Chapter 4 in ISL: Generalized Linear Models, Logistic Regression, & Classification

19 March 2019

Generalized Linear Models

It should be noted that depending on the time frame that you may read about "GLMs", this may refer to two different types of modeling schemes

- GLM → General Linear Model: This is the older scheme that now refers to Linear Mixed Models or LMMs. This has more with correlated error terms and "random effects" in model.
- These days GLM refers to **Generalized Linear Model**. Developed by Nelder, John; Wedderburn, Robert (1972). This takes the concept of a linear model and generalizes it to response variables that do not have a normal distribution. **This is what GLM means today**. (I haven't run into an exception, but there might be ones out there depending on the discipline.)

Components of Linear Model

We will start with the idea of the regression model we have been working with the following model in different ways.

$$E\left[Y|\underline{X}=\underline{x}\right]=\mu(\underline{x})=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_px_p.$$

Here we are saying that the mean value of Y, $\mu(\underline{x})$, is determined by the values of our predictor variables.

For making predictions, we have to add in a ranom component (because no model is perfect).

$$Y=\mu(\underline{x})+\epsilon=\beta_0+\beta_1x_1+\beta_2x_2+\cdots+\beta_px_p+\epsilon.$$

We have two components here.

1. A deterministic component.
2. A random component: $\epsilon\sim N(0,\sigma^2)$.

The random component means that $Y\sim N(\mu(\underline{x}),\sigma^2)$.

The important part here is that the mean value of Y is assumed to be *linearly*.

Introduction to Link Functions

We have played around a bit with different functions we can try on Y if we do not get the proper distribution for the residuals.

$$g(Y) = \log(Y) \quad = \eta(\underline{x}) + \epsilon = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$$

What does this mean for Y itself?

$$\begin{aligned} Y &= g^{-1}\left(\eta(\underline{x}) + \epsilon\right) = e^{\eta(\underline{x})+\epsilon} \\ \mu(\underline{x}) &= e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_p x_p+\epsilon} \end{aligned}$$

- Another common option is $g(Y) = Y^p$ where it could be hoped that $p = 1$ but more often the best p is not...

These functions are used to get a linear relationship between $g(Y)$ and our predictor variables $x_1, \ldots, x_p$. They are supposed to be the **link** of Y with Normality and *linearity*.

Form of Generalized Linear Models

The form of Generalized Linear Models takes the basic form:

$$\begin{aligned} g\left(E\left[Y|\underline{X} = \underline{x}\right]\right) &= g\left(\mu(\underline{x})\right) \\ &= \eta(\underline{x}) \end{aligned}$$

There are now three components we consider:

1. A deterministic component.
2. A random component: $\epsilon \sim N(0, \sigma^2)$.
3. A link function $g(y)$ such that if $E\left[Y|\underline{X} = \underline{x}\right] = \mu(\underline{x})$, then

$$\begin{aligned} g\left(\mu(\underline{x})\right) &= \eta(\underline{x}) \\ &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p \end{aligned}$$

With GLMs, we are now paying attention to a few details.

- Y is not restricted to Normal distributions; that's the entire point...
- The link function is determined by the distribution of Y
- The distribution of Y determines the form of $E\left[Y|\underline{X} = \underline{x}\right]$ which we write as $\mu(\underline{x})$

Various Types of GLMs

Distribution of Y	**Support of Y**	**Typical Uses**	**Link Name**	**Link Function:** $g(y)$
Normal	$(-\infty, \infty)$	Linear response data	Identity	$g(\mu(\underline{x})) = \mu(\underline{x})$
Exponentail	$(0, \infty)$	Exponential Processes	Inverse	$g(\mu(\underline{x})) = \frac{1}{\mu(\underline{x})}$
Poisson	$0, 1, 2, \ldots$	Counting	Log	$g(\mu(\underline{x})) = \log(\underline{x})$
Binomial	$0, 1, 2, \ldots, n$	Classification	Logit	$g(\mu(\underline{x})) = \ln\left(\frac{\mu(\underline{x})}{1-\mu(\underline{x})}\right)$

We will mainly explore Logistic Regression, which uses the Logit function.

This is one of the ways of dealing with Classification.

First, Classification Problems, In General.

Classification can be simply define as determing what the outcome is for a discrete random variable.

- An online banking service must be able to determine whether or nnot a transaction being performed on the site is fraudulent, on the basis of the user's past tranaction history, balance, an other potential predictors. This a common use of statistical classification known as **Fraud detection**
- On the basis of DNA sequence data for a number of patients with and without a given disease a biologist would like to figure out which DNA mutations are disease-causing and which are not.
- A person arrives at the emergency room with a set of symptoms that possible be attributed tyo one of three medical conditions: *stroke, drug overdose, epilectic seizure*. We would want to choose or **classify** the person into one of the three categories.

In each of these situations what is the distribution of $Y =$ the category an indidual falls into?

An Example, Default

We will consider an example where we are interested in predicting whether an individual will default on his or her credit card payment, on the basis of annual income, monthly credit card balance and student status. (Did I pay my bills... Maybe I should check.)

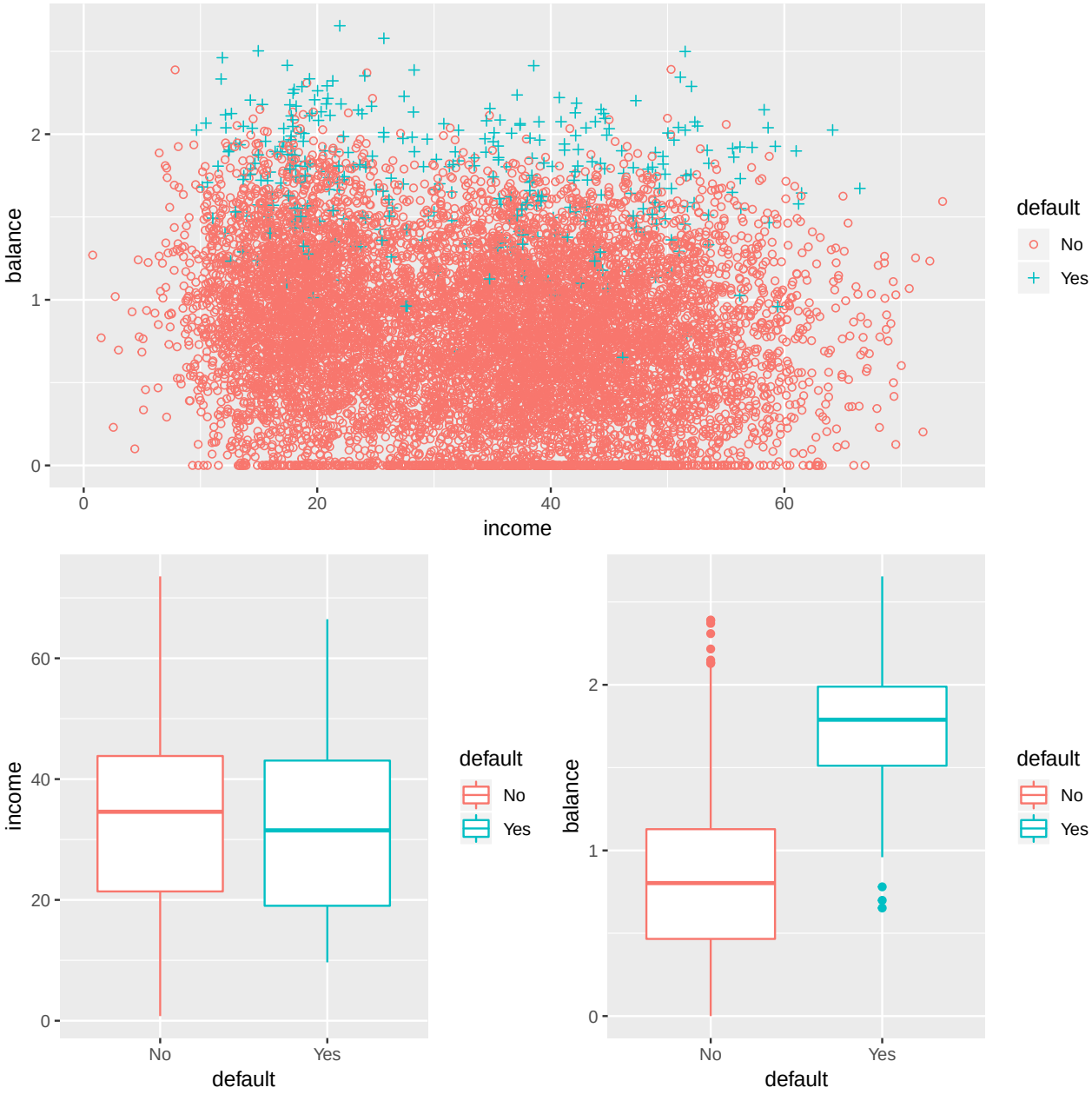
```
Default <- read.csv(file = "data/Default.csv")

# Converting Balance and Income to Thousands of dollars

Default$balance <- Default$balance/1000
Default$income <- Default$income/1000

head(Default)
```

```
##   X default student   balance    income
## 1 1      No      No 0.7295265 44.361625
## 2 2      No     Yes 0.8171804 12.106135
## 3 3      No      No 1.0735492 31.767139
## 4 4      No      No 0.5292506 35.704494
```

```
## 5 5      No      No 0.7856559 38.463496
## 6 6      No     Yes 0.9195885  7.491559
```

- `student` whether the person is a student or not.
- `default` is whether they defaulted
- `balance` is there total balance on their credit cards.
- `income` is the income of the individual.

A little bit of EDA



Try to describe what you see.

Logistic Regression

For each individual Y_i we are trying to model if that individual is going to default or not.

Hopefully you can see that there is not a clear division between those that default versus those that don't. So there is a chance that an individual will default, and a chance they won't that is going to be dependent on their balance, and maybe income.

We will start by modeling $P(\text{Default} \mid \text{Balance}) \equiv P(Y|X)$.

First, we we will code the default to a 1, 0 format to make modeling this probability compatible with the standard form of the with a special case of the Binomial(n, p) distribution where $n = 1$

$$P(Y|X) = p(X)^y(1 - p(X))^{1-y} \text{ where } y = 0, 1$$

Logistic regression creates a model for $P(Y = 1|X)$ which we will abbreviate as $p(X)$

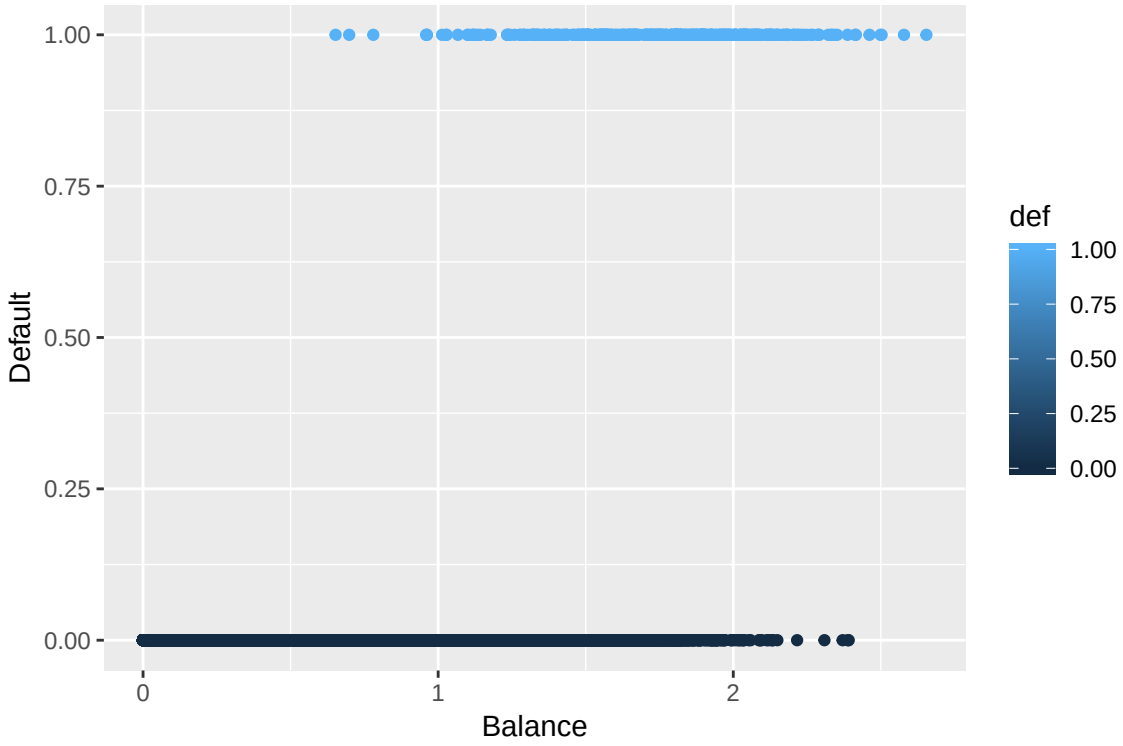
```
# Set an ifelse statement to handle the variable coding

#Create a new varible called def with the coded values
Default$def <- ifelse(Default$default == "Yes", 1,0)
```

Before we get into the actual Logistic Regression model, lets start with the idea that if we are trying to predict $p(X)$, when should we should we say that someone is going to default?

- A possiblity may be we classify the person as potentially defaulting if $p(X) > 0.5$
- Would it make sense to predict somone as a risk for defaulting at some other cutoff?

Plotting

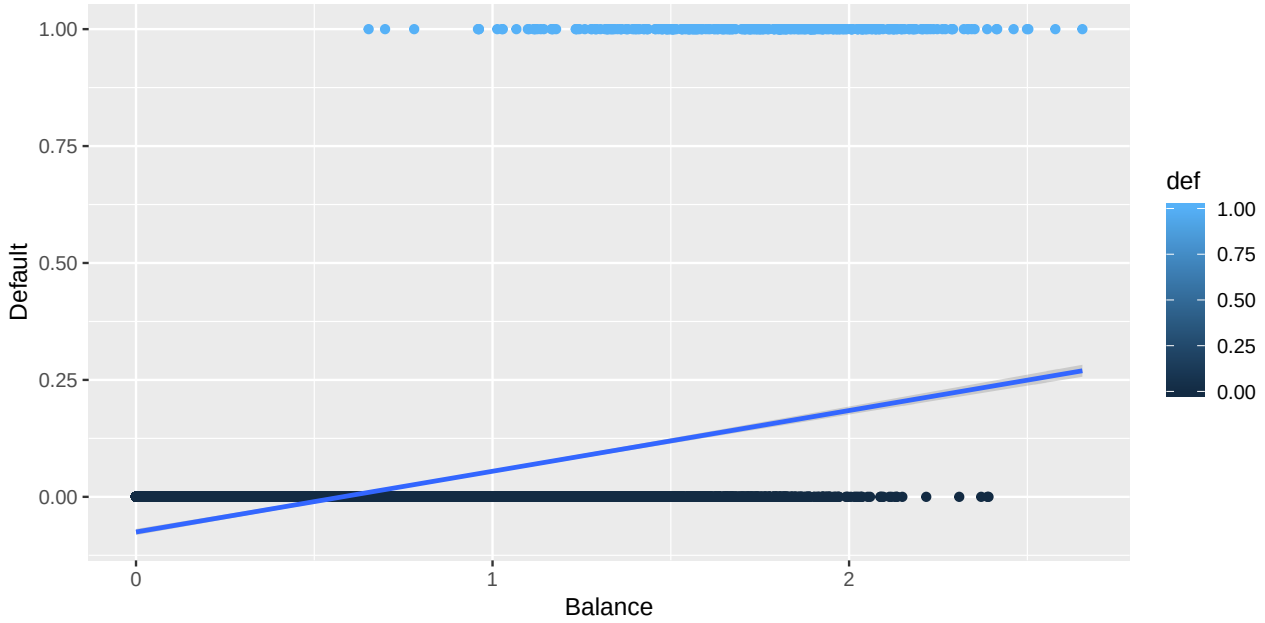
```
def.plot <- ggplot(Default, aes(x = balance, y = def, color = def)) + geom_point() +
  xlab("Balance") + ylab("Default")

def.plot
```



Why Not Linear Regression

We *could* use standard linear regression to model $p(\text{Balance}) = p(X)$. The line on this plot represents the estimated probability of defaulting.

```
def.plot.lm <- ggplot(Default, aes(x = balance, y = def, color = def)) + geom_point()+
  xlab("Balance") + ylab("Default") +
  geom_smooth(method = 'lm')

def.plot.lm
```

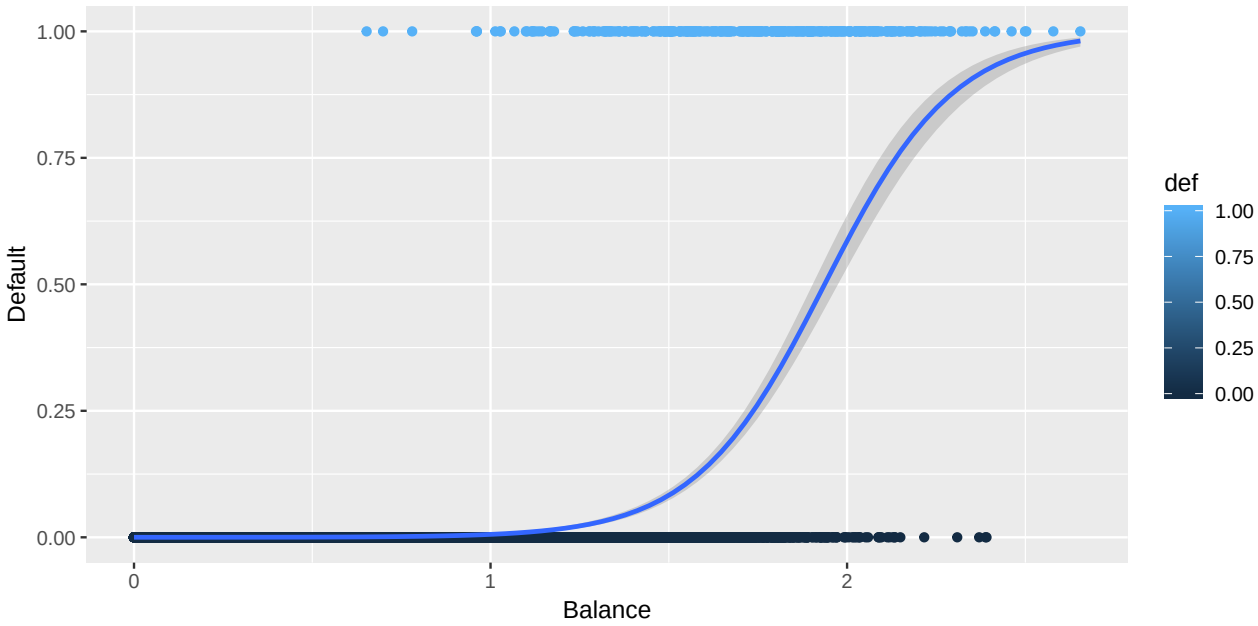


Can you think of any issues with this? Think about possible values of $p(X)$.

Plotting the Logistic Regression Curve

Here is the curve for a logistic regression.

```
def.plot.logit <- ggplot(Default, aes(x = balance, y = def, color = def)) + geom_point() +
  xlab("Balance") + ylab("Default") +
  geom_smooth(method = "glm", method.args = list(family = "binomial"))

def.plot.logit
```



How does this compare to the previous plot?

The Logistic Model in GLM

Even though we are trying to model a $p(X)$, we are trying to model a mean.

What is the mean or Expected Values of a binomial random variable $Y|X \sim \text{Binomial}(1, p(X))$?

Before we get into the link function, lets look at the function that models $p(X)$

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

The **link functions** for Logistic Regression is the **logit function** which is

$$\text{logit}(x) = \log\left(\frac{m}{1-m}\right) \text{ where } 0 < x < 1$$

Which for $E[Y|X] = \mu(X) = p(X)$ is

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

Notice that now we have a linear function of the coefficients.

Something to pay attention to: When we are interpreting the coefficients, we are talking about how the **log odds** change.

The coefficients tell us what the percentage increase of the odds ratio woud be.

$$\text{Odds} = \frac{p(X)}{1-p(X)}$$

Estimating The Coefficients: `glm` function

The function that is used for GLMs in R is the `glm` function.

`glm(formula, family = gaussian, data)`

- `formula`: the linear formula you are using when the link function is applied to $\mu(vX)$. This has same format as `lm`, e.g, `y ~ x`
- `family`: the distribution of Y which in logistic regression is `binomial`
- `data`: the dataframe... as has been the case before.

GLM Function on Default Data

```
default.fit <- glm(def ~ balance, family = binomial, data = Default)

summary(default.fit)
```

```
##
## Call:
## glm(formula = def ~ balance, family = binomial, data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2697  -0.1465  -0.0589  -0.0221   3.7589
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.6513     0.3612  -29.49   <2e-16 ***
## balance       5.4989     0.2204   24.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1596.5  on 9998  degrees of freedom
## AIC: 1600.5
##
## Number of Fisher Scoring iterations: 8
```

Default Predictions

"Predictions" in GLM models get a bit trickier than previously.

- We can predict in terms of the linear response, $g(\mu(x))$, e.g., log odds in logistic regression.
- Or we can predict in terms of the actual response variable, e.g., $p(x)$ in logistic regression.

$$\log\left(\frac{\widehat{p}(x)}{1-\widehat{p}(x)}\right) = -10.6513 + 5.4090x$$

Which may not be very useful in application.

$$\widehat{p}(x) = \frac{e^{-10.6513+5.4090x}}{1+e^{-10.6513+5.4090x}}$$

So for someone with a credict balance of 1000 dollars, we would predict that their probability of defaulting is

$$\widehat{p}(x) = \frac{e^{-10.6513+5.4090\cdot 1}}{1+e^{-10.6513+5.4090\cdot 1}} = 0.00526$$

And for someone with a credit balance of 2000 dollars, our prediction probability for defaulting is

$$\widehat{p}(x) = \frac{e^{-10.6513+5.4090\cdot 2}}{1+e^{-10.6513+5.4090\cdot 2}} = 0.54158$$

Predict Function for GLMS

We can compute our predictions by hand but, that's not super efficient. We can infact use the `predict` function on `glm` objects just like we did with `lm` objects.

With GLMs, our `predict` function now takes the form:

```
predict(glm.model, newdata, type)
```

- `glm.model` is the `glm` object you created to model the data.
- `newdata` is the data frame with values of your predictor variables you want predictions for. If no argument is given (or its incorrectly formatted) you will get the fitted values for your training data.
- `type` chooses which form of prediction you want.
  - `type = "link"` (the default option) gives the predictions for the link function, i.e, the linear function of the GLM. So for logistic regression it will spit out the predicted values of the log odds.
  - `type = "response"` gives the prediction in terms of your response variable. For Logistic Regression, this is the predicted probabilities.
  - `type = "terms"` returns a matrix giving the fitted values of each term in the model formula on the linear predictor scale.

Using `predict` on Default Data

Let's get predictions from the logistic regression model that's been created.

```
predict(default.fit, newdata = data.frame(balance = c(1,2)), type = "link")
```

```
##          1          2
## -5.1524137  0.3465032
```

```
predict(default.fit, newdata = data.frame(balance = c(1,2)), type = "response")
```

```
##           1           2
## 0.005752145 0.585769370
```

Classifying Predictions

If we are looking at the log odds ratio, we will classify an observation as a $\widehat{Y} = 1$ if the log odds is non-negative.

```
Default$pred.link <- predict(default.fit)
Default$pred.class1 <-as.factor(ifelse(Default$pred.link >=0, 1, 0))
```

Or we can classify based off the predicted probabilities, i.e., $\widehat{Y} = 1$ if $\widehat{p}(x) \geq 0.5$.

```
Default$pred.response <- predict(default.fit, type = "response")
Default$pred.class2 <-as.factor(ifelse(Default$pred.response >=0.5, 1, 0))
```

Which will produce identical classifications.

```
sum(Default$pred.class1 != Default$pred.class2)
```

```
## [1] 0
```

Assessing Model Accuracy

We can assess the accuracy of the model using a confusion matrix using the `confusionMatrix` function, which is part of the `caret` package.

- Note that this will require you to install the `e1071` package (which I can't fathom why they would do it this way...).

```
confusionMatrix(Predicted, Actual)
```

```
library(caret)

confusionMatrix(Default$pred.class1, as.factor(Default$def))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9625  233
##          1   42  100
##
##                Accuracy : 0.9725
##                  95% CI : (0.9691, 0.9756)
##     No Information Rate : 0.9667
##     P-Value [Acc > NIR] : 0.0004973
##
##                   Kappa : 0.4093
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9957
##             Specificity : 0.3003
##          Pos Pred Value : 0.9764
##          Neg Pred Value : 0.7042
##              Prevalence : 0.9667
##          Detection Rate : 0.9625
##    Detection Prevalence : 0.9858
##       Balanced Accuracy : 0.6480
##
##        'Positive' Class : 0
##
```

Categorical Predictors

Having categorical variables and including multiple predictors is pretty easy. It's just like with standard linear regression.

Let's look at the model that just uses the `student` variable to predict the probability of defaulting.

```
default.fit2 <- glm(def ~ student, family = binomial, data = Default)

summary(default.fit2)
```

```
##
## Call:
## glm(formula = def ~ student, family = binomial, data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.2970  -0.2970  -0.2434  -0.2434   2.6585
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.50413    0.07071  -49.55  < 2e-16 ***
## studentYes   0.40489    0.11502    3.52 0.000431 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 2908.7  on 9998  degrees of freedom
## AIC: 2912.7
##
## Number of Fisher Scoring iterations: 6
```

```
predict(default.fit2, newdata = data.frame(student = c("Yes", "No")), type = "response")
```

```
##          1          2
## 0.04313859 0.02919501
```

Would student status by itself be useful for classifying individuals as defaulting or no?

Multiple Predictors

Similarly (Why isn't there an 'i' in that?) to categorical predictors we can easily include multiple predictor variables. Why? Because with the link function, we are just doing linear regression.

$$p(\underline{X}) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2 \cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\beta_2 X_2 \cdots+\beta_p X_p}}$$

The **link functions** is still the same.

$$\text{logit}(m) = \log\left(\frac{m}{1-m}\right) \text{ where } 0 < x < 1$$

Which for $E\left[Y|\underline{X}\right] = \mu(\underline{X}) = p(\underline{X})$ is

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 \cdots + \beta_p X_p$$

Multiple Predictors in Default Data

Lets look at how the model does with using `balance`, `income`, and `student` as the predictor variables.

```
default.fit3 <- glm(def ~ balance + income + student, family = binomial, data = Default)

summary(default.fit3)
```

```
##
## Call:
## glm(formula = def ~ balance + income + student, family = binomial,
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4691  -0.1418  -0.0557  -0.0203   3.7383
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.869045   0.492256 -22.080  < 2e-16 ***
## balance       5.736505   0.231895  24.738  < 2e-16 ***
## income        0.003033   0.008203   0.370  0.71152
## studentYes   -0.646776   0.236253  -2.738  0.00619 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1571.5  on 9996  degrees of freedom
## AIC: 1579.5
##
## Number of Fisher Scoring iterations: 8
```

```
predict(default.fit3, newdata = data.frame(balance = c(1,2), student = c("Yes", "No"), income=40), type
```

```
##           1           2
## 0.003477432 0.673773774
```

```
predict(default.fit3, newdata = data.frame(balance = c(1), student = c("Yes", "No"), income=c(10, 50)),
```

```
##           1           2
## 0.003175906 0.006821253
```