RegressIt

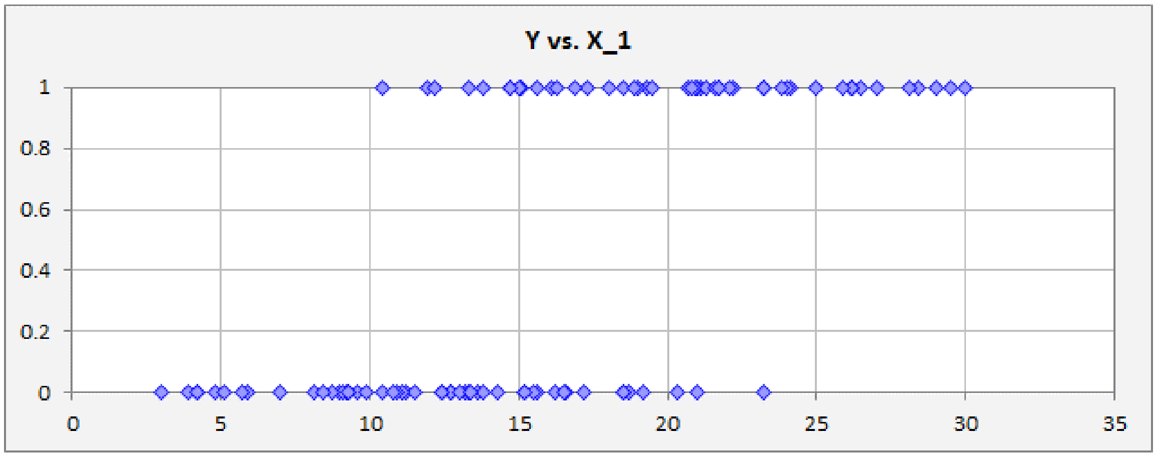
# Notes on logistic regression, illustrated with RegressItLogistic output[1]

In many important statistical prediction problems, the variable you want to predict does not vary continuously over some range, but instead is **binary**, that is, it has only one of two possible outcomes. For example:

- Will the web page visitor click or not click on the ad, given the search phrase?
- Will the citizen vote or not vote for the candidate, given the demographics?
- Will the debtor default or not default on the loan, given the credit history?
- Does the patient have or not have the disease, given the test results?
- Is the e-mail message spam or not, given its key words and attachments?

For the purposes of this discussion, let's suppose the values of the dependent variable are just coded as 1 and 0. In some settings what is needed is a **predicted probability** that it is 1 for given values of the independent variables. In other settings you want an actual **binary prediction**, i.e., to definitely predict 1 or 0 in each case, perhaps because the model is going to provide input to a decision problem in which which there are only two alternatives. If the model predicts 1, you'll go with the first alternative, and if it predicts 0, you'll go with the other. In the latter setting, what is commonly done is to first build a model that makes predictions in terms of probabilities, and then separately choose a "cutoff level" for the probability above which the predicted outcome is 1 and below which it is 0. For example, if the cutoff level is 0.6, you predict a definite value of 1 if the predicted probability is greater than or equal to 0.6. and predict a definite value of 0 otherwise. The choice of a cutoff level generally is based on the economic consequences of "type-I errors" (false positives) and "type-II errors" (false negatives). If a type-I error is much more costly, then you might set the cutoff to a value close to 1 rather than the default value of 0.5 that is often provided by your software.

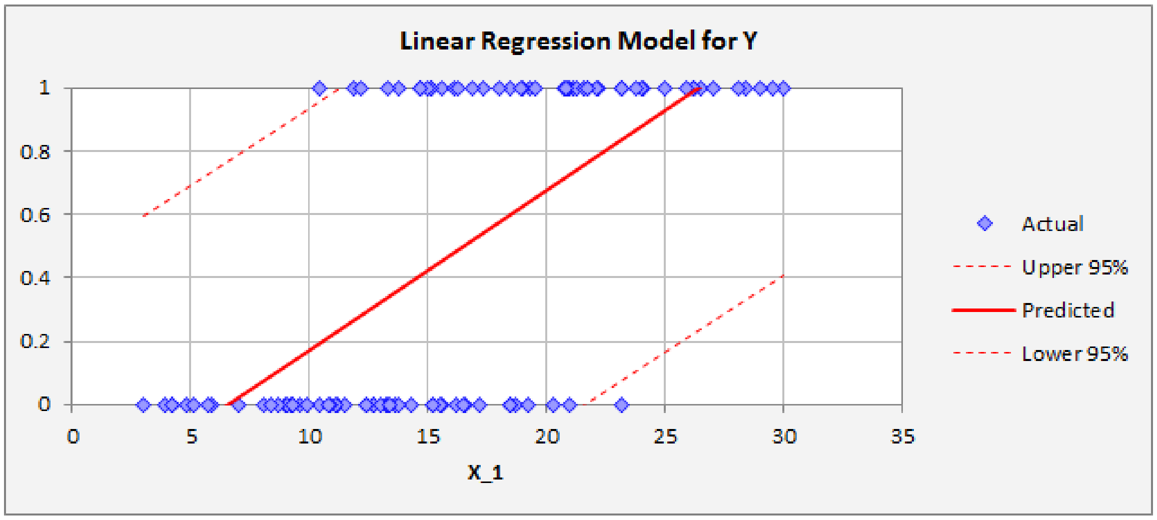
Let's start by considering the problem of how to make a prediction in the form of a probability, and consider the simplest case in which there is a single predictor. Let Y denote the binary variable to be predicted, and let X_1 denote the predictor variable, and suppose its possible values are continuously distributed over some range, say 0 to 30. If we draw a scatterplot of Y versus X_1, it might look like this:



[1] August 19, 2020. Visit https://regressit.com for the software, data, and complete documentation.

The values of Y are all clumped at the bottom and top, and in this case it appears that Y is morely likely to be 1 for large values of X_1 and more likely to be 0 for small values of X_1. So, how to predict the *probability* that it will be 1 for any given value of X_1? If you have a really huge amount of data that covers the whole range of X_1, you wouldn't necessarily need a fancy parametric model. If all you want to do is make predictions in units of probability, you could just break the X_1 range up into small bins, and compute the actual fraction of 1's in each bin, and let that fraction be your prediction for the probability that Y will be equal to 1 when the value of X_1 falls in that bin in the future. But in many cases your data is somewhat sparse and/or not evenly distributed over the X_1 range and/or you have some other reason for wanting to use a **line or smooth curve** rather than a set of bins for making predictions. These issues become even more important when there are many independent variables. For example, if you have 10 continuously-distributed independent variables and you break the range of each one up into 10 bins, then you actually have 10-to-the-10th (10 billion) bins in the space of all possible combinations of their values. It would take a lot of data to provide detail on all of them.

So, how to proceed? A naïve approach would be to use a **simple regression model** to predict Y from X_1, treating Y as if it was any old numeric variable and arbitrarily imposing a linearity assumption on its relationship with X_1. This approach has some problems, though, as are evident in the picture below. The data does not look anything like what we normally fit with a simple regression model. The way to think of the regression line in this model is not that it represents the line along with the data is most likely to fall, because the Y values cannot fall on the line—they are all 0's and 1's. Rather, the height of the line at any given value of X_1 represents the *predicted fraction of 1's* that will be obtained when X_1 has that value. So, what is wrong with this picture according to that re-interpretation of the regression line?



For one thing, there is no guarantee that the predicted probabilities and their lower and upper confidence limits will fall between 0 and 1 (and here they don't). You would therefore need to truncate them to keep the model from being completely illogical. For another, you might be especially interested in making accurate predictions of very large or very small probabilities that are associated with very large or small values of X_1, especially when dealing with rare events that have serious consequences for decisions. But a simple regression model would probably fit especially badly at the extreme ends of the X_1 range, as it does here. A linear regression model often fits best near the center of the multivariate data distribution and not as well out in the extremes, particularly if its assumptions (linear, additivity, normality…) are not exactly true.

Also, the linear regression model's confidence limits for predictions do not make sense here. They are based on an assumption that values of Y are normally distributed around the true regression line, and the standard deviation of that normal distribution is an additional model parameter to be estimated. With binary data. the error associated with a predicted probability of p can have only one of two possible values (p or 1-p).

The problems illustrated above are conceptually similar to the problems that arise in other linear regression applications in which the relationship between the dependent and independent variables is not linear and/or the errors of the model are not normally distributed. We've seen that in such cases a **nonlinear transformation** of the dependent variable (such as a log transformation in the case of beer sales data[2]) may make it possible to use a linear prediction equation after all. So, perhaps there is some nonlinear transformation of the probability scale that could be used here?

In fact, there are several possible nonlinear transformations of the probability scale that are in common use. The one you are probably most familiar is the **odds** scale. The **probability** of an event can equally be expressed in terms of the "**odds in its favor**". For example, if the probability is 75%, we say the odds are "3-to-1 in favor." If the probability is 99%, the odds are "99-to-1 in favor". For probabilities less than one-half (odds of 1-to-1) we sometimes flip the scale around and talk of "**odds against**". For example, a probability of 20% is often described as odds of "4 to 1 against", but we could also say that the odds are "¼ to 1 in favor". The mathematical formula for converting probabilities to odds-in-favor is this: **if the probability of an event is p, then the odds in its favor are "p/(1-p) to 1":**

$$\text{Probability} = p \quad \leftrightarrow \quad \text{odds in favor} = p/(1-p) \text{ to } 1.$$

Here is a table that shows the conversion between probability and odds:.

| Probability | Odds in favor |
|---:|---:|
| p | p/(1-p) to 1 |
| 0 | 0 |
| 0.001 | 1/999 to 1 |
| 0.01 | 1/99 to 1 |
| 0.1 | 1/9 to 1 |
| 0.2 | 1/4 to 1 |
| 0.25 | 1/3 to 1 |
| 0.3333 | 1/2 to 1 |
| **0.5** | **1 to 1** |
| 0.6666 | 2 to 1 |
| 0.75 | 3 to 1 |
| 0.8 | 4 to 1 |
| 0.9 | 9 to 1 |
| 0.99 | 99 to 1 |
| 0.999 | 999 to 1 |
| 1 | infinity |

[2] The beer sales example is discussed here: https://regressit.com/linear-regression.html

So, odds-in-favor is one possible nonlinear transformation of the probability scale that might be used for modeling. But it has its problems too as a candidate for use in a linear model. As you can see from the table, its scale is not symmetric: it's bounded below by zero and above by plus-infinity, with the center-point of the scale at odds-in-favor = 1, corresponding to a probability of ½.

OK, how about some further transformation of the probability scale? For use in linear regression, what we need is a transformation that can convert an **asymmetric scale of zero to infinity**, in which 1 is regarded as the center point, to a **symmetric scale from minus-infinity-to-plus infinity**, in which zero is regarded as the center point, which conveniently is the range of a standard normal distribution. The **logarithm** function provides such a transformation, and the version of it that we usually prefer is the **natural log,** which is the **LN** function in Excel. (It is the "base-*e*" log where *e* is the magic number 2.718…) So, suppose we apply the natural log function to the odds-in-favor function. This is the so-called **log odds scale**.

$$\text{Probability} = p \quad \leftrightarrow \quad \text{Log odds} = \text{LN}(p/(1\text{-}p))$$

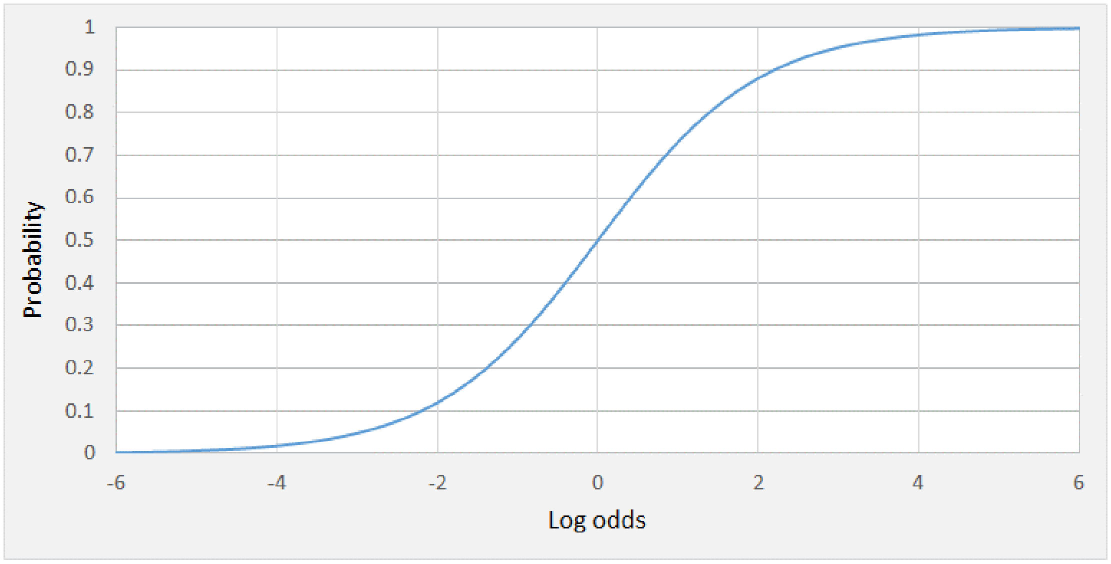
Here is a table of conversions between probability and log odds:

| Probability | Log odds |
|---:|---:|
| p | LN(p/(1-p)) |
| 0 | minus infinity |
| 0.001 | -6.91 |
| 0.01 | -4.60 |
| 0.1 | -2.20 |
| 0.2 | -1.39 |
| 0.25 | -1.10 |
| 0.3333 | -0.69 |
| **0.5** | **0.00** |
| 0.6666 | 0.69 |
| 0.75 | 1.10 |
| 0.8 | 1.39 |
| 0.9 | 2.20 |
| 0.99 | 4.60 |
| 0.999 | 6.91 |
| 1 | infinity |

The inverse transformation, from log odds back to probability, involves the exponential function $e^X$ , which is EXP(x) in Excel, and it looks like this:

$$\text{Log odds} = w \quad \leftrightarrow \quad \text{Probability} = \text{EXP}(w)/(1 + \text{EXP}(w)) = 1/(1 + \text{EXP}(-w))$$

The two formulas on the right side are equivalent. Sometimes you will see one and sometimes the other when reading books or articles about logistic regression. You don't have to memorize these formulas, but you should know what a plot of the probability function looks like. It is an S-shaped curve called a **logistic curve**, and it looks like this:



We now have a recipe for making a prediction in the form of a probability for the event Y=1 from a variable X that is measured on a continuous scale. First, use a linear equation to make a prediction for the log odds of Y=1:

$$\{\text{Predicted log odds of } Y=1\} \quad = \quad \beta_0 + \beta_1 X$$

If we also make the usual regression assumption that the errors in the predictions are i.i.d normally distributed, we can also put confidence bands around these predictions in the usual way, adding or subtracting some appropriate number of standard errors. (Remember that these are confidence limits for log odds, not bands around a region where raw data is expected to fall. They are what would be called "confidence limits for means" in linear regression analysis.)

Then we can translate the predicted log odds to a predicted probability:

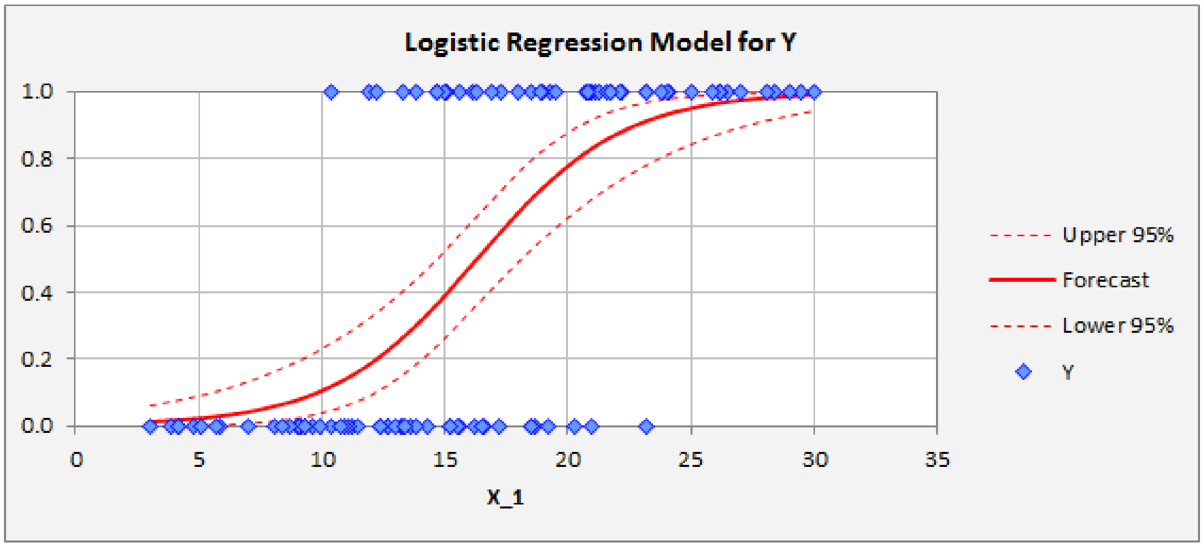
$$\{\text{Predicted probability of } Y=1\} \quad = \quad 1/(1 + \text{EXP}(-(\beta_0 + \beta_1 X)))$$

…and we can use the same formula to translate the lower and upper confidence limits for the predicted log odds into low and upper confidence limits for the predicted probabilities. Mathematically, the point predictions and lower and upper confidence limits that we get in this way must be bounded below and above by 0 and 1.

This is beginning to look a bit complicated, but the formula above follows by just chaining together a linear regression equation and a log-odds-to-probability conversion. *The formula is printed out for you at the top of the logistic regression output worksheet in RegressIt if you unhide the model equation.*

When you fit a logistic regression model to predict a binary variable Y from a continuous variable X, yielding predictions in units of probability, the last equation shown above is the actual one that is fitted. If you plot the predicted probability of Y=1 versus the value of X, the result is again a logistic curve, merely with the X-axis scaled according to the units of the independent variable and the values of $\beta_0$ and $\beta_1$.

The picture below shows what the logistic curve looks like for the model fitted to variables Y and X_1: much more reasonable! The predicted probabilities of Y=1 smoothly approach 0 or 1 as X_1 approaches its lower or upper bounds, and the confidence intervals also tighten as the predictions get close to 0 or 1.



Note: you don't always see the whole "S". In some cases the curve will only go partway to 0 or 1 before the independent variable hits its lower or upper bound in the data set. The curve may actually be an almost-straight line if the model does not make predictions that get very close to either 0 or 1.

There's just one minor technical problem. **We cannot fit the logistic regression equation to the data by applying ordinary linear regression to a transformed Y variable, as was done in the beer sales example**. In the beer model, we just converted the input data from units of dollars-per-case and cases-sold to units of log dollars and log cases, and fitted a straight line to the resulting graph. We cannot similarly apply a log-odds transformation to values of Y: it blows up if you plug in 0's or 1's!

So, how can we fit this curve to the given data? One approach would be to find the values of $\beta_0$ and $\beta_1$ that minimize the sum of squared errors in predicting the 0's and 1's from this nonlinear equation, which you could do with Solver in Excel, but the usual method is **"maximum likelihood estimation**". The details of this method are beyond the scope of this discussion, but the basic idea is this. For any hypothetical values of the coefficients $\beta_0$ and $\beta_1$, we can compute the predicted probability of the actual value of Y that was observed for each value of X. If it happened that Y = 1 for a given X, then the predicted probability of that Y was 1/(1 + EXP(-($\beta_0$ + $\beta_1$X))), and if it happened that Y = 0 for the given X, then the predicted probability of that Y was *1 minus* 1/(1 + EXP(-($\beta_0$ + $\beta_1$X))). Now, do this calculation for all the paired values of Y and X that are in the sample, and then *multiply all the predicted probabilities together*. The rationale for multiplying the probabilities together is that the events in the sample are assumed to be *statistically independent*, so the *product* of their predicted probabilities is the predicted *joint* probability of all of them happening together The resulting product (usually a very tiny number!) is called the "likelihood" of the observed data for the given values of $\beta_0$ and $\beta_1$. It seems reasonable that, in choosing values for $\beta_0$ and $\beta_1$ to plug into our equation that will be used for making predictions, we ought to try to find the values for which the likelihood of the data is as high as possible.

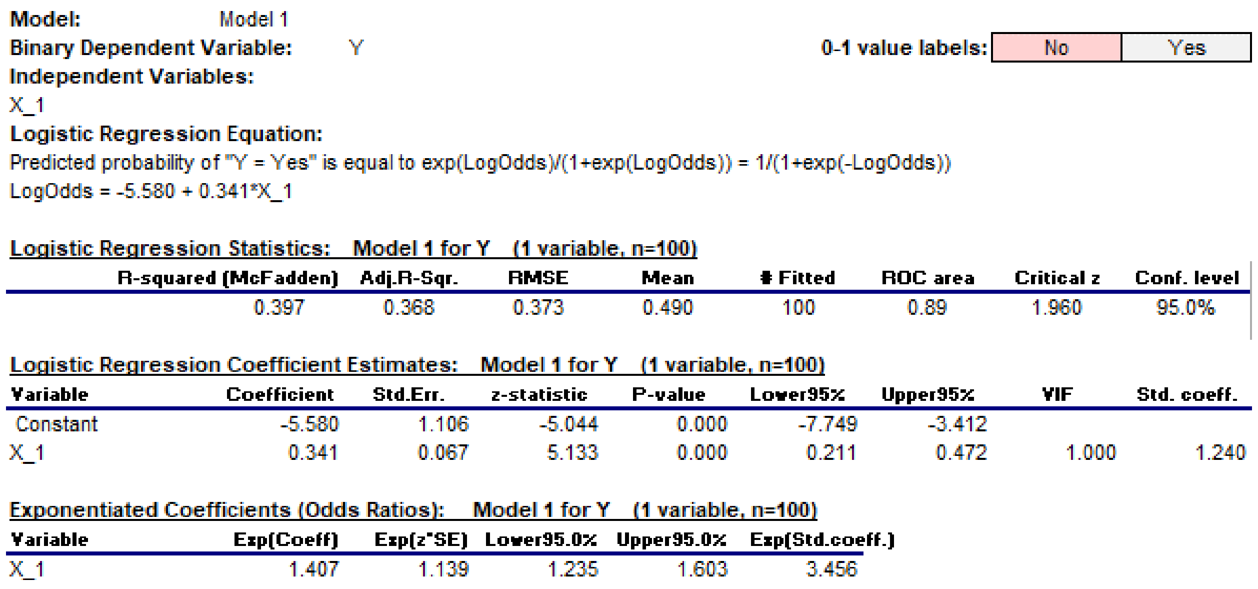
In maximum likelihood estimation, we let the computer find the exact optimal values of $\beta_0$ and $\beta_1$ according to this criterion. In principle you could use Solver in Excel to do this calculation, but somewhat different algorithms are used in practice in order to get additional outputs such as confidence intervals. Also, the objective that is actually maximized is not the likelihood function itself, but rather its logarithm, which is called the "log likelihood". This is mathematically equivalent to maximizing the unlogged value, because the log is an increasing smooth function, but the logged value is much more nicely behaved for purposes of optimization, and the quantity *minus*-log-likehood behaves very similarly to a *sum of squared errors* that you want to *minimize*. The output of a logistic regression model includes an **"analysis of deviance"** table which looks exactly like the analysis of variance table for a linear regression model, and it similarly decomposes the log likelihood function into explained and unexplained parts and applies a test for the explanatory power of the model as a whole.

Another important difference between a linear and logistic regression model is that the noise in the data comes from different sources. **In a linear regression model, the parameters of the linear equation determine the point prediction, but the model also has one additional parameter: the standard deviation of the noise (the model's true errors), of which the standard error of the regression is your estimate.** A regression model with a given set of coefficient estimates could either produce very narrow confidence limits or very wide confidence limits for predictions, depending on the standard deviation of the noise. (The width of confidence limits for predictions also depends to some extent on estimated errors in the coefficient estimates, but this effect is relatively small if the sample size is large.) The errors in the predictions of the true model are assumed to be drawn from the same normal distribution, so all confidence intervals for predictions from a linear regression model have roughly the same width, and they are symmetric around the point estimates. For example, the 95% confidence interval for a prediction is (approximately) the point estimate plus or minus two times the standard error of the regression.

In the logistic regression model, there is no additional parameter that measures the amount of intrinsic noise in the data. **The assumed noise in a logistic model consists entirely of the randomness in generating 0's and 1's from the probabilities that are determined by the true model equation.** In the computer output of a logistic regression model, you will see a root-mean-squared-error (RMSE) statistic reported in the same place where the standard error of the regression would be for a linear regression model. It is the same measure of prediction error within the sample, apart from the lack of a degree of freedom adjustment, but it does not represent an estimate of an additional parameter of the model. Also, it ought not to be larger than 0.5, which is the worst-case value that is obtained when exactly 50% of the values of Y are equal to 1 and the predictions are all equal to 0.5, i.e., maximally uninformative. In general, the amount by which the RMSE is less than 0.5 will depend on the extent to which the model makes non-trivial predictions that are close to 0 or 1. One more difference is that the distribution of errors for a logistic model is not the same for all predictions: it is asymmetric, because a prediction of p either has an error of p (if Y turns out to be 0) or it has an error of 1-p (if Y turns out to be equal to 1). **The distribution of errors of a logistic model therefore depends on the distribution of the predictions in the sample,** which (theoretically) is not true for a linear regression model when its parameters are accurately known. This has important implications for out-of-sample testing: **it is crucial to select test sets randomly so that the distributions of predictions in the training and test sets are roughly the same.**

The following pages show some of the output that RegressIt yields for the model fitted to the variables shown above. The summary table at the top looks almost exactly like it does for a linear regression model, except that RMSE appears in place of the standard error of the regression, the number that is reported as R-squared is a "pseudo R-squared", one of several possible definitions, and (very minor difference) the confidence intervals are based on the z rather than t distribution (because there isn't a good theoretical basis for a degree-of-freedom adjustment in this type of model).

Another important difference, when trying to compare this output to that of an ordinary regression model, is that the coefficients in the logistic model equation do not have a physical bang-for-the-buck meaning: **logistic regression coefficients are changes in predicted log odds (something that is rather hard to think about!) per unit of change in the dependent variable, other things being equal**. We can look to see whether the estimated values of the coefficients are significantly different from zero, but it is hard to attach meaning to their specific values. The optional outputs of this procedure also include **exponentiated coefficients**, which measure the effects of the independent variables on predictions expressed in units of odds rather than log-odds.



**Model:** Model 1
**Binary Dependent Variable:** Y    **0-1 value labels:** No | Yes
**Independent Variables:**
X_1
**Logistic Regression Equation:**
Predicted probability of "Y = Yes" is equal to exp(LogOdds)/(1+exp(LogOdds)) = 1/(1+exp(-LogOdds))
LogOdds = -5.580 + 0.341*X_1

**Logistic Regression Statistics: Model 1 for Y (1 variable, n=100)**

| R-squared (McFadden) | Adj.R-Sqr. | RMSE | Mean | # Fitted | ROC area | Critical z | Conf. level |
|---|---|---|---|---|---|---|---|
| 0.397 | 0.368 | 0.373 | 0.490 | 100 | 0.89 | 1.960 | 95.0% |

**Logistic Regression Coefficient Estimates: Model 1 for Y (1 variable, n=100)**

| Variable | Coefficient | Std.Err. | z-statistic | P-value | Lower95% | Upper95% | VIF | Std. coeff. |
|---|---|---|---|---|---|---|---|---|
| Constant | -5.580 | 1.106 | -5.044 | 0.000 | -7.749 | -3.412 | | |
| X_1 | 0.341 | 0.067 | 5.133 | 0.000 | 0.211 | 0.472 | 1.000 | 1.240 |

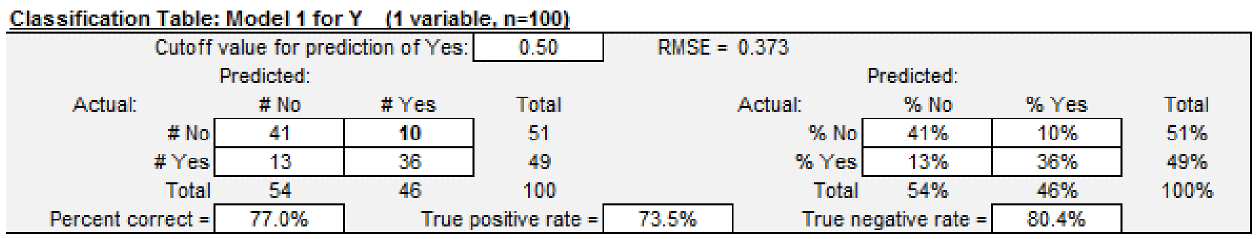
**Exponentiated Coefficients (Odds Ratios): Model 1 for Y (1 variable, n=100)**

| Variable | Exp(Coeff) | Exp(z*SE) | Lower95.0% | Upper95.0% | Exp(Std.coeff.) |
|---|---|---|---|---|---|
| X_1 | 1.407 | 1.139 | 1.235 | 1.603 | 3.456 |

The logistic regression prediction equation (which is given in terms of log odds) is shown at the top. For example, if X_1 = 20, then the predicted log odds of Y=1 is -5.580 + 0.341*20 = 1.249, and the predicted odds in favor of Y=1 is 3.487 to 1, where 3.487 = EXP(1.249). The corresponding predicted probability of Y=1 is 3.487/(1 + 3.487) = 0.777. Now, the exponentiated coefficient of X_1 is 1.407 (=EXP(0.341)). The interpretation of this number is that **it multiplies the predicted odds in favor of Y=1 when X_1 is increased by one unit.** So, if X_1 increases from 20 to 21, the predicted odds value is now 1.407 * 3.487 to 1, i.e., 4.907 to 1, which corresponds to a probability of 4.907/(1+4.907) = 0.831. Confidence limits for predicted probabilities can be determined by plugging confidence limits for coefficients and exponentiated coefficients into these same formulas.

The rest of the output for a logistic regression model is very specialized and quite different from linear regression output.

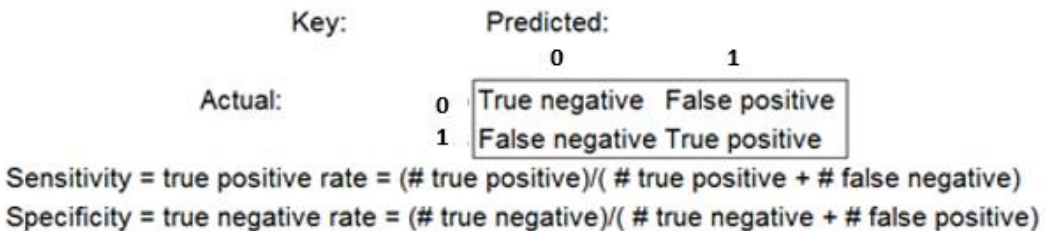
If the ultimate goal of the analysis is to make definitive **binary predictions** by applying a **cutoff level** to the predicted probabilities, then the **classification table** is of interest.[3] It shows, for a given cutoff level, how many of the actual 0's and 1's were correctly or incorrectly predicted. The table on the left shows classification results in terms of counts, while the table on the right shows them as percentages of the total. Because this sample has exactly 100 points, the numbers in both tables just happen to look the same in this case. **In RegressIt, the cutoff values in the classification table can be interactively adjusted on the worksheet after the model is fitted and you can watch all the numbers change.** The data and formula logic for this feature is contained within the worksheet, and it will work without RegressIt running, even if it is the only sheet in the file.



Classification Table: Model 1 for Y (1 variable, n=100)

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Cutoff value for prediction of Yes: | 0.50 | RMSE = 0.373 | | | | |
| | Predicted: | | | | | Predicted: | | |
| Actual: | # No | # Yes | Total | | Actual: | % No | % Yes | Total |
| # No | 41 | **10** | 51 | | % No | 41% | 10% | 51% |
| # Yes | 13 | 36 | 49 | | % Yes | 13% | 36% | 49% |
| Total | 54 | 46 | 100 | | Total | 54% | 46% | 100% |
| Percent correct = | 77.0% | | True positive rate = | 73.5% | | True negative rate = | 80.4% | |

The terminology for classification accuracy can get a little confusing. Here is a table that explains the cast of characters:

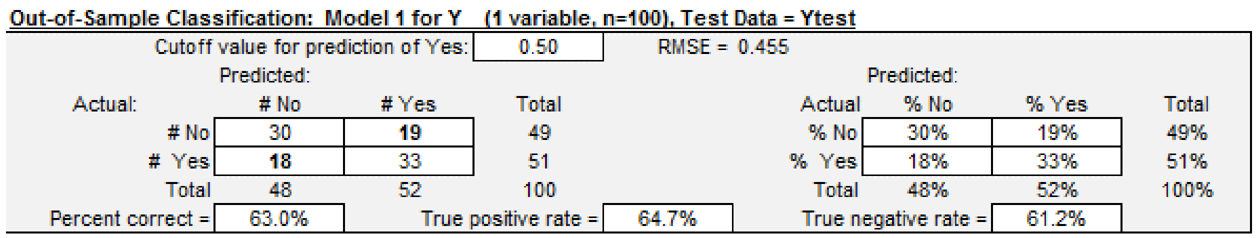


| Key: | | Predicted: | |
|---|---|---|---|
| | | 0 | 1 |
| Actual: | 0 | True negative | False positive |
| | 1 | False negative | True positive |

Sensitivity = true positive rate = (# true positive)/( # true positive + # false negative)

Specificity = true negative rate = (# true negative)/( # true negative + # false positive)

In data analytics in general it is good practice to hold out some data while going through the initial process of model identification and estimation and then test the fitted model on the out-of-sample data. RegressItLogistic includes a feature for doing this. To use it, make two copies of the dependent variable, one that includes only the sample to be used for estimation and another that includes the out-of-sample test data (or with or without the sample data). In this example the data set includes an additional 100 rows that were held out. An out-of-sample classification table has been produced, and its accuracy measures can be compared to those of the sample data.



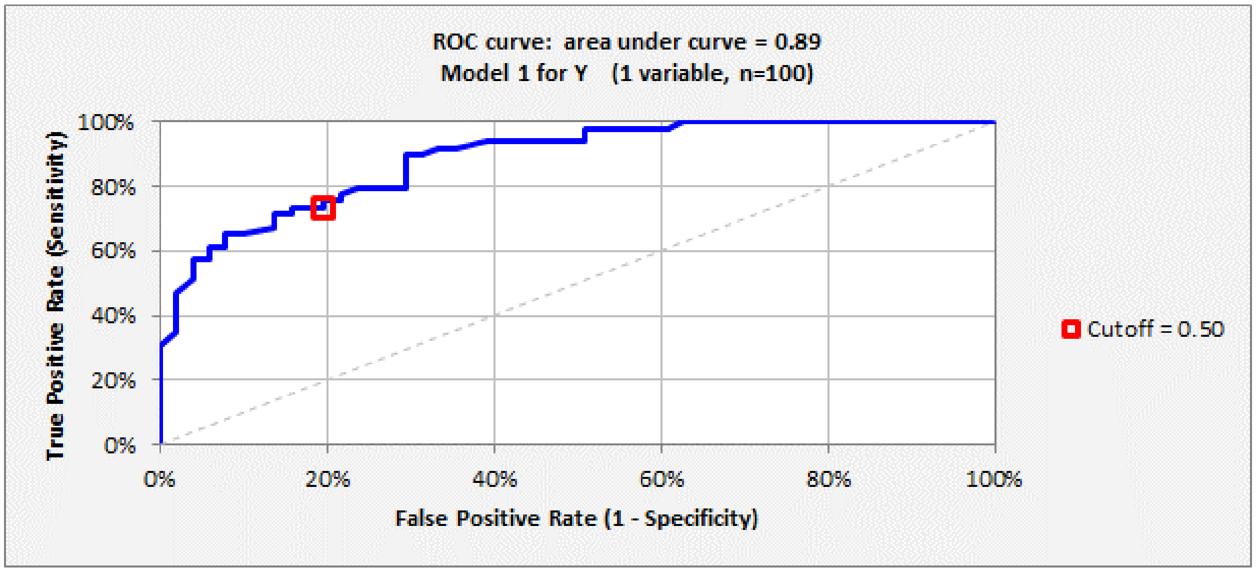
Out-of-Sample Classification: Model 1 for Y (1 variable, n=100), Test Data = Ytest

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Cutoff value for prediction of Yes: | 0.50 | RMSE = 0.455 | | | | |
| | Predicted: | | | | | Predicted: | | |
| Actual: | # No | # Yes | Total | | Actual | % No | % Yes | Total |
| # No | 30 | **19** | 49 | | % No | 30% | 19% | 49% |
| # Yes | **18** | 33 | 51 | | % Yes | 18% | 33% | 51% |
| Total | 48 | 52 | 100 | | Total | 48% | 52% | 100% |
| Percent correct = | 63.0% | | True positive rate = | 64.7% | | True negative rate = | 61.2% | |

[3] The classification table is sometimes given the cutesy name of "confusion matrix", although the model is not any more confused than other statistical prediction models for random variables. The errors for binary predictions just happen to have only two possible values: 0 ("guessed right") or 1 ("guessed wrong").

The model's out-of-sample performance is actually a little bit worse, both in terms of RMSE and classification accuracy. This seems like a reasonable thing to expect, although in general the relation between in-sample and out-of-sample classification accuracy will also depend on whether the two subsets of data are a priori equally representative of the population (e.g., whether they were randomly sampled). **The various error statistics and classification accuracy statistics in the output of a logistic regression model are not necessarily estimates of population parameters**. Rather they depend to some extent on the idiosyncracies of the sample. Some samples may have a lot of "hard" cases to classify, while others may contain a lot of "easy" cases, depending on the values of the independent variables. This is unlike the situation that exists in linear regression analysis, where (with sufficiently accurate parameter estimates) a model's prediction errors are expected to have the same normal distribution for any values of the independent variables.

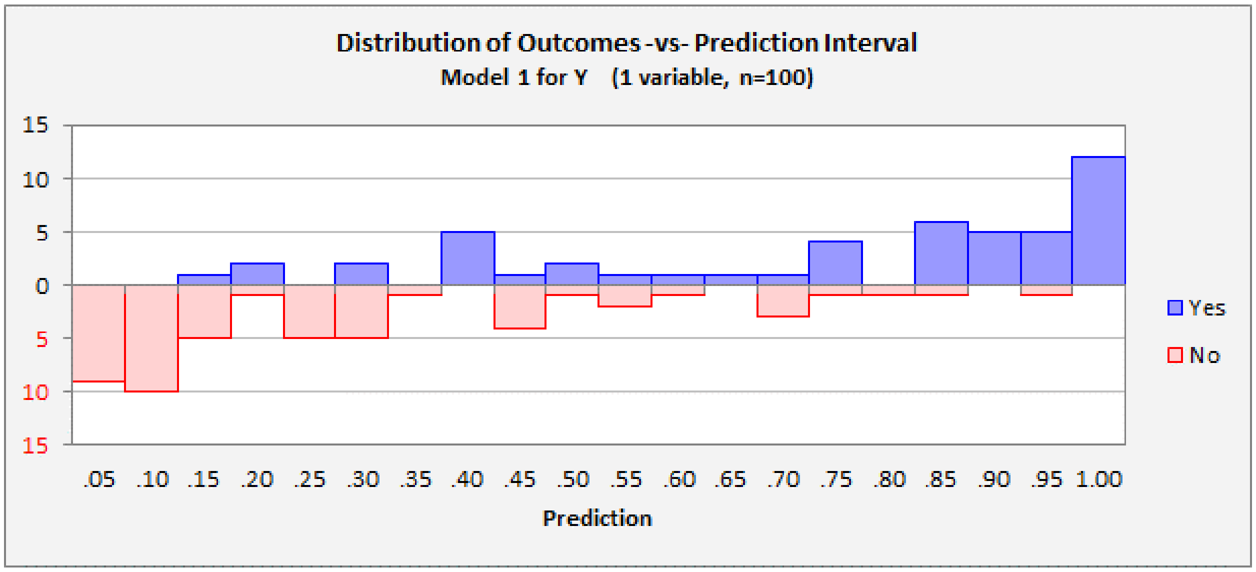
One standard piece of chart output for a logistic regression model is the **"Receiver Operating Characteristic" (ROC) curve. This is a plot of the true positive rate versus the false positive rate, and it shows the classification accuracy of the model over the full range of cutoff levels.** Theoretically this curve should lie above the diagonal, because you can achieve a diagonal line by using a constant-only model, although you may see examples in which a poor model yields a curve that dips below the diagonal somewhere. Ideally the curve approaches the upper left corner, meaning that there is some cutoff level that simultaneously achieves a true positive rate close to 100% and a false positive rate close to zero. The area under this curve is a commonly used measure of the overall predictive power of the model. *In RegressIt, the point on the curve determined by a chosen cutoff level is plotted on top of the curve, and you can interactively vary the cutoff level to see how it moves.* The result of doing this is often quite interesting!



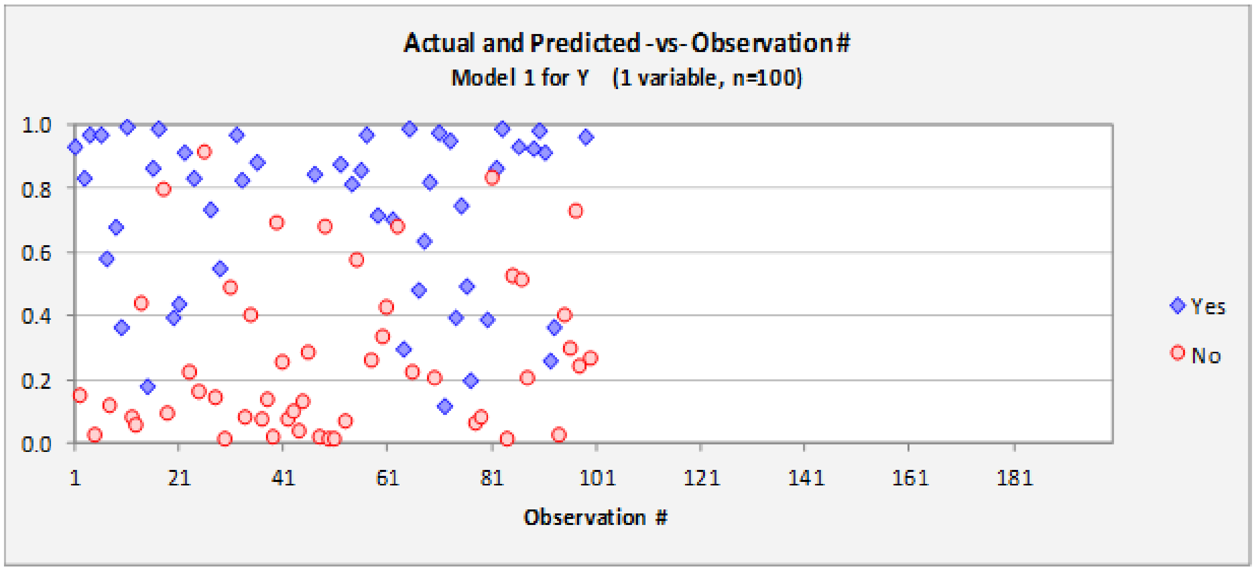
There is not a lot of consistency across software packages in terms of the selection and formatting of table and chart output that is provided for logistic regression models. RegressIt includes a number of other charts whose features can be interactively varied to explore and demonstrate other properties of the model. I*t is always a good idea to try to look at graphs in order visualize the assumptions of the model and the extent to which it does or does not fit the data. You never know what you might see!*

The four additional charts on the following pages provide more views of the distribution of prediction errors by size and location in the data set and the model's classification accuracy as a function of the cutoff level.

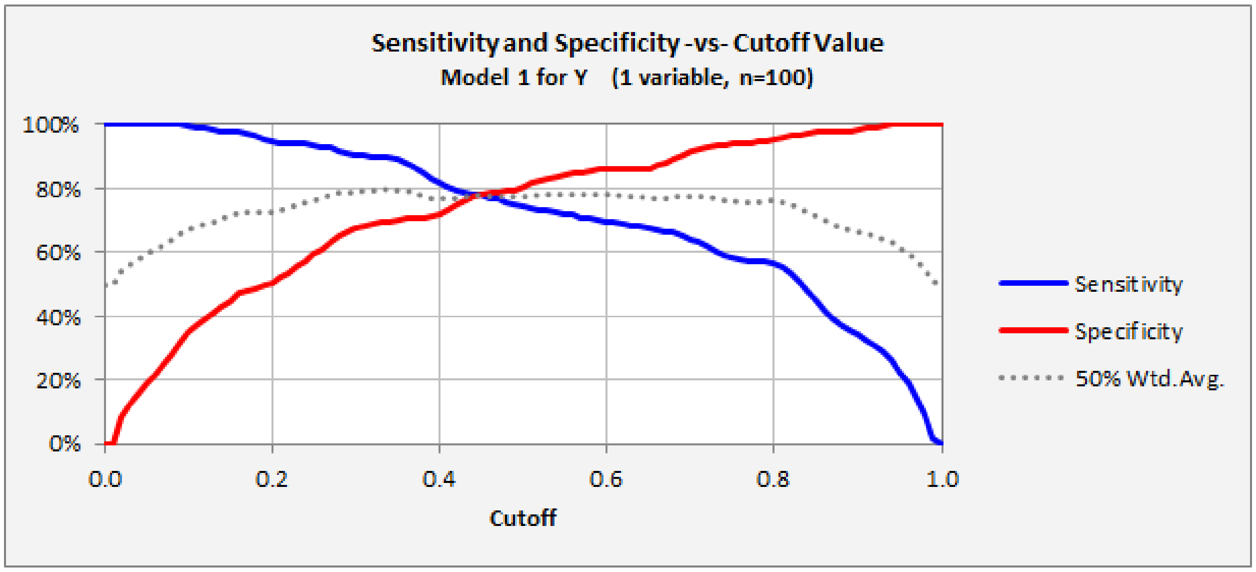
The first of these additional charts shows back to back histograms of the predictions that were made for the outcomes that turned out to be positive (blue bars) and the predictions that were made for the outcomes that turned out to be negative (red bars).

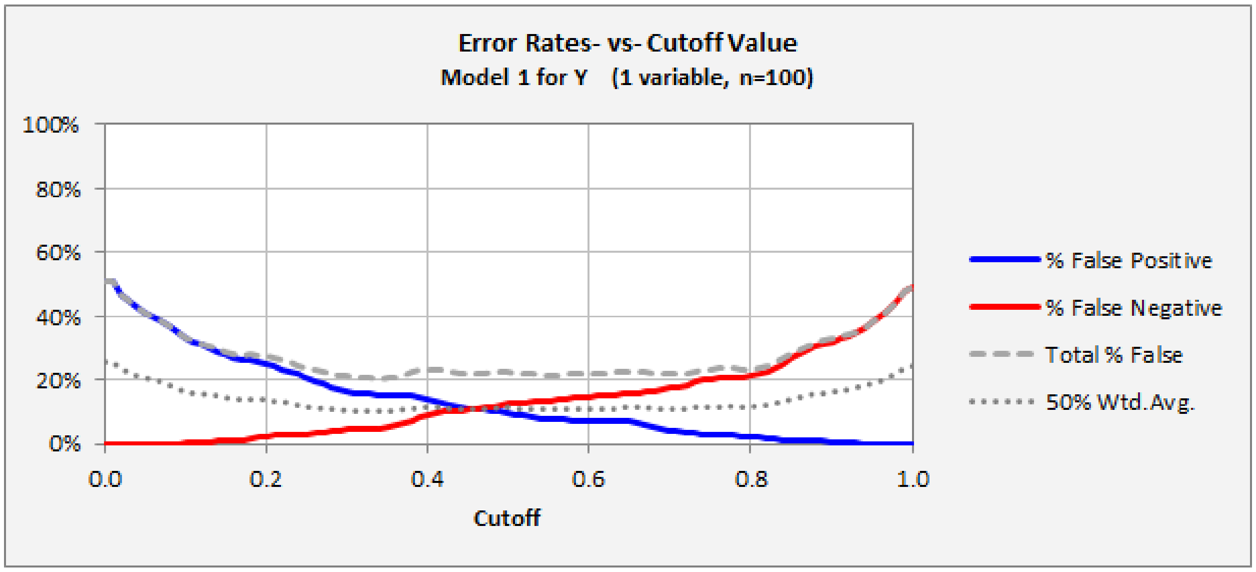


The second chart shows the predictions plotted versus row number with positive and negative outcomes identified by point color. The relation between these two charts is as follows: if you rotate the second chart 90 degress clockwise and let the red and blue point symbols fall down to the bottom and collect them in bins of width 0.05, you get the first chart.



The last two charts provide two other views of how the model's classification performance depends on the cutoff value. The first chart is a plot of true positive rate (sensitivity) and true negative rate (specificity) versus the cutoff level, and bigger is better. (This is the detail data on which the ROC chart is based.) The second chart shows the *false* positive and negative rates versus cutoff levels, and smaller is better. The gray dotted lines on these charts are weighted averages whose weight is interactively controlled by a spinner next to the chart. You can vary the weight to see where the weighted average is maximized or minimized. If the model is being developed for purposes of making binary decisions in which there are very different costs associated with false positive and false negative errors, this may help to dramatize the appropriate choice of a cutoff level.


Sensitivity and Specificity -vs- Cutoff Value
Model 1 for Y (1 variable, n=100)
Cutoff
Sensitivity
Specificity
50% Wtd.Avg.


Error Rates- vs- Cutoff Value
Model 1 for Y (1 variable, n=100)
Cutoff
% False Positive
% False Negative
Total % False
50% Wtd.Avg.