluded	0.000	222	

All Rows

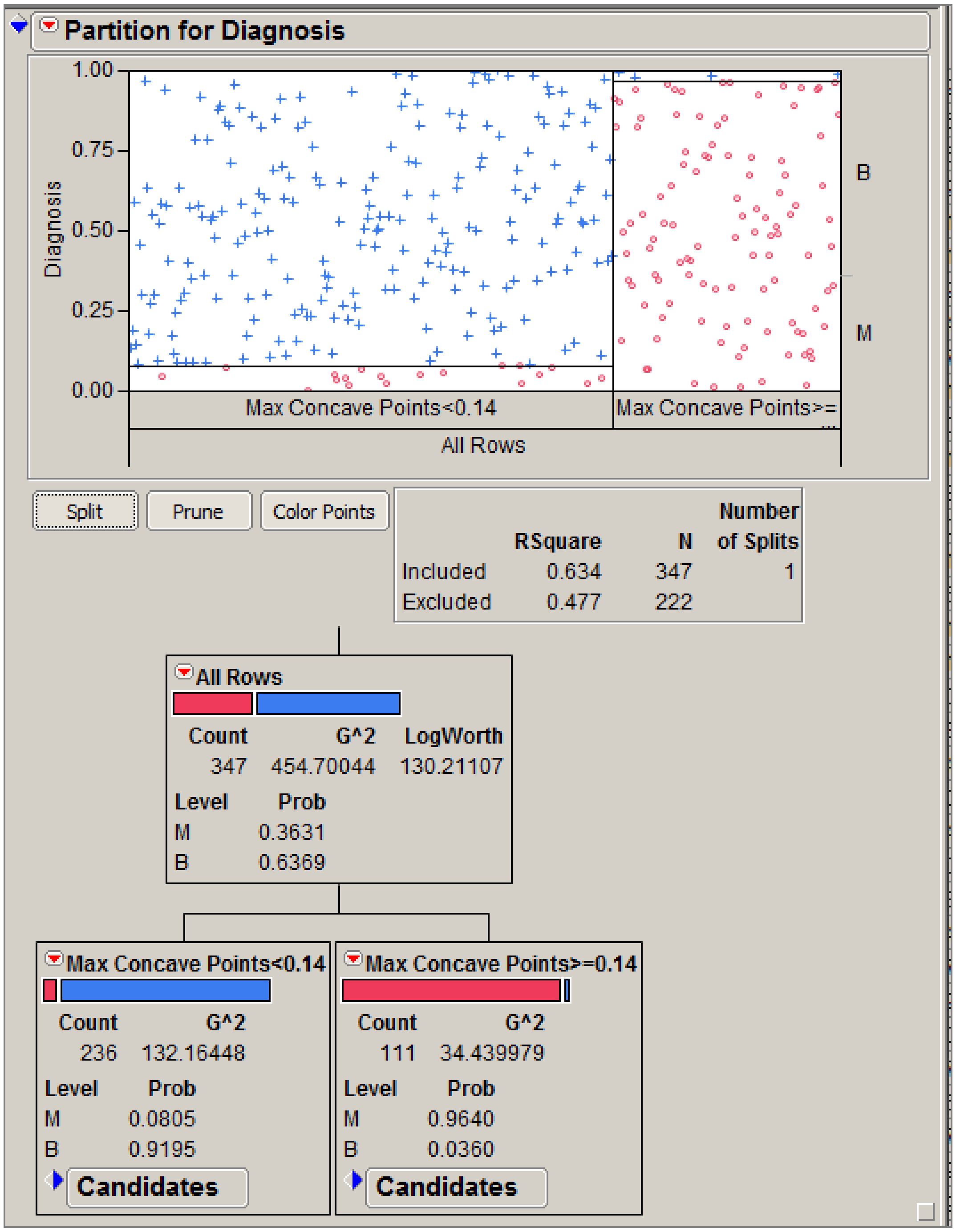
🟥 🟦

Count	G^2
347	454.70044

Level	Prob
M	0.3631
B	0.6369

We begin to split by clicking on the `Split` button. JMP's algorithm determines the best variable on which to split and the best cutting value of that variable. The tree, after the first split, is shown in Figure 20. The first split is on the variable `Max Concave Points`, and the observations are split at the value where `Max Concave Points` = 0.14. Of the observations where `Max Concave Points` >= 0.14 (the rightmost node), 96.40% are malignant. Of those for which `Max Concave Points` < 0.14, 91.95% are benign.

Figure 20. Partition Report after First Split



	RSquare	N	Number of Splits
Included	0.634	347	1
Excluded	0.477	222	

All Rows

Count	G^2	LogWorth
347	454.70044	130.21107

Level	Prob
M	0.3631
B	0.6369

Max Concave Points<0.14

Count	G^2
236	132.16448

Level	Prob
M	0.0805
B	0.9195

Max Concave Points>=0.14

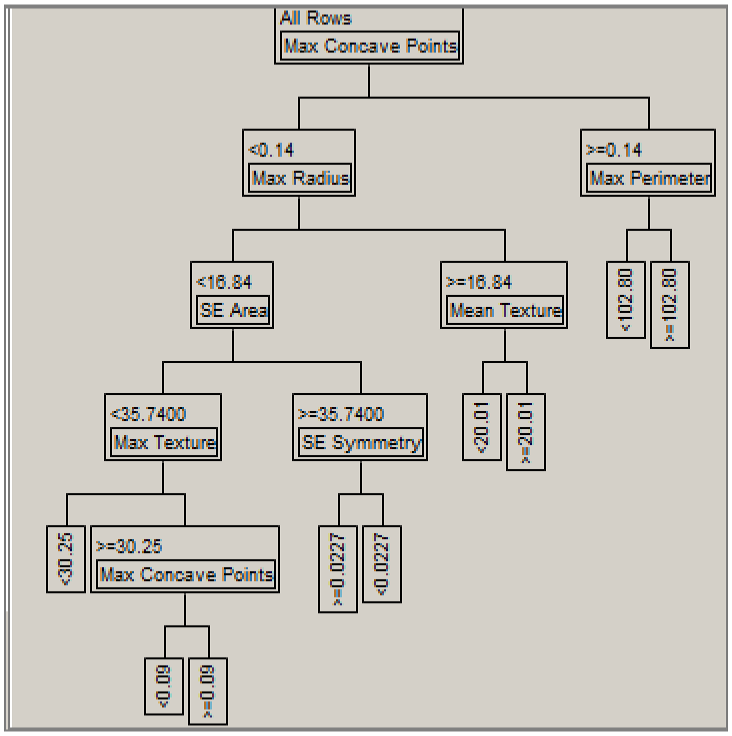
Count	G^2
111	34.439979

Level	Prob
M	0.9640
B	0.0360

We continue to split for a total of eight splits. At this point, no further splits occur. This is because there is a default minimum split size of five and further splitting of nodes would result in nodes of smaller size than five. It is absolutely true that we could have stopped splitting before reaching eight splits. In fact, we may have overfit the data, given that some of the final nodes contain very few observations.

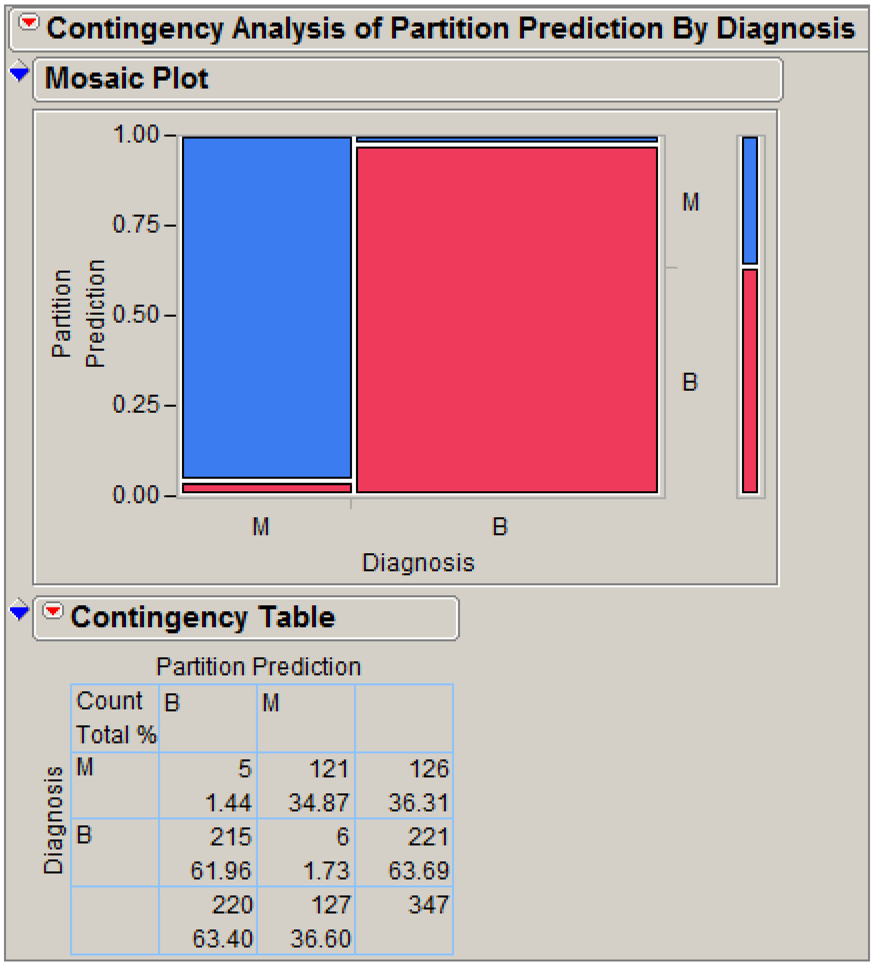
The red triangle at the top of the partition report gives the user many options, including `ROC Curve`, `Lift Curve`, `Leaf Report`, `K-Fold Cross Validation`, etc. The `Small Tree View` that is provided is shown in Figure 21. This report is given by the script `Partition Model`. Under the red triangle, we go to `Save Columns > Save Prediction Formula`. This saves two new columns to the data table, Prob(Diagnosis==M) and Prob(Diagnosis==B). These columns give the predicted probabilities, which are simply terminal node proportions, of the respective diagnosis classes. So, the classification rule consists of determining which node a new observation falls into, and predicting its probability of class membership based on the sample proportions in that node.

Figure 21. Small Tree View for Partition Model



In the data table, we define a new column called Partition Prediction. We construct a formula to define this column; this formula classifies an observation as “M” if Prob(Diagnosis==M)>=0.5, and as “B” otherwise. Once this is done, we go to Analyze > Fit Y by X. We enter Partition Prediction as Y, Response, and Diagnosis as X, Factor. The resulting mosaic plot and contingency table are shown in Figure 22. Of the 347 rows, eleven are misclassified. At first blush, this model does not seem to be performing as well as the logistic model. In a later section, we will compare it to the logistic model by evaluating it on the validation data set.

Figure 22. Mosaic Plot and Contingency Table for Classification Based on Partition Model



Partition Prediction

Diagnosis	Count / Total %	B	M	
	M	5 1.44	121 34.87	126 36.31
	B	215 61.96	6 1.73	221 63.69
		220 63.40	127 36.60	347

NEURAL NET

BACKGROUND

As we mentioned earlier, data mining algorithms for classification and prediction can be based on ***neural nets***. Neural net algorithms were originally inspired by how biological neurons are believed to function. Starting in the 1940s, scientists in the area of artificial intelligence pursued the idea of designing algorithms that “learn” in a fashion that emulates neurons in the human body.

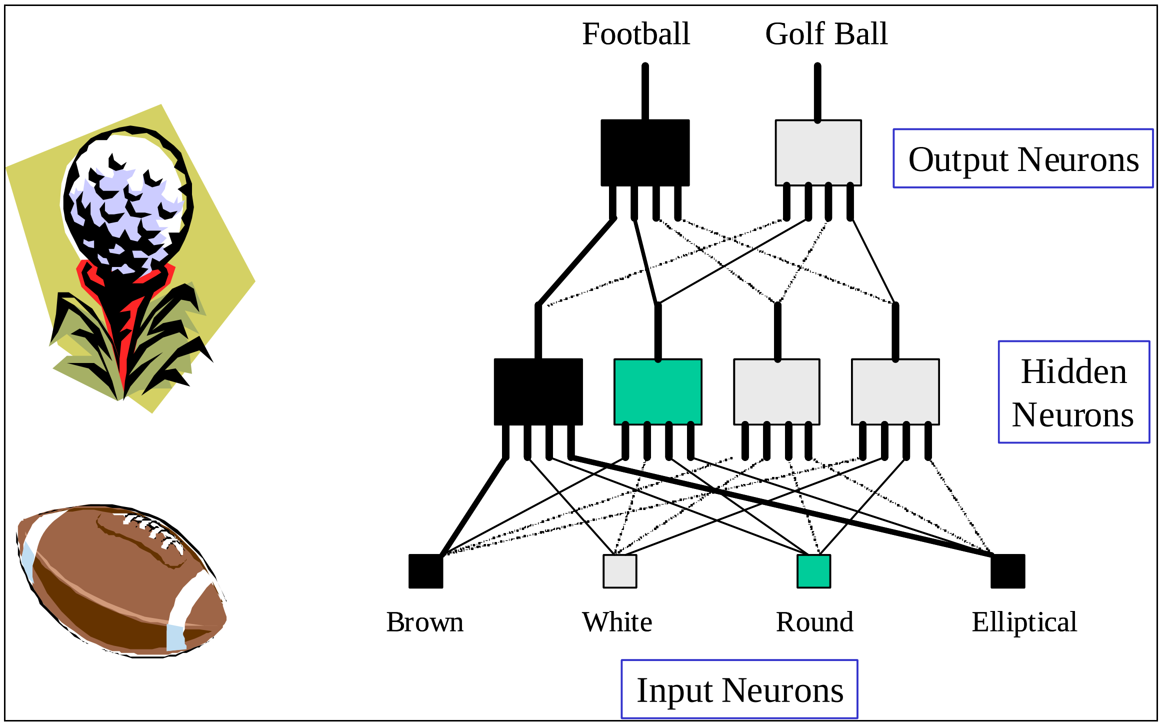
The science of biologically informed computation had its origins in a seminal paper called “A Logical Calculus of Ideas Immanent in Nervous Activity,” by Warren McCulloch and Walter Pitts (both at MIT), published in 1943. This paper showed that the “ideas” in nerve cells were carried by the entire collection of neurons as a whole. They were implicit. (***Immanen***t, in the title of the paper, means “having existence or effect only within the mind or consciousness” – Webster’s.)

Research since these early days has leveraged the idea of utilizing neuron-based processing elements arranged in nets to produce the algorithms that appear today in neural net software. Neural nets are based on neuron-like processing elements arranged in nets. There is an input layer of neurons, an output layer, and a hidden layer where processing occurs.

For some intuition on the way neural net algorithms work, consider the diagram in Figure 23, where a neural net is attempting to distinguish a golf ball from a football. Note that the neurons are arranged in three layers: Input neurons, hidden neurons (the ***hidden layer***), and output neurons. Neurons can be excited to various degrees – they are not completely "on" (excited, black) or "off" (unexcited, white). The input neurons are sensory, in that they react to a stimulus and output a value that reflects the intensity of that stimulus. A pattern of weights connects the remaining neurons and determines the degree to which each is excited. The output and hidden layer neurons sum their inputs and compare this sum to a threshold value to determine their output.

The net "learns" when it makes an error. The degree of error is propagated back through the network. Connections that were wrong are down-weighted, while those that were correct are strengthened. Active neurons with strong connections that were wrong are strongly penalized. In our golf ball and football example, prior to training, the connection strengths and excitation levels would be random.

Figure 23. Schematic of a Neural Net Being Trained to Distinguish a Football from a Golf Ball



Now, in a mathematical sense, a neural net is nothing more than a nonlinear regression model. In its implementation of neural nets, JMP uses standard nonlinear least squares regression methods. Although a general neural net can have many hidden layers, one layer is considered sufficient for most modeling situations, and JMP utilizes a single hidden layer. Each hidden node is modeled using the logistic function applied to a linear function of the predictors. In a classification situation, such as ours, the output value consists of the logistic function applied to a linear function of the hidden nodes. This means that, for example, for thirty input variables, one response, and k hidden nodes, the number of parameters to be estimated is 31*k + k + 1.

With so many parameters, it is easy to see that a major advantage of a neural net is its ability to model a variety of response surfaces. But the large number of parameters comes at a cost. There are many local optima, and convergence to a global optimum can be difficult. Also, with so many parameters, overfitting is problematic. (This is why validation sets are critical to neural net modeling strategies.) Another disadvantage of neural net models is that they tend not to be interpretable, due to the hidden layer.

JMP's implementation provides a user-specified overfitting penalty to help minimize overfitting issues. The user is also able to set the number of nodes in the hidden layer. Here, a small number can lead to underfitting and a large number can lead to overfitting. Each application of the algorithm has a random start, and JMP refers to these individual fits as ***tours***. About twenty tours are recommended in order to find a global optimum.

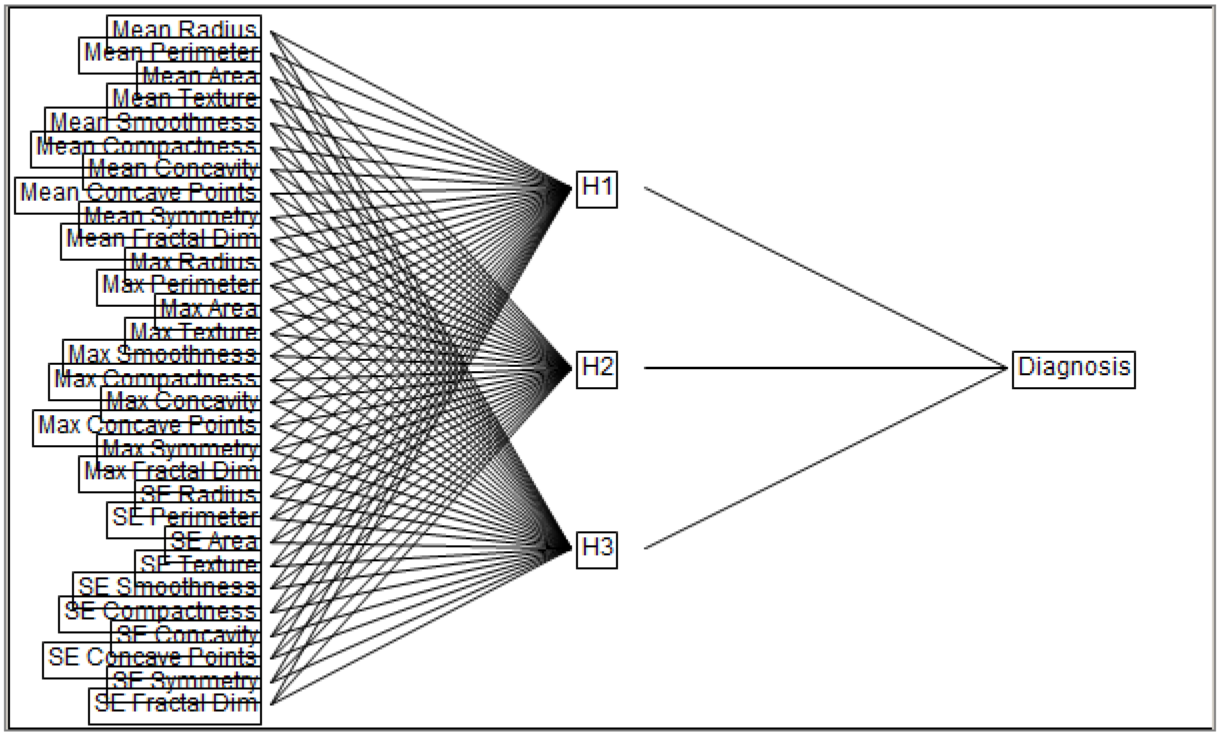
JMP also provides two methods to help a user select a neural net model that will extend well to new data. One of these methods is called `Random Holdback`. In this method, a sample of the observations is withheld (the ***holdback*** sample) and the remaining observations are used to train a neural net. JMP computes an R^2 value for the training sample, and then applies the neural net to the holdback sample and calculates an R^2 for that sample, which is called the ***crossvalidation*** $\boldsymbol{R^2}$, denoted `CV RSquare` in the JMP report. The user can vary the number of nodes and the overfit penalty in an attempt to find a model that generalizes well to the holdback sample. This method works well for large samples, where one can easily fit a model to 75% or fewer of the observations (JMP uses 2/3 of the complete data set by default).

The second method is called `K-Fold Crossvalidation`. Here, a neural net model is fit to all of the data to provide starting values. Then, the observations are divided randomly into K groups (or ***folds***). Each of these groups, in turn, is treated as a holdback sample. For each of the K groups, a model is fit to the data in the other (K – 1) folds, using the starting values from the full fit. The model is then extended to the holdback group. An R^2 is calculated for each holdback sample, and these are averaged to give a `CV RSquare` that represents how the model might perform on new observations. (The starting values from the full fit are used because the function being optimized is multimodal, and this practice attempts to bias the estimates for the submodels to the mode of the overall fit.)

A FIRST MODEL – NEURAL NET 1

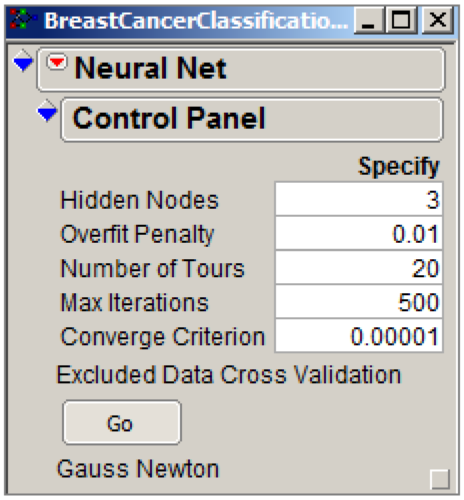
We will begin by fitting a model to our breast cancer data without any use of cross-validation. To fit a neural net model, we go to `Analyze > Modeling > Neural Net`. As usual, we enter `Diagnosis` as `Y, Response` and all thirty predictors as `X, Factors`. A diagram of a model with three hidden nodes is shown in Figure 24; this can be obtained by choosing `Diagram` under the red triangle in the report window. The final model will be a linear combination of three models, each of which relates one of the hidden nodes, H1, H2, and H3, to the thirty predictors.

Figure 24. Diagram of Neural Net Model for Thirty Predictors with Three Hidden Nodes



The settings for the neural net are entered in the `Control Panel`, shown in Figure 25. Here, we have accepted the JMP 8 default settings, with the exception of requesting 20 as the `Number of Tours`. Note that the overfit penalty in JMP 8 is set by default at 0.01, a value that we accept for this model. (Smaller values of the overfit penalty led to models with sharp curves, suggesting overfitting.) When one clicks `GO`, JMP performs the fit. Depending on the data set size and the settings in the `Control Panel`, the fitting procedure can take a noticeable amount of time.

Figure 25. Neural Net Control Panel Settings



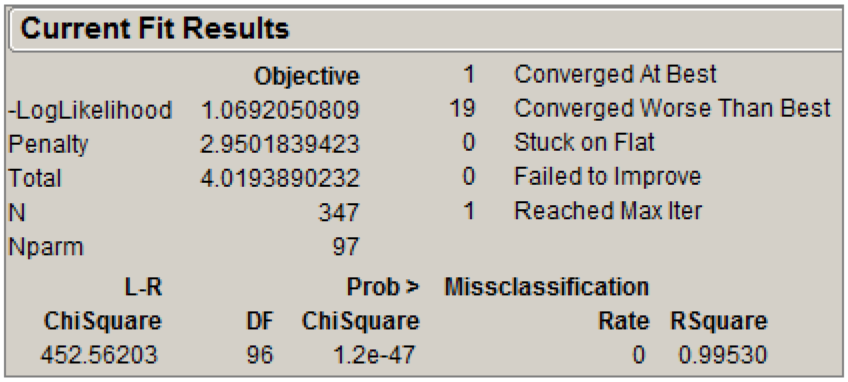
	Specify
Hidden Nodes	3
Overfit Penalty	0.01
Number of Tours	20
Max Iterations	500
Converge Criterion	0.00001

Excluded Data Cross Validation

Gauss Newton

Results for the current fit (JMP provides a `Fit History` panel that retains history for all fits) are shown in the `Current Fit Results` panel (Figure 26). We note that one of the 20 tours `Converged at Best`, and this is the fit that JMP uses. Note that 97 parameters (`Nparm`) have been estimated. These are listed in the `Parameter Estimates` panel, but they are not of intrinsic interest.

Figure 26. Results on Fit to Diagnosis with Thirty Predictors



Current Fit Results

	Objective
-LogLikelihood	1.0692050809
Penalty	2.9501839423
Total	4.0193890232
N	347
Nparm	97

1	Converged At Best
19	Converged Worse Than Best
0	Stuck on Flat
0	Failed to Improve
1	Reached Max Iter

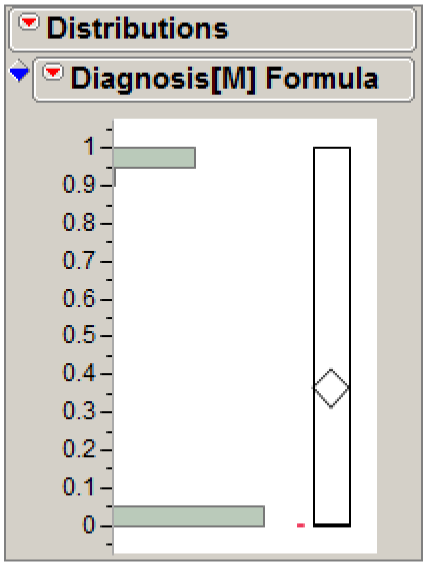
L-R ChiSquare	DF	Prob > ChiSquare	Missclassification Rate	RSquare
452.56203	96	1.2e-47	0	0.99530

We have saved the script for this fit in `Neural Net Report`. However, since the tours have random starting values, you will likely not obtain the same fit using this script. To obtain the same model as the one we explore below, run the script `Neural Net 1`.

Under the red triangle, we choose the option to `Save Formulas`. This adds five new columns to the data table: the formulas for the three hidden nodes, called `H1 Formula`, `H2 Formula`, and `H3 Formula`; a formula called `SoftSum`, which calculates an intermediate result; and the final estimated probability, in our case called `Diagnosis[M] Formula`, which applies a logistic function to the estimated linear function of the hidden nodes.

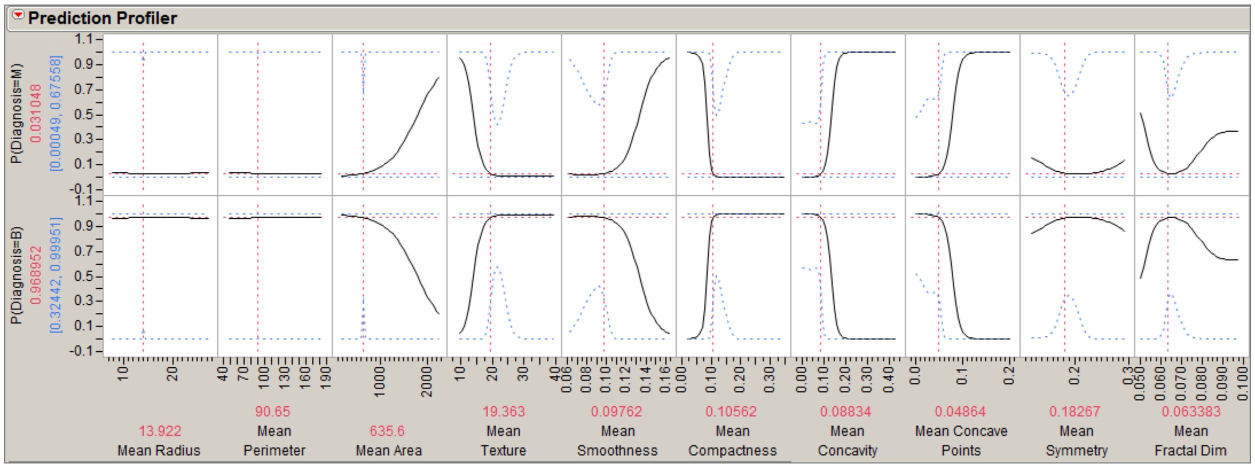
A histogram for the predicted probabilities of malignancy, `Diagnosis[M] Formula`, is shown in Figure 27. Note that the model probabilities tend to be close to 1 or to 0.

Figure 27. Histogram for Neural Net Estimate of Probability of Malignant



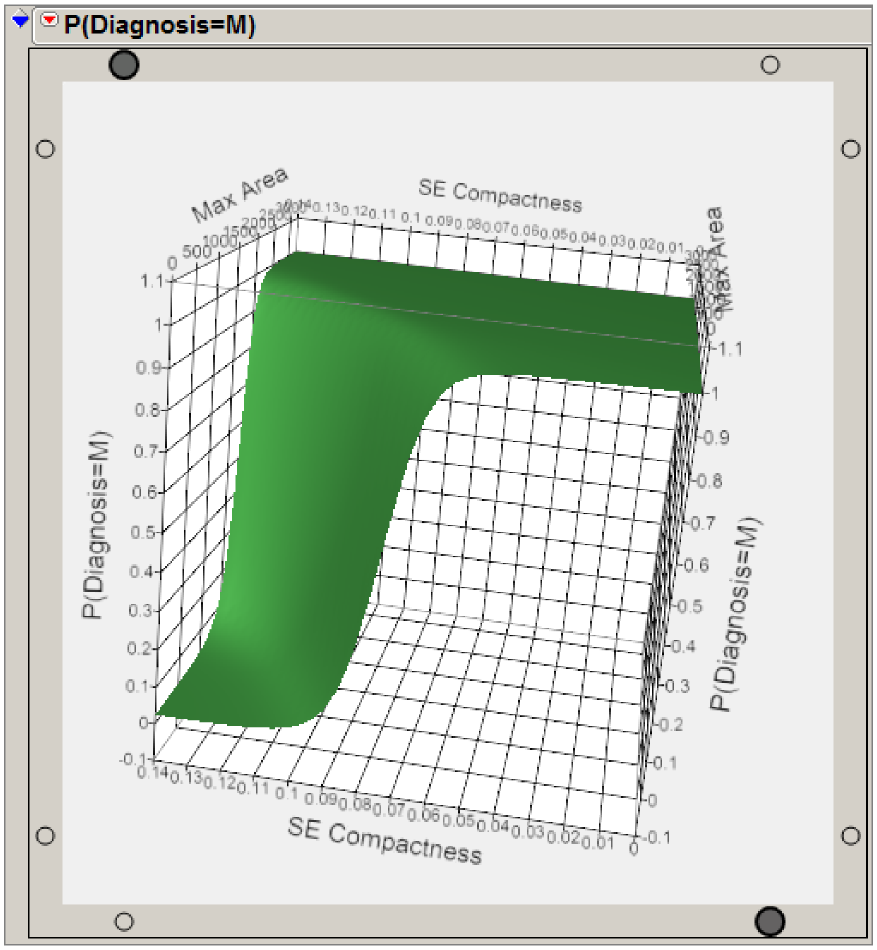
To get some sense of what our 31-dimensional model looks like, we have various options. Under the red triangle in the Neural Net report window, we can choose `Profiler`. This opens the `Prediction Profiler`, a portion of which is shown in Figure 28. The dotted red vertical lines represent settings for each of the variables. For a given variable, the traces in the `Prob(Diagnosis=M)` and `Prob(Diagnosis=B)` squares represent the cross section of the fitted model in that variable's direction, for the settings of the other variables. When we change one variable's value, we can see the impact of this change on the surface for all other variables. As one changes values and scrolls through the plots, one sees that the surface appears fairly smooth, with some steep peaks, but no very jagged areas.

Figure 28. Part of Prediction Profiler for Neural Net Model 1



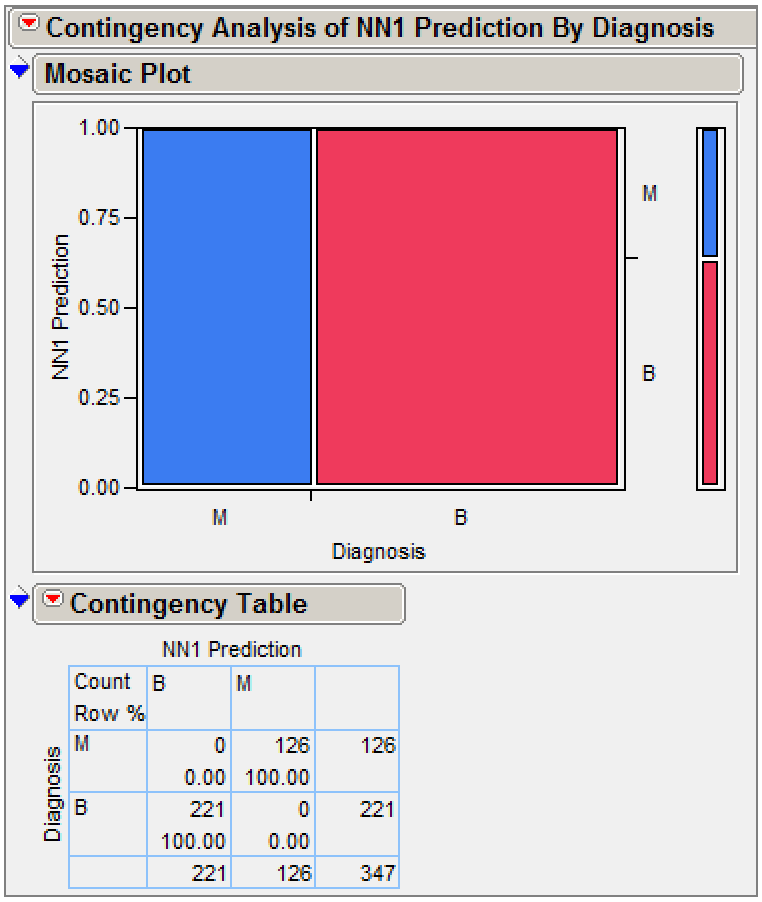
Another way to visualize the surface is using the `Surface Profiler`. This gives a three dimensional view of the effect of predictor variables, taken two at a time, on `Prob(Diagnosis=M)` and `Prob(Diagnosis=B)`. Figure 29 shows one of the 30*29/2 = 435 possible plots.

Figure 29. Surface Plot of Probability of Malignancy as a function of Max Area and SE Smoothness in Neural Net Model 1



To assess the performance of this model on the training set, we define a variable called `NN1 Prediction`. As usual, predicted probabilities equal to or greater than 0.50 will lead to a classification of M, while values less than 0.50 will result in a classification of B. The resulting `Fit Y by X` analysis, for the training set, is shown in Figure 30. There are no misclassifications. But keep in mind that neural nets have a tendency toward overfitting.

Figure 30. Mosaic Plot and Contingency Table for Classification Based on Neural Net Model 1



NN1 Prediction

Count / Row %	B	M	
M	0 / 0.00	126 / 100.00	126
B	221 / 100.00	0 / 0.00	221
	221	126	347

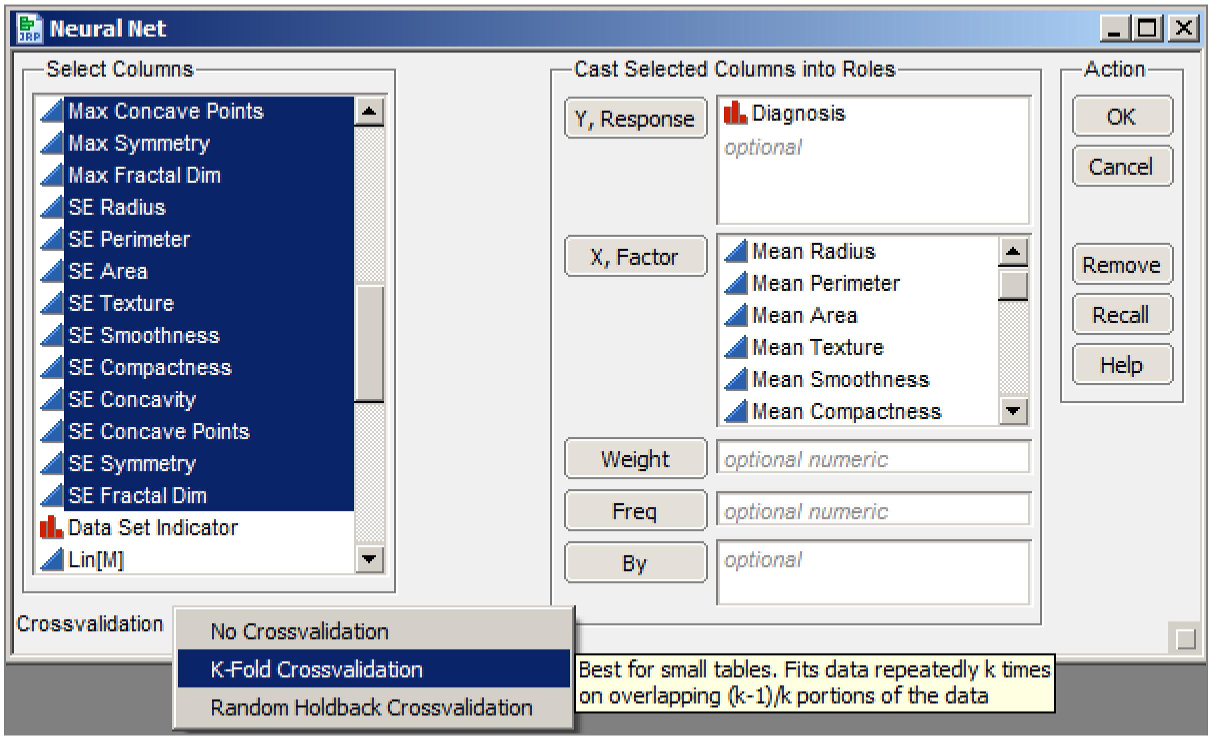
Diagnosis

One can explore the effects of various variable combinations and specifications of hidden layer nodes. For example, if one uses the variables from our logistic stepwise selection and a single hidden node, then the resulting neural net model gives predicted probabilities that are similar to those obtained using the logistic fit. One can also utilize cross-validation to determine a model that might generalize well to new data. This is the topic of the next section.

A SECOND MODEL – NEURAL NET 2

In this section, we will explore various neural net model architectures using K-fold cross-validation. Once again, we go to `Analyze > Modeling > Neural Net`. We fill in the launch window as before, with `Diagnosis` as `Y` and all thirty predictors as our `X`s. However, we make another selection, namely, we choose `K-Fold Crossvalidation` from the drop-down menu at the bottom of the launch window, as shown in Figure 31. (We note that we will be doing crossvalidation within the context of our 347 observation training set, in order to have consistency with our other three models. In practice, with K-fold crossvalidation, we would not hold out a separate validation set.)

Figure 31. Launch Window for K-Fold Crossvalidation Neural Net Model

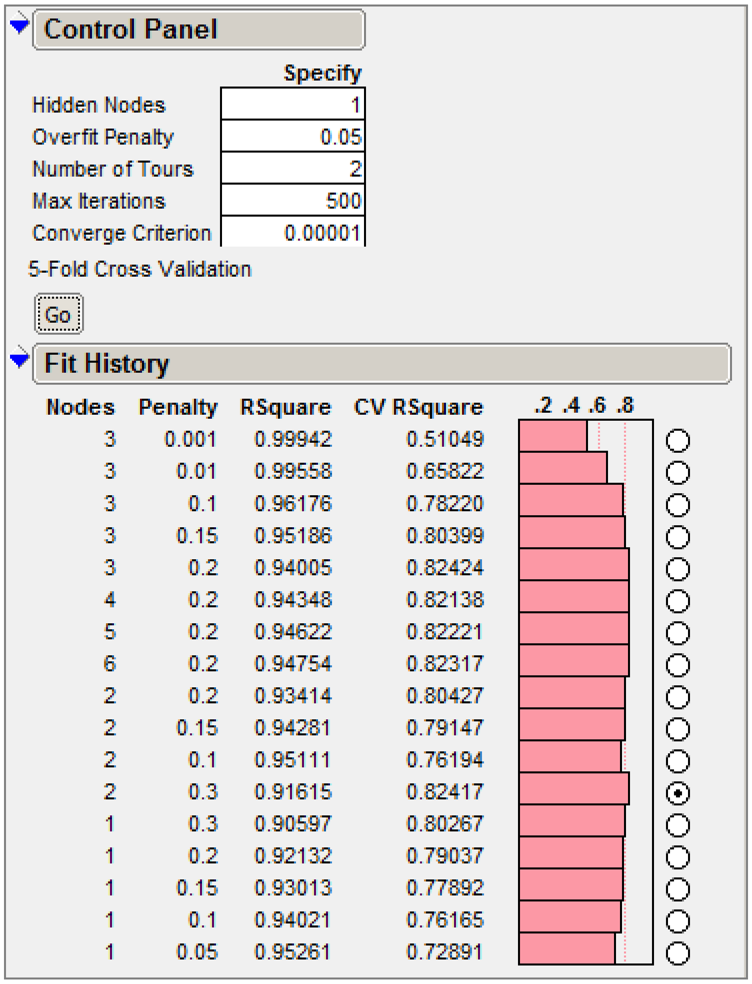


Clicking `OK` brings up the `Control Panel`. The default number of groups is 5, and we consider that reasonable given the size of our data set – we have 347 observations, and so each of the five training samples will contain at least 276 rows (there are about 347/5 = 69 observations in each fold, and four folds in the training sample, so about 69*4 = 276 observations in each training set). When we click `Go`, the `Control Panel` changes back to give us the choices that we have seen earlier. In particular, we will explore the effect of different numbers of hidden nodes and various values of the overfit penalty.

The `History Panel` in Figure 32 shows details for about 17 sequential fits to the data, where we manually varied the overfit penalty and the number of nodes. (We mention in passing that JMP automates this process using the option `Sequence of Fits` under the red triangle.) We would like to choose a model that is simple, meaning that it has a small number of nodes, and which seems to generalize well based on the `CV RSquare`. To this end, we choose a model with two nodes and an overfit penalty of 0.3. In Figure 32, this model has been chosen by clicking the radio button to its right. This updates the report window with information for this model.

As before, because of the random starts, the results that we obtain may differ from yours. However, the script `Neural Net 2` reproduces the final model that we utilize below.

Figure 32. Fit History Panel for Seventeen Exploratory Model Fits



Control Panel

	Specify
Hidden Nodes	1
Overfit Penalty	0.05
Number of Tours	2
Max Iterations	500
Converge Criterion	0.00001

5-Fold Cross Validation

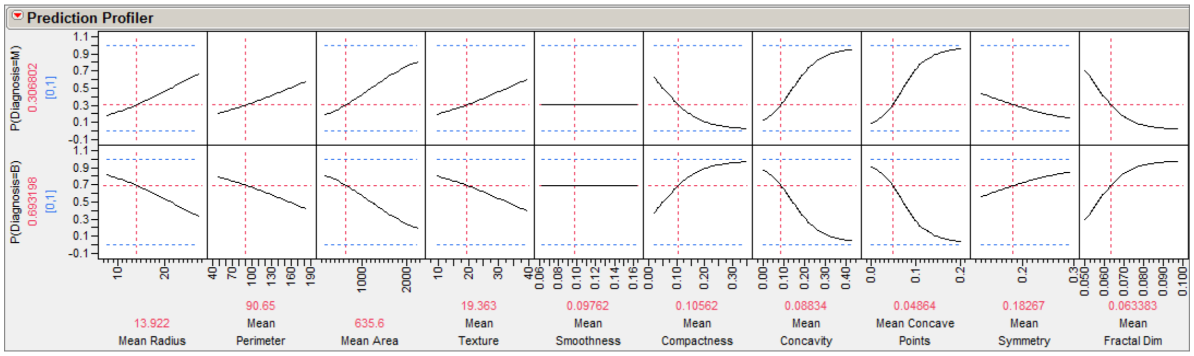
Go

Fit History

Nodes	Penalty	RSquare	CV RSquare	
3	0.001	0.99942	0.51049	○
3	0.01	0.99558	0.65822	○
3	0.1	0.96176	0.78220	○
3	0.15	0.95186	0.80399	○
3	0.2	0.94005	0.82424	○
4	0.2	0.94348	0.82138	○
5	0.2	0.94622	0.82221	○
6	0.2	0.94754	0.82317	○
2	0.2	0.93414	0.80427	○
2	0.15	0.94281	0.79147	○
2	0.1	0.95111	0.76194	○
2	0.3	0.91615	0.82417	⦿
1	0.3	0.90597	0.80267	○
1	0.2	0.92132	0.79037	○
1	0.15	0.93013	0.77892	○
1	0.1	0.94021	0.76165	○
1	0.05	0.95261	0.72891	○

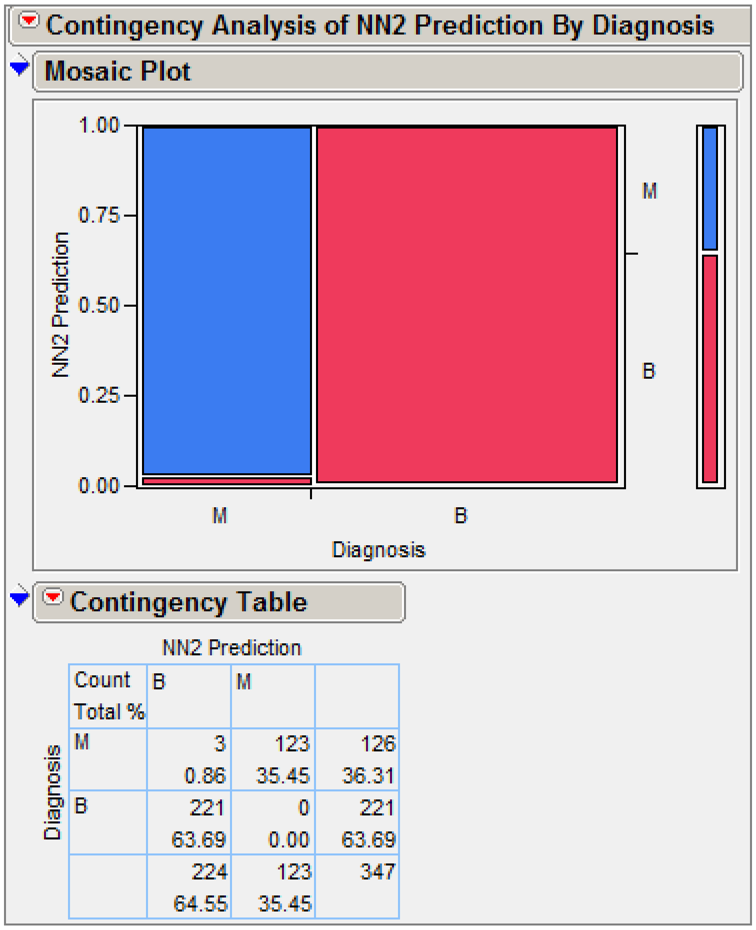
Now we can go to the red triangle and click on `Profiler` and `Surface Profiler` to get a sense of what this model looks like. When we do this, we see that this model seems considerably smoother than our Neural Net 1 model. For comparison, we show some of the profiler traces in Figure 33.

Figure 33. Part of Prediction Profiler for the Neural Net Model 2



In particular, since we have chosen this model with the radio button, JMP allows us to save this model's formulas. To do this, we choose `Save Formulas` from under the red triangle. This inserts four new columns into the data table: `H1 Formula 1`, `H2, Formula 2`, `SoftSum 2`, and `Diagnosis[M] Formula 2`. We define a new column called `NN2 Prediction` that gives the classifications for Neural Net 2. The `Fit Y by X` analysis in Figure 34 shows three misclassifications. Recall that there were no misclassifications for Neural Net 1 on the training data (Figure 30). Perhaps Neural Net Model 1 overfit the data? We will keep that in mind during the model validation step.

Figure 34. Mosaic Plot and Contingency Table for Classification Based on Neural Net Model 2



		NN2 Prediction		
	Count Total %	B	M	
Diagnosis	M	3 0.86	123 35.45	126 36.31
	B	221 63.69	0 0.00	221 63.69
		224 64.55	123 35.45	347

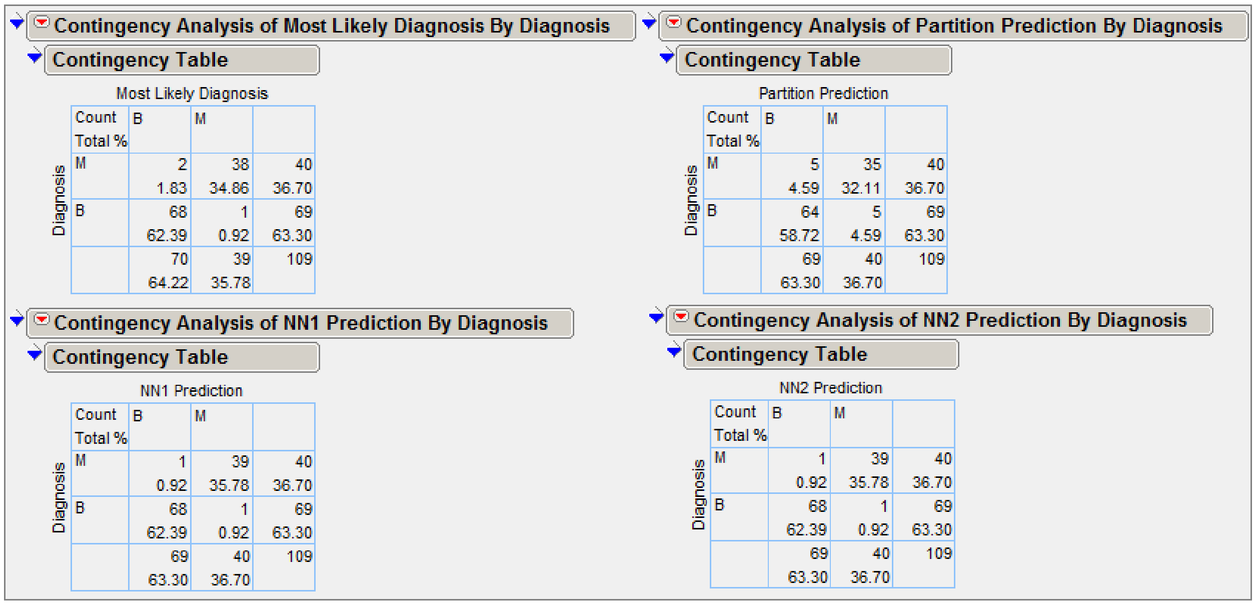
COMPARISON OF CLASSIFICATION MODELS

At this point, we will select a single model by comparing the performance of all four models on our validation set. This kind of activity is sometimes referred to as "competing models". Our single performance criterion will be the overall misclassification rate. But note that, in a real setting, we might want to weight the two types of misclassification errors differently, since overlooking a malignant tumor seems more serious than incorrectly classifying a benign tumor. Although JMP provides mechanisms for accomplishing this (for example, one could define a loss function and fit models using the nonlinear platform), we will not delve into this here.

We can easily switch our analyses to our validation set – simply locate the row state variable `Validation Set` in the `Columns` panel of the data table, left click on the red star to its left, and select `Copy to Row States`. All but the 109 observations in our validation set are now excluded and hidden. We go to `Analyze > Fit Y by X`, and we enter, `as Y, Response`, the variables `Most Likely Diagnosis` (the logistic classification), `Partition Prediction`, `NN1 Prediction` and `NN2Prediction`. As `X, Factor`, we enter `Diagnosis` (the actual diagnosis for that tumor). The script is saved as `Performance Comparison`.

The report is shown in Figure 35. The Logistic and Neural Net models seem to outperform the Partition model. Both Neural Net models slightly outperform the logistic model. Of the two Neural Nets, given that they are equal in terms of misclassification rate on the validation set, our preference is to choose the simpler model. So, on this basis, we choose Neural Net Model 2 as our classifier.

Figure 35. Comparison of Four Models on Validation Data



Contingency Analysis of Most Likely Diagnosis By Diagnosis

Contingency Table

Diagnosis	Count / Total %	B	M	
		Most Likely Diagnosis		
M	Count	2	38	40
	Total %	1.83	34.86	36.70
B	Count	68	1	69
	Total %	62.39	0.92	63.30
	Count	70	39	109
	Total %	64.22	35.78	

Contingency Analysis of Partition Prediction By Diagnosis

Contingency Table

Diagnosis	Count / Total %	B	M	
		Partition Prediction		
M	Count	5	35	40
	Total %	4.59	32.11	36.70
B	Count	64	5	69
	Total %	58.72	4.59	63.30
	Count	69	40	109
	Total %	63.30	36.70	

Contingency Analysis of NN1 Prediction By Diagnosis

Contingency Table

Diagnosis	Count / Total %	B	M	
		NN1 Prediction		
M	Count	1	39	40
	Total %	0.92	35.78	36.70
B	Count	68	1	69
	Total %	62.39	0.92	63.30
	Count	69	40	109
	Total %	63.30	36.70	

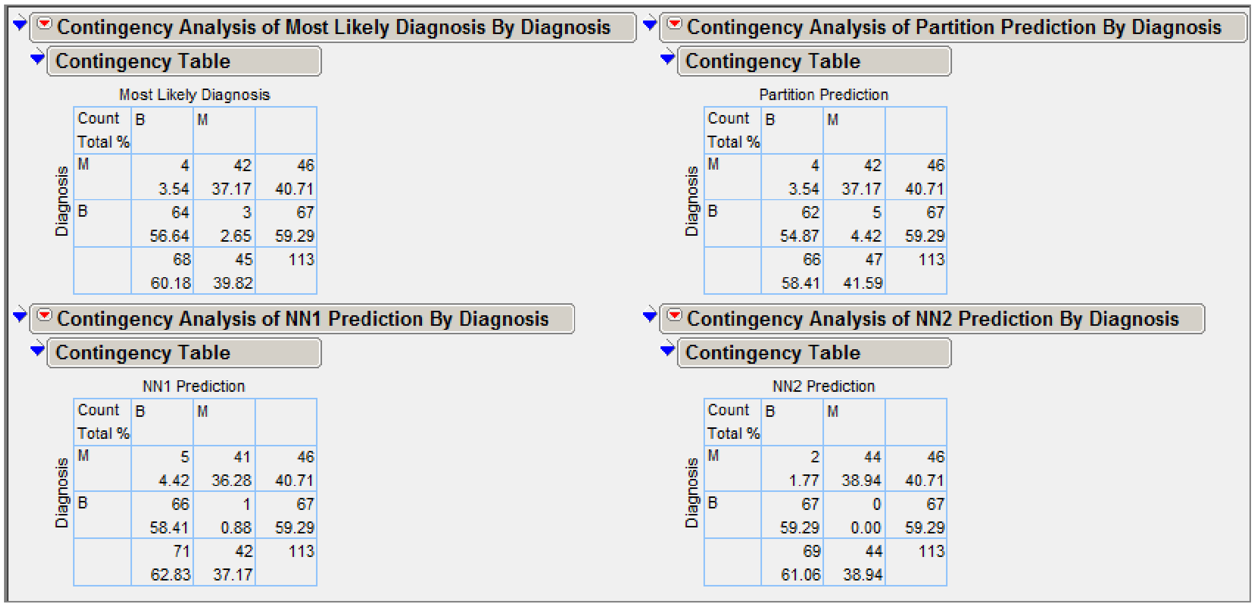
Contingency Analysis of NN2 Prediction By Diagnosis

Contingency Table

Diagnosis	Count / Total %	B	M	
		NN2 Prediction		
M	Count	1	39	40
	Total %	0.92	35.78	36.70
B	Count	68	1	69
	Total %	62.39	0.92	63.30
	Count	69	40	109
	Total %	63.30	36.70	

Now, to obtain a sense of how this model will perform on new, independent data, we apply it to our test set. Using the row state variable `Test Set`, we apply its row states to the data table in order to exclude all but the test set observations. We rerun the script `Performance Comparison` to see the performance of Neural Net Model 2 on the test data. Although the report (Figure 36) shows results for all four models, it is Neural Net Model 2 that is of primary interest at this point, as it is our chosen model. As hoped for, our chosen model outperforms the others on the test data.

Figure 36. Performance of Four Models on Test Data



Contingency Analysis of Most Likely Diagnosis By Diagnosis

Contingency Table

Diagnosis	Count / Total %	B	M	
		Most Likely Diagnosis		
M	Count	4	42	46
	Total %	3.54	37.17	40.71
B	Count	64	3	67
	Total %	56.64	2.65	59.29
	Count	68	45	113
	Total %	60.18	39.82	

Contingency Analysis of Partition Prediction By Diagnosis

Contingency Table

Diagnosis	Count / Total %	B	M	
		Partition Prediction		
M	Count	4	42	46
	Total %	3.54	37.17	40.71
B	Count	62	5	67
	Total %	54.87	4.42	59.29
	Count	66	47	113
	Total %	58.41	41.59	

Contingency Analysis of NN1 Prediction By Diagnosis

Contingency Table

Diagnosis	Count / Total %	B	M	
		NN1 Prediction		
M	Count	5	41	46
	Total %	4.42	36.28	40.71
B	Count	66	1	67
	Total %	58.41	0.88	59.29
	Count	71	42	113
	Total %	62.83	37.17	

Contingency Analysis of NN2 Prediction By Diagnosis

Contingency Table

Diagnosis	Count / Total %	B	M	
		NN2 Prediction		
M	Count	2	44	46
	Total %	1.77	38.94	40.71
B	Count	67	0	67
	Total %	59.29	0.00	59.29
	Count	69	44	113
	Total %	61.06	38.94	

CONCLUSION

The goal of this paper was to illustrate some of the features of JMP that support classification and data mining. We began by illustrating various visualization techniques that provide an understanding of the data and relationships among the variables. We partitioned our data into a training set, a validation set, and a test set. We then fit four models using the training data: a logistic model, a partition model, and two neural net models. The best classification, based on performance on the validation set, was obtained using a neural net model whose structure was chosen using K-fold crossvalidation.

We note that the logistic and neural net models had similar performance. Partition models tend not to perform as well as nonlinear (or linear) regression techniques when the predictors are continuous. They can be very useful, though, when there are categorical predictors, and especially when these have many levels. And, unlike neural net models and even logistic models, partition models are very intuitive and interpretable. In our situation, where classification was the primary goal, the interpretability of the model was less important than its ability to classify accurately.

We also wish to underscore the importance of guarding against overfitting, which, in the case of neural net models, often results in claims of exaggerated model performance. The application of K-fold crossvalidation helped us arrive at a neural model that was simple and that generalized well to our test set. Also, in the case of neural nets, where overfitting is so easy, it is important to assess the model's performance on an independent data set. Without this step, claims about model performance risk being exaggerated.

REFERENCES

Mangasarian, OL, Street, WN, and Wolberg, WH, “Breast Cancer Diagnosis and Prognosis via Linear Programming,” Mathematical Programming Technical Report 94-10, Dec. 19, 1994, pp. 1-9.

Street, WN, Wolberg, WH, and Mangasarian, OL, “Nuclear Feature Extraction for Breast Tumor Diagnosis,” 1993 International Symposium on Electronic Imaging: Science and Technology, Vol. 1905, 1993, pp. 861-870.

Bishop, CM, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

Ripley, BD, Pattern Recognition and Neural Networks, Cambridge University Press, 1996.

McCulloch, WS and Pitts, WH, “A Logical Calculus of the Ideas Immanent in Nervous Activity,” Bull. Math. Biophys. 5, 115-133, 1943.

ACKNOWLEDGMENTS

Mia Stephens and Phil Ramsey, both partners in the North Haven Group, made significant contributions to this paper. Chirstopher Gotwalt at JMP was extremely generous with details on JMP’s implementation of neural nets. Ian Cox, also at JMP, reviewed this paper with care and made many useful suggestions. Many thanks are due to each of these individuals.

RECOMMENDED READING

Berry, MJA and Linoff, GS, Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management, John Wiley and Sons, 1997.

Hastie, T, Tibshirani, R, and Friedman, J, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, 2001.

Haykin, S, Neural Networks: A Comprehensive Foundation, 2nd Edition, Prentice Hall, 1998.

Sall, JP and Maahs-Fladung, C, “Trees, Neural Nets, PLS, I-Optimal Designs and Other New JMP® Version 5 Novelties”, SUGI 27, http://www2.sas.com/proceedings/sugi27/p268-27.pdf.

CONTACT INFORMATION

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

Name	Marie Gaudard
Enterprise	North Haven Group
Address	PO Box 296
City, State ZIP	Hernando, FL 34442
Work Phone:	352-560-0312
Fax:	
E-mail:	mgaudard@northhavengroup.com
Web:	www.northhavengroup.com