Topic 4
Logistic Regression and Classification
(Draft)

Instructor: Farid Alizadeh *

February 5, 2020

1 Classification with Probability Regression

We now try to extend the techniques of linear regression to classification problems. For simplicity, we assume that the response variable has only two levels (classes) which we encode as 0 and 1. We will discuss multi-level classification at the end.

The first question is: "Can we simply model such a response variable as a linear function of features?"

$$y = b_0 + b_1 x_1 + \cdots + b_d x_d$$

We need to estimate the parameters b_i.

There are several problems with this model:

1. y can be either 0 or 1, but the right hand side (RHS) of the model can in general be any real number (assuming that some variables X_i are quantitative). So there is no way to make the RHS directly spit out only 0 and 1 (or any number of discrete set of variables.)

2. We can instead replace y with $\mathbb{E}(y)$, the expected value of y. Remember that y as a binary random variable, so it can attain only values of 0 and 1. So its expected value is:

   $$\mathbb{E}(y) = \Pr[y = 0] \times 0 + \Pr[y = 1] \times 1 = \Pr[y = 1] = p$$

   So we could attempt to model instead of y its expected value which is the same as $p = \Pr[y = 1]$:

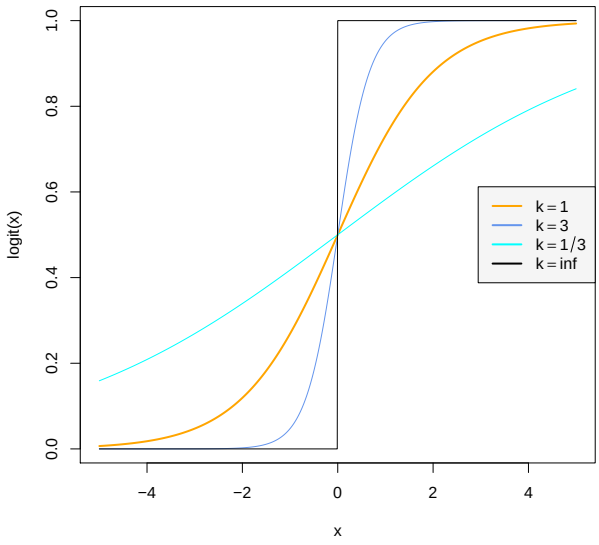
   $$p = \Pr[y = 1] = \mathbb{E}(y) = b_0 + b_1 x_1 + \cdots + b_d x_d$$

Still, this formulation does not resolve the issue either, since again in principle the right-hand-side can be any real number from $-\infty$ to $+\infty$, while $0 \leq \Pr[y = 1] \leq 1$.

3. We can make a transformation as follows:

$$p = \text{sigmoid}(x) = \frac{e^x}{1 + e^x} = \frac{1}{1 + e^{-x}}$$

where p is a probability and x any real number (negative or positive, and arbitrarily large or small.) This kind of transformation is called logistic function or sigmoid function. Here is the graph of sigmoid(x) and sigmoid$(3x)$:



Note that the logistic function is *S-shaped* and looks like a cumulative distribution function (CDF). It has the effect of mapping any real number (from $-\infty$ to $+\infty$ into the interval $[0, 1]$. Indeed, in principle, any CDF function can also be used to do such a mapping. For example, a common alternative is the CDF of the normal distribution. Any "S-shaped" function that maps the real line $(-\infty, \infty)$ to the interval $[0, 1]$ is called a *transfer function* or an *activation function* function.

Also note that in the graph of logistic(x) the transition from zero to one is slower than that of logistic$(3x)$, which has a sharper, more sudden transition from zero to one. Indeed for logistic(kx) as k gets larger the transition from 0 to 1 is sharper. At the extreme case of $k = \infty$, we get the function that $f(x) = 0$ if $x < 0$, and $f(x) = 1$ if $x > 1$. At $x = 0$ this function

is undefined (or is multi-valued and any number between 0 and 1 can be regarded as $f(0)$).

4. Conversely, we can write x as a function of p:

$$p = \frac{\exp(x)}{1+\exp(x)} \quad \Rightarrow \quad x = \ln\left(\frac{p}{1-p}\right)$$

Note the ratio $\frac{p}{1-p}$ is *odds* if the event (so it is a number between 0 and ∞). So x is the logarithm of the odds. Note also that the log of odds can be any number between $-\infty$ to $+\infty$. If log of odds is negative, it means $\frac{p}{1-p} < 0$, which means $p < 1/2$, so it is more likely that the event would not happen. If, on the other hand log of odds is positive, that is $\frac{p}{1-p} > 0$ then $p > 1/2$ and the probability of the event happening is more than it not happening.

In *logistic regression* we can now write the logarithm of odds *logodds* as a linear regression:

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1x_1 + \cdots + b_dx_d$$

where $p = \Pr[y = 1]$. Equivalently,

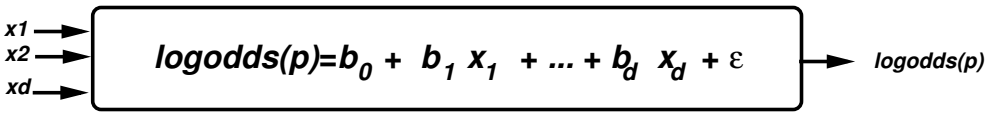
$$\Pr[y = 1] = \frac{\exp(b_0 + b_1x_1 + \cdots + b_dx_d)}{1+\exp(b_0 + b_1x_1 + \cdots + b_dx_d)}$$

Therefore, the log odds function, which is the inverse of the activation function is called the *link function*. So, in the case of logistic regression, the logistic function is the transfer function and the log odds function, also know as the *logit function* is the link function.

In summary, the regression model is now quite similar to regular linear regression on the surface. However, there is still a big difference.

2 Maximum Likelihood in Logistic Regression

Like the regular linear regression our model looks like this:



Unlike regular regression, though, it is not appropriate to assume that the error ϵ is normally distributed. In fact, even the assumption of homoscedastic errors (equal variance for all possible values of x_i) is not correct either. This will cause our approach to setting up the maximum likelihood method to estimating the coefficients b_i to be somewhat different than the normal error case.

First, let us define a general concept. The *general linear model (GLM)* is an extension of the ordinary linear model where the random (unexplained) error term ϵ in

$$y = b_0 + b_1 x_1 + \cdots + b_d x_d + \epsilon$$

can follow different types of probability distributions, and is not limited to following the normal distribution.

In the case of the logistic regression, the left hand side is log of odds. Here is how we can analyze the distribution of the error term. To make matters simple let us look at the special case

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x + \epsilon$$

In this case using Bayes rule if $p = \Pr[y = 1] > 1/2$ we set $y = 1$, and if $p < 1/2$ we set $y = 0$. Notice that if $p < 1/2$ if and only if the log of odds $\ln\left(\frac{p}{1-p}\right) < 0$, and similarly $p > 1/2$ if and only if $\ln\left(\frac{p}{1-p}\right) > 0$. Now we have

$$\Pr[y = 1|b_0, b_1] = p(y|b_0, b_1) = \frac{\exp(b_0 + b_1 x)}{1 + \exp(b_0 + b_1 x_1)} = \frac{1}{1 + \exp(-b_0 - b_1 x_1)}$$

From here we wish to build a *likelihood* function from the data. In general the density function of the binary random variable y is given by the *bernoulli distribution*:

$$p(y) = \Pr[y|b_0, b_1] = p^y (1-p)^{1-y}$$

where $p = \Pr[y = 1|b_0, b_1]$. This is a shorthand form of saying that $\Pr[y = 1] = p$ and $\Pr[y = 0] = 1 - p$. In short $p^y(1-p)^{1-y}$ is the density function of the binary random variable y, with p depending on parameters b_0, b_1.

Now let's build the *likelihood function* based on this density. Suppose our data is given in the following table:

y	x
1	3
0	2
1	5

In this case the contribution of the first line of the table to the likelihood function is

$$p(1|b_0, b_1) = \frac{\exp(b_0 + b_1 \times 3)}{1 + \exp(b_0 + b_1 \times 3)}$$

and the contribution of the second line is

$$p(0|b_0, b_1) = 1 - \frac{\exp(b_0 + b_1 \times 2)}{1 + \exp(b_0 + b_1 \times 2)} = \frac{1}{1 + \exp(b_0 + b_1 \times 2)}$$

So the likelihood function from the data can be computed as follows: For each row of data that $y_i = 1$ write $\frac{1}{1+\exp(-b_0 - b_1 x_i)}$ and for each row that

$y_i = 0$ write: $\frac{1}{1+\exp(b_0+b_1x_i)}$. The likelihood function is the product of all these terms from the rows of the data. This is so because the *joint distribution* of the y_i is the product of density of each of the y_i since they are assumed to be independent observations.

It will be more convenient to use the log likelihood function. After simplification, and now considering the general model $\ln\left(\frac{p}{1-p}\right) = b_0 + b_1x_1 + \cdots + b_dx_d$, and setting the vector $\mathbf{b} = (b_0, b_1, \ldots, b_d)$ we get the following formula form the log likelihood:

$$\log \text{Lik}(\mathbf{b}) = \sum_{i=1}^{N} y_i(b_0+b_1x_{i1}+\cdots+b_dx_{id}) - \sum_{i=1}^{N} \ln\Big(1+\exp\big(b_0+b_1x_{i1}+\cdots+b_dx_{id}\big)\Big)$$

For example, for the data above, the log likelihood function would be:

$$\begin{aligned}\log \text{Lik}(b_0, b_1) = \big(b_0 + 3b_1 + b_0 + 5b_1\big) - \\ \ln\Big(1 + \exp\big(b_0 + 3x_1\big)\Big) - \\ \ln\Big(1 + \exp\big(b_0 + 2x_1\big)\Big) - \\ \ln\Big(1 + \exp\big(b_0 + 5x_1\big)\Big)\end{aligned}$$

The maximum likelihood estimate of the parameters b_i is now calculated by maximizing this log likelihood.

In general, the likelihood of the logisitic regression with response values $y_i = \pm 1$ can be written as

$$\text{Lik}(\mathbf{b} \mid X) = \prod_{i=1}^{n} \frac{1}{1 + \exp\big(-y_i\mathbf{b}^\top \mathbf{x}_i\big)}$$

Here we use $y_i = -1$ instead of $y_i = 0$, for convenience. We also assume that the data X has a column (feature) of all ones and that is used for b_0 which is included in the vector $\mathbf{b}$ The log likelihood is given by

$$-\log \text{Lik}(\mathbf{b} \mid X) = \sum_{i=1}^{N} \log\Big(1 + \exp\big(-y_i\mathbf{b}^\top \mathbf{x}_i\big)\Big)$$

3 Details of the Optimization Process

To compute the maximum likelihood estimations for $\mathbf{b}$, most algorithms require the gradient of the log likelihood with respect to $\mathbf{b}$. Some algorithms (Newton's method) also require the Hessian (the matrix of 2nd partial derivatives) as well. Here are the gradient computations:

$$\mathbf{g}_\mathbf{b}^\top = \nabla_\mathbf{b}\big(-\log \text{lik}(\mathbf{b} \mid X)\big) = \sum_{i=1}^{N} \frac{y_i \exp\big(-y_i\mathbf{b}^\top \mathbf{x}_i\big)\mathbf{x}_i^\top}{1 + \exp\big(-y_i\mathbf{b}^\top \mathbf{x}_i\big)} = \sum_{i=1}^{N} \frac{-y_i\mathbf{x}_i^\top}{1 + \exp\big(y_i\mathbf{b}^\top \mathbf{x}_i\big)}$$

We need to set this set of $d+1$ ($b_i, i = 0, \ldots, d$) equations equal to zero and solve for b_i.

Note that unlike linear regression with normal errors, the derivatives of this function with respect to the parameters b_i are not linear functions of b_i. As a result setting these $d+1$ derivatives equal to zero and attempting to solve the resulting system of equations cannot be done by a simple formula. Instead we use an iterative procedure.

If we want to choose the Newton method, we need to compute the Hessian of the log likelihood function. Recall that the Hessian in this case is a $(d+1) \times (d+1)$ matrix whose i, j entry is the partial derivative $\frac{\partial^2 \log lik}{\partial b_i \partial b_j}$. Note that since the order of differentiation with (with respect to bi, b_j) does not matter the Hessian matrix is symmetric.

The Hessian matrix is given by:

$$\mathbf{H}_{\mathbf{b}} = \nabla^2_{\mathbf{b}} = \sum_{i=1}^{N} \frac{y_i^2 \exp(-\mathbf{b}^\top \mathbf{x}_i)\mathbf{x}_i\mathbf{x}_i^\top}{\left(1+\exp(-y_i \mathbf{b}^\top \mathbf{x}_i)\right)^2} = X^\top D_{\mathbf{b}} X \quad \text{with} \quad D_{\mathbf{b}} = \text{Diag}\left(\frac{\exp(y_i \mathbf{b}^\top \mathbf{x}_i)}{\left(1+\exp(-y_i \mathbf{b}^\top \mathbf{x}_i)\right)^2}\right)$$

Here $D_{\mathbf{b}}$ is a diagonal matrix as given above. Note that the Hessian is a positive definite matrix, and so the negative log likelihood function is convex.

Since the system of equations $\nabla_{\mathbf{b}}\left(\log \text{lik}(\mathbf{b} \mid X)\right) = 0$, is not linear, we must solve it by an *iterative process*. The general idea is as follows: We start with an initial guess to the maximum likelihood estimate $\mathbf{b}_{\text{ML}}$, say $\mathbf{b}^{(0)}$. Each iteration of the optimization algorithm finds from previous estimate $\mathbf{b}^{(k)}$ a new estimate $\mathbf{b}^{(k+1)}$. The sequence of estimates $\mathbf{b}^{(0)} \to \mathbf{b}^{(1)} \to \cdots \to \mathbf{b}^{(k)} \to \ldots$ converges to the optimal $\mathbf{b}_{\text{ML}}$.

For the *Gradient Descent (GD)* method the iteration is like this:

$$\mathbf{b}_{k+1} \leftarrow \mathbf{b}_k - \alpha_k \mathbf{g}_{\mathbf{b}_k}$$

So from the current estimate $\mathbf{b}_k$ we move in the direction of the negative of gradient of log likelihood function. The quantity α_k determines how far in that direction we go; it is called the *learning rate*. One approach is to choose it so that it minimizes the negative log likelihood in the direction of $-\mathbf{g}_{\mathbf{b}_k}$. In other cases a slower, less aggressive strategy for α_k is selected.

One problem with the approach given above is that if N, the number of data points, is huge say in the order of hundreds of thousands or millions or more, then evaluation of the log likelihood function and its gradient (and Hessian as below) becomes very expensive. In the *Stochastic Gradient Descent (SGD)* method, we sample a much smaller number n out of the N data points, and form the likelihood function and its gradient on these points alone. In each iteration, we choose a different random set of n points. The SGD method converges somewhat slower than GD method, but the amount of computations per iterations is much smaller. The SGD is one of the most popular methods for optimization used in machine learning. It is also used for large scale logistic regression.

Using *Newton method* requires the following iteration:

$$\mathbf{b}_{k+1} \leftarrow \mathbf{b}_k - \mathbf{H}_{\mathbf{b}_k}^{-1} \mathbf{g}_{\mathbf{b}_k}$$

This method converges to the optimal $\mathbf{b}_{ML}$ extremely fast especially when we are reasonably close. However, each iteration requires forming the $(d+1) \times (d+1)$ Hessian matrix, and then solving a linear system of equations, which costs $\mathcal{O}(d^3)$. For cases where there the number of features is large, the Newton method is impractical.

4 Prediction With Logistic Regression

Once the maximum likelihood estimate $\mathbf{b}_{ML} = (\hat{b}_0, \hat{b}_1, \ldots, \hat{b_d})$ is computed, we can use them to predict the class of new items from observing their x_i values. For new values of $x_{new} = (x_1, \ldots, x_d)$ we can predict y_{new} as follows.

First note that from the relation:

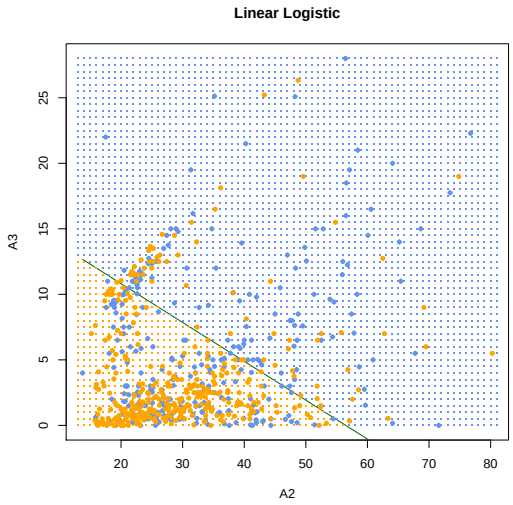
$$\hat{p} = \Pr[y_{new} = 1] = \frac{\exp(\hat{b}_0 + \hat{b}_1 x_1 + \cdots + \hat{b}_d x_d)}{1 + \exp(\hat{b}_0 + \hat{b}_1 x_1 + \cdots + \hat{b}_d x_d)}$$

and by simply plugging into the new values of $x_1, \ldots, x_d$ we can estimate the probability $\hat{p}$ that y belongs to class 1. Then, by Bayes decision rule, we classify $y = 1$ if $\hat{p} > 1/2$, and classify as $y = 0$ (or $y = -1$), otherwise.

Notice that $\hat{p} < 1/2$ then the odds $\frac{\hat{p}}{1-\hat{p}} < 1$, and $\ln\left(\frac{\hat{p}}{1-\hat{p}}\right) < 0$. Since this log odds equals $\hat{b}_0 + \hat{b}_2 x_1 + \cdots + \hat{b}_d x_d$, it follows that we can simply check this value and predict y as class 1 or 0, based on whether this is positive or negative:

Classify $y = 1$ if $\hat{b}_0 + \hat{b}_1 x_1 + \cdots + \hat{b}_d x_d > 0$ and $y = 0$ (or $y = -1$) otherwise.

As a result, the *hyperplane* $\hat{b}_0 + \hat{b}_1 x_1 + \cdots + b_d x_d = 0$ separates the 0 and the 1 classes. For instance, consider the German credit screen data from UCI archives. Using `A2` and `A3` as the only two (quantitative) features, we get the following classification:



As can be seen, the boundary between predicted reject (orange) and predicted accept (blue) areas is a line. This may seem too restrictive compared to non-parametric methods like kNN or CRT. We will see in the next lecture that it is possible to make logistic regression far more flexible, and allow it to have flexible "curved" boundaries.

Testing Effectiveness of the Logistic Models, The frequentest Approach

Once the coefficients are estimated by the maximum likelihood method it is possible to evaluate the effectiveness of the entire model, or individual variables, or a subset of variables. The technique used is the *likelihood ratio test* which we learned in general for maximum likelihood method in the previous lecture.

Here are the details. Suppose you are considering two models:

1. The smaller model uses only variables $x_1, \ldots, x_d$, and is called the *reduced model*:

   The reduced model:

   $$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x_1 + \cdots + b_d x_d$$

2. The larger models has all the variables in the smaller model, namely $x_1, \ldots, x_d$, *plus* another extra k variables $x_{d+1}, \ldots, x_{d+k}$, and is called the *the complete model:*

   The complete model:

   $$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x_1 + \cdots + b_d x_d + b_{d+1} x_{d+1} + \cdots + b_{d+k} x_{d+k}$$

The *maximum* log likelihood for the reduced model is

$$\log \text{Lik}(d) = \log \text{Lik}(b_0, b_1, \ldots, b_d),$$

and the maximum log likelihood for the complete model is

$$\log \text{Lik}(d+k) = \log \text{Lik}(b_0, b_1, \ldots, b_d, b_{d+1} \ldots, b_{d+k}).$$

Now consider the following hypothesis testing:

Null hypothesis: The extra variables $x_{d+1}, \ldots, x_{d+k}$ are not significant, and thus $b_{d+1} = \cdots = b_{d+k} = 0$,

Alternative hypothesis: at least one of extra variables are significant, that is at least one of $b_{d+1}, \ldots, b_{d+k}$ is not zero.

Define the likelihood ratio as:

$$\Lambda = \frac{\text{Lik}(b_0, \ldots, b_d)}{\text{Lik}(b_0, \ldots, b_d, b_{d+1} \ldots, b_{d+k})}$$

Then according to the likelihood ratio test, if the null hypothesis is true, then the quantity

$$-2\ln(\Lambda) = -2\big(\log \text{Lik}(b_0, \ldots, b_d) - \log \text{Lik}(b_0, \ldots, b_d, b_{d+1} \ldots, b_{d+k} \text{bigr}) \sim \chi^2(k)$$

that is follows the $\chi^2(k)$ distribution with k degrees of freedom. Note that this is an *approximate* result, and applies when the data is large enough.

For a model $\log \text{lik}(y) = b_0 + b_1 x_1 + \cdots + b_d x_d$ the quantity $-2 \log \text{Lik}(b_0, b_1, \ldots, b_d)$ is called the *deviance* of the model. So if the Null hypothesis is true, then the difference of deviance of the reduced and complete models should follow $\chi^2(k)$ distribution, and in particular it should hover around the mean of the $\chi^2(k)$ distribution, which is k. On the other hand, if the null hypothesis is false, then the difference of deviances will be much too large, and its p-value, the probability of getting such a $-2\ln(\Lambda)$ or larger will be too small. So if this p-value is small then we reject the null hypothesis and declare that at least one of the new variables is significant.

The process we just discussed is called *analysis of deviance*. It is the generalization of ANOVA (analysis of variance).

Example: Analysis of Deviance for the German Credit Card Data In the German credit approval example, we consider two models. One, relating `A16` to `A2` and `A3`. The other one will add `A8` and `A14`. Here are the results in R for the reduced model:

```
> summary(logitModel1)

Call:
glm(formula = A16 ~ A2 + A3, family = binomial, data = credit1)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0481  -1.0264  -0.8626   1.2192   1.5985

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.322043   0.242165  -5.459 4.78e-08 ***
A2           0.022752   0.007078   3.215  0.00131 **
A3           0.082751   0.017457   4.740 2.13e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 914.71  on 664  degrees of freedom
Residual deviance: 872.70  on 662  degrees of freedom
AIC: 878.7

Number of Fisher Scoring iterations: 4
```

And here are the results for the complete model:

```
> summary(logitModel2)

Call:
glm(formula = A16 ~ A2 + A3 + A8 + A14, family = binomial, data = credit1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4231  -0.9196  -0.7809   1.1606   1.8704

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.9886322  0.2760648  -3.581 0.000342 ***
A2           0.0030389  0.0077276   0.393 0.694132
A3           0.0568999  0.0183105   3.108 0.001887 **
A8           0.2848740  0.0422926   6.736 1.63e-11 ***
A14         -0.0008073  0.0005441  -1.484 0.137858
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 914.71  on 664  degrees of freedom
Residual deviance: 808.43  on 660  degrees of freedom
AIC: 818.43
```

```
Number of Fisher Scoring iterations: 5
```

The deviance is the item called `Residual deviance` in the summary output; it is 872.70 for the reduced model, and 808.43 for the complete one. We could test if the new variables `A8` and `A14` were significant as follows:

```
> pchisq(logitModel$deviance-logitModel2$deviance, df=2,lower.tail=F)
[1] 1.105093e-14
```

A more streamlined way is to run the function `anova` in R. Despite its inaccurate name, this function is capable of running the analysis if deviance:

```
> anova(logitModel1,logitModel2, test="LRT")
Analysis of Deviance Table

Model 1: A16 ~ A2 + A3
Model 2: A16 ~ A2 + A3 + A8 + A14
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1       662     872.70
2       660     808.43  2   64.273 1.105e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Note that they both produced a p-value of `1.105e-14` indicating that at least one of the two variables `A8` and `A14` is significant.

Assessing the Entire Logistic Model

It is also possible to use this method to test the significance of the entire model. In the hypothesis testing above if for the reduced model $d = 0$, that is none of the variables are used, then the only way to estimate y is by the proportion of items with a 1 under the y column. If this proportion is larger than $1/2$ then everything is classified as 1, otherwise everything is classified as 0. Suppose n_1 out of N observations are 1 in the data. Then we know that the log likelihood function is

$$-\log \text{Lik}(p) = -n_1 \ln(p) - (N - n_1)\ln(1-p).$$

This quantity is the `Null deviance`[1] reported in R. So the difference between `Null deviance` and `Residual deviance` follows approximately the $\chi^2(d)$ distribution if the Null hypothesis that says all d variables in the model are insignificant were to be correct. For the `logitModel2` above this is 914.71-808.43=106.29. This quantity is way too far from its expect value of 4, and it's p-value is 4.05e-22. So we reject the Null hypothesis and declare that at least one of the four variables `A2, A3, A8` and `A14` are significant.

[1] Notice also, that this `Null deviance` is exactly the cross entropy we learned earlier.

Assessing Each Feature Variable Individually

The likelihood ratio test can be applied to individual variables too. Suppose we wish to asses if variable x_i is significant. The reduced model here would be the one we get after removing x_i from the model. The complete model would be the one with x_i. The difference of the residual deviances would follow $\chi^2(1)$ distribution if x_i were insignificant in the model. For instance, in `logitModel2` the following test gives the significance of variable `A4`:

```
> anova(glm(A16~A2+A3+A8,family=binomial,data=credit1),
        logitModel2,test="LRT")
Analysis of Deviance Table

Model 1: A16 ~ A2 + A3 + A8
Model 2: A16 ~ A2 + A3 + A8 + A14
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       661     810.74
2       660     808.43  1   2.3128   0.1283
>
```

Since p-value is about 0.128 we cannot reject the null hypothesis that states `A14` is insignificant. Compare this p-value with the one printed next to `A4` row in the summary, which indicated a p-value of 0.138. The reason for discrepancy is that in the case of individual variables, there is another slightly more precise way to compute these p-values using standard normal distribution. In any case the two methods should produce approximately equal, but usually not exactly equal, p-values.

5 Handling Non-Binary Classification

When the response variable y is nonnumerical but has more than two possible values, the methods of logistic regression described above generalize naturally. However, in this case for each possible value of y we need to create a separate probability. For instance, suppose y can take values of "blue", "orange" and "green". Since there are three values, the logistic regression must somehow calculate the three probabilities:

$$\begin{aligned} p_B &= \Pr[y = \text{blue} \mid \text{data}] \\ p_O &= \Pr[y = \text{orange} \mid \text{data}] \\ p_G &= \Pr[y = \text{green} \mid \text{data}] \end{aligned}$$

Of course, these three quantities are *not* independent, since $p_B + p_O + p_G = 1$.

Under *multi-class logistic regression*, we come up with a different set of multipliers or coefficients for each class. Suppose the feature we are considering are $x_1, x_2, \ldots, x_d$. Then

For class "blue": $b_0^{(B)} + b_1^{(B)} x_1 + \cdots + b_d^{(B)} x_d$,
For class "orange": $b_0^{(O)} + b_1^{(O)} x_1 + \cdots + b_d^{(O)} x_d$,
For class "green": $b_0^{(G)} + b_1^{(G)} x_1 + \cdots + b_d^{(G)} x_d$.

These values are now translated to probabilities through a function known as the *softmax function*. In general, if we have K classes, then for a real vector $\mathbf{u} \in \mathbb{R}^K$ the *softmax* is function from $\mathbb{R}^K \to [0,1]^K$ and is given by K functions:

$$\text{softmax}_i(\mathbf{u}) = \frac{e^{u_i}}{\sum_j^K e^{u_j}}$$

This is a softening of the *max* function, which is defined from $\mathbb{R}^d \to \{0,1\}^K$ where $\max(\mathbf{x})$ is a 0-1 vector with exactly one 1 at the i^{th} position, if $x_i = \max(\mathbf{x})$.

For example, for the three color class suppose we have the following data:

Y	X
b	1
g	10
r	40
b	3
r	40
g	15
b	2
r	50
g	11
b	1
r	55
g	19

We are looking for three models (actually two because the probabilities add up to one.) For the blue class, $y_B = 1$ if the class is blue, and $y_B = 0$ otherwise. Similarly, for the green $y_G = 1$ if the class is green and $y_G = 0$ otherwise. And likewise for the red class. WE have the following models:

$$p_B = p[\text{blue} \mid \text{data}] = \frac{\exp(b_0^B + b_1^B x)}{\exp(b_0^B + b_1^B x) + \exp(b_0^G + b_1^G x) + \exp(b_0^R + b_1^R x)}$$

$$p_G = p[\text{green} \mid \text{data}] = \frac{\exp(b_0^G + b_1^G x)}{\exp(b_0^B + b_1^B x) + \exp(b_0^G + b_1^G x) + \exp(b_0^R + b_1^R x)}$$

$$p_R = p[\text{red} \mid \text{data}] = \frac{\exp(b_0^R + b_1^R x)}{\exp(b_0^B + b_1^B x) + \exp(b_0^G + b_1^G x) + \exp(b_0^R + b_1^R x)}$$

where the probabilities do add up to one. So we have three sets of b's, one for each class. Note that putting these together yields the softmax function.

Now, in the likelihood function each row of the data gives us the values of x and y_B, y_G and y_R. For instance, the likelihood contribution of the i^{th} line of data is

$$p_B^{y_{iB}} p_G^{y_{iG}} p_R^{y_{iR}}$$

where p_B, p_G and p_R are functions of their corresponding b_i. Of course only one of y_B, y_G, y_R is one and the other two are zero in each case. So, the contribution of the first line, since $y_{1B} = 1$ and $y_{1G} = y_{1R} = 0$, and $x_1 = 1$, is

$$p_B^{y_{1B}} = p_B = \frac{\exp\left(b_0^{(B)} + b_1^{(B)} \times 1\right)}{D}$$

where $D = \exp\left(b_0^{(B)} + b_1^{(B)} \times 1\right) + \exp\left(b_0^{(G)} + b_1^{(G)} \times 1\right) + \exp\left(b_0^{(R)} + b_1^{(R)} \times 1\right)$.

The likelihood function is the product of all these terms over all data rows. The log likelihood is then

$$\begin{aligned}\log \text{lik}\left(b_0^{(B)}, b_1^{(B)}, b_0^{(G)}, b_1^{(G)}, b_0^{(R)}, b_1^{(R)} \mid \text{data}\right) = \\ \sum_{i, y_{iB}=1} b_0^{(B)} + b_1^{(B)} x_i + \\ \sum_{i, y_{iG}=1} b_0^{(G)} + b_1^{(G)} x_i + \\ \sum_{i, y_{iR}=1} b_0^{(R)} + b_1^{(R)} x_i + \\ \sum_i \log\left(\exp\left(b_0^{(B)} + b_1^{(B)} x_i\right) + \exp\left(b_0^{(G)} + b_1^{G)} x_i\right) + \exp\left(b_0^{(R)} + b_1^{(R)} x_i\right)\right)\end{aligned}$$

More generally, if we have K classes and d b's for each class has to be calculated, then the log likelihood function is

$$\log \text{lik}(B \mid X) = \sum_{j=1}^{N} \sum_{i=1}^{K} y_{ji} \left(\mathbf{b}^{(i)}\right)^\top \mathbf{x}_j + \left(\sum_{j=1}^{N} \log\left(\sum_{i=1}^{K} \exp\left(\left(\mathbf{b}^{(i)}\right)^\top \mathbf{x}_i\right)\right)\right)$$

Here $y_{ji} = 1$ if the i^{th} data point is in class 'j', and is zero otherwise. So the inner sum $\sum_{i=1}^{K} y_{ji} \left(\mathbf{b}^{(i)}\right)^\top \mathbf{x}_j$ actually has only one nonzero term, corresponding to the class i that the j^{th} data points belongs to. Also, the $d \times K$ matrix B contains the model's parameters b_{ij}, the j^{th} column is the weight vector for class $\mathbf{b}^{(i)}$.

The optimization process is to maximize this log likelihood function with respect to the B matrix:

$$B_{ML} = \text{argmax}_B \left[\sum_{j=1}^{N} \sum_{i=1}^{K} y_{ji} \left(\mathbf{b}^{(i)}\right)^\top \mathbf{x}_j + \left(\sum_{j=1}^{N} \log\left(\sum_{i=1}^{K} \exp\left(\left(\mathbf{b}^{(i)}\right)^\top \mathbf{x}_i\right)\right)\right)\right]$$

This problem is also a convex optimization problem. and can be solved by the gradient methods, and similar approaches.