CHAPTER 13

GENERALIZED LINEAR MODELS

13.1 AN OVERVIEW OF GENERAL LINEAR MODELS

In Chapter 11, the linear regression models we examined each had a continuous response variable. However, what happens if we want to build a regression model for a binary response instead? Or for a numeric discrete response? Luckily, there is a family of linear models that includes all three cases – continuous, numeric discrete, and binary – of regression response variables: *General Linear Models* (GLMs).

To explain how regression for three different kinds of responses can be related, we will briefly take another look at the parametric regression equations for each case. Once we establish how they are related, we will then use their descriptive versions, just as we did in Chapter 11.

Recall the parametric model for multiple regression, given here.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

The sum $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$ is called the *linear predictor*. For brevity, we will write the linear predictor as $\boldsymbol{X\beta}$. The formula that connects the linear predictor to the mean μ of the y variable at a set of given values of predictor variables is called the *link function*, $g(\mu)$.

Different link functions entail different regression models, with each link function associated with a particular response type. For each different response type we discuss, we will specify a particular $g(\mu)$ so that $X\beta = g(\mu)$, and solve for μ to obtain the final form of the model.

We begin by demonstrating how GLMs work by showing how linear regression can be expressed as a GLM. We then demonstrate two new regression models as GLMs: logistic regression and Poisson regression.

Data Science Using Python and R, First Edition. Chantal D. Larose and Daniel T. Larose.

13.2 LINEAR REGRESSION AS A GENERAL LINEAR MODEL

When the response variable has a Normal distribution at each set of given predictor variable values, then we are back in the realm of linear regression. In this realm, the link function is simply the identity function, where

$$g(\mu) = \mu$$

Setting $\boldsymbol{X\beta}$ equal to $g(\mu)$ using this identity link gives us $\boldsymbol{X\beta} = g(\mu) = \mu$.

Once we have a functional relationship between $\boldsymbol{X\beta}$ and μ, we can work backwards to obtain the final form of the regression model by expanding our abbreviated notation. Doing so gives us the population equation for linear regression:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \varepsilon$$

from which we can obtain the descriptive form

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p$$

which we worked with in Chapter 11.

13.3 LOGISTIC REGRESSION AS A GENERAL LINEAR MODEL

Next, suppose we are trying to predict a binary response, such as whether or not a customer has a store credit card. In this case, the distribution of our response variable will be binary: 1 or 0, indicating a Yes or No.

The link function for a binary response variable is $g(\mu) = \ln\left(\frac{\mu}{1-\mu}\right)$. We set this function equal to our linear predictor $\boldsymbol{X\beta}$ to obtain

$$\boldsymbol{X\beta} = \ln\left(\frac{\mu}{1-\mu}\right)$$

To isolate μ, we use the fact that $e^{\ln(x)} = x$, and obtain

$$\mu = \frac{e^{X\beta}}{1 + e^{X\beta}}$$

The above formula ensures the mean value of the response variable, μ, will always be between zero and one. In other words, the value the regression model may be used to estimate the *probability* that $y = 1$.

To clarify that our predicted values from logistic regression are probabilities, instead of binary values, let us write the regression model as predicting $p(y)$, the

probability that $y = 1$. If we work backwards from our abbreviated notation, we get the parametric form of the model

$$p(y) = \frac{\exp\left(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p\right)}{1 + \exp\left(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p\right)} + \varepsilon$$

and can write the descriptive form as

$$\hat{p}(y) = \frac{\exp\left(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p\right)}{1 + \exp\left(b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p\right)}$$

13.4 AN APPLICATION OF LOGISTIC REGRESSION MODELING

Let us revisit the *clothing_sales_training* and *clothing_sales_test* data sets. This time, our goal is to determine whether or not customers have a store credit card, so our marketing team can send out advertisements to non-holders, enticing them to sign up for a card. Our response variable in this case is binary: Yes, the customer has a card; or No, the customer does not. Since the response variable is binary, we will use logistic regression.

Our provisional logistic regression model will be

$$\hat{p}(credit\ card) = \frac{\exp\left(b_0 + b_1\left(Days\ between\ Purchases\right) + b_2\left(Web\ Account\right)\right)}{1 + \exp\left(b_0 + b_1\left(Days\ between\ Purchases\right) + b_2\left(Web\ Account\right)\right)}$$

The results of the regression of Credit Card on the two predictor variables are shown in Figure 13.1. The *p*-values shown in the output tell us that both variables belong in the model. When we cross-validate the results with the test data set, we obtain the results shown in Figure 13.2.

The test model confirms that both variables belong in the model. Using the coefficients from the training data set, we obtain our final logistic regression model:

$$\hat{p}(credit\ card) = \frac{\exp\left(0.496 - 0.004\left(Days\ between\ Purchases\right) + 1.254\left(Web\ Account\right)\right)}{1 + \exp\left(0.496 - 0.004\left(Days\ between\ Purchases\right) + 1.254\left(Web\ Account\right)\right)}$$

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	0.4962	0.0887	5.5968	0.0000	0.3224	0.6699
Days	-0.0037	0.0004	-8.4491	0.0000	-0.0046	-0.0028
Web	1.2537	0.3307	3.7914	0.0001	0.6056	1.9018

Figure 13.1 Python logistic regression results for the training data set.

```
------------------------------------------------------------------
          Coef.    Std.Err.      z      P>|z|     [0.025    0.975]
------------------------------------------------------------------
const     0.4634     0.0873    5.3105   0.0000    0.2924    0.6345
Days     -0.0035     0.0004   -8.2261   0.0000   -0.0043   -0.0026
Web       1.0973     0.2830    3.8780   0.0001    0.5427    1.6519
==================================================================
```

Figure 13.2 Python logistic regression results for the test data set.

So, how do we interpret the logistic regression coefficients? Each regression coefficient describes the estimated change in the log-odds of the response variable when the coefficient's predictor variable increases by one. To illustrate, consider the binary predictor variable *Web Account*. The regression coefficient for *Web Account* is 1.254. By calculating $e^{1.254} = 3.504$, we find that a customer is about 3.5 times as likely to have a store credit card if they have a web account compared to if they do not have a web account.

We can perform a similar operation for the coefficient of *Days between Purchases*. By calculating $e^{-0.004} = 0.996$, we find that for every additional day between purchases, the customer is 0.4% less likely to have a store credit card. Since counting the individual days might be too narrow of a measurement, we can multiply the coefficient by 30 to obtain $e^{30 \times -0.004} = 0.89$, and discover that, for every 30 days without a purchase, the customer is another 11% less likely to have a store credit card.[1]

13.4.1 How to Perform Logistic Regression Using Python

Load the required packages, and import the *clothing_sales_training* and *clothing_sales_test* data sets as *sales_train* and *sales_test*, respectively.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
from scipy import stats
sales_train = pd.read_csv("C:/.../clothing_sales_
training.csv")
sales_test = pd.read_csv("C:/.../clothing_sales_test.
csv")
```

For simplicity, we separate the variables into predictor variables **X** and response variable **y**. Add a constant to the **X** data frame in order to include a constant term in our regression model.

```
X = pd.DataFrame(sales_train[['Days', 'Web']])
X = sm.add_constant(X)
y = pd.DataFrame(sales_train[['CC']])
```

[1]For more on logistic regression, see *Data Mining and Predictive Analytics*.

To perform logistic regression, use the **Logit()** and **fit()** commands. Save the model output and run the **summary2()** command on the saved model output to view the model results.

```
logreg01 = sm.Logit(y, X).fit()
logreg01.summary2()
```

An excerpt of the results is shown in Figure 13.1.

To validate the model, perform the same steps on the test data set. The code is given below.

```
X_test = pd.DataFrame(sales_test[['Days', 'Web']])
X_test = sm.add_constant(X_test)
y_test = pd.DataFrame(sales_test[['CC']])
logreg01_test = sm.Logit(y_test, X_test).fit()
logreg01_test.summary2()
```

An excerpt of the results is shown in Figure 13.2.

13.4.2 How to Perform Logistic Regression Using R

Import the *clothing_sales_training* and *clothing_sales_test* data sets as *sales_train* and *sales_test*, respectively.

To run the logistic regression model, we will use the **glm()** command.

```
logreg01 <- glm(formula = CC ~ Days + Web, data = sales_
train, family = binomial)
```

Much of the code is similar to that in Chapter 11; the **formula** input lists the response and predictor variables, and the **data = sales_train** input specifies the data set. The only changes are the **glm()** command and adding the **family = binomial** input. The **glm()** command will run GLM analysis, and **family = binomial** specifies a logistic regression model. Save the model output as **logreg01**.

To view the summary of the model, run the **summary()** command with the name of the saved model as the sole input. An excerpt from the output is shown in Figure 13.3.

```
summary(logreg01)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.4961706  0.0886529   5.597 2.18e-08 ***
Days        -0.0037016  0.0004381  -8.449  < 2e-16 ***
Web          1.2536955  0.3306672   3.791  0.00015 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 13.3 Logistic regression results from R for the training data set.

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.4634478  0.0872706   5.310 1.09e-07 ***
Days        -0.0034721  0.0004221  -8.226  < 2e-16 ***
Web          1.0972994  0.2829570   3.878 0.000105 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 13.4 Logistic regression results from R for the test data set.

To validate the model, run the same model on the test data and obtain the summary of the model (Figure 13.4). The code is given below.

```
logreg01_test <- glm(formula = CC ~ Days + Web, data =
sales_test, family = binomial)
summary(logreg01_test)
```

13.5 POISSON REGRESSION

There are many other kinds of regression models that fall under the umbrella of GLM. We will examine one other: Poisson regression. Poisson regression is used when you want to predict a count of events, such as how many times a customer will contact customer service. The distribution of the response variable will be a count of occurrences, with a minimum value of zero.

The link function for a count response variable is $g(\mu) = \ln(\mu)$. We set the link function equal to our linear predictor to obtain

$$\boldsymbol{X\beta} = \ln\left(\mu\right)$$

After isolating μ, we have

$$\mu = e^{\boldsymbol{x\beta}}$$

Working backwards from our abbreviated notation, we find the parametric version of the Poisson regression equation

$$y = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p} + \varepsilon$$

from which we can write the descriptive form

$$\hat{y} = e^{b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_p x_p}$$

13.6 AN APPLICATION OF POISSON REGRESSION MODELING

We will use the *churn* data set to build a model that estimates the number of customer service calls based on whether a customer churned. Our response variable is an integer-valued variable, which is why we use Poisson regression instead of linear regression for this estimation.

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.3714      0.016     23.877      0.000       0.341       0.402
Churn = True   0.4305      0.034     12.582      0.000       0.363       0.498
==============================================================================
```

Figure 13.5 Python Poisson regression results for predicting number of customer service calls.

The structure of our Poisson regression model will be

$$CustServ\ Calls = \exp\left(b_0 + b_1\left(Churn\right)\right)$$

The result of the regression analysis is given in Figure 13.5. Using the coefficients given above, we can build the Poisson regression model

$$CustServ\ Calls = \exp\left(0.3714 + 0.4305\left(Churn = True\right)\right)$$

Now, how do we interpret the Poisson regression coefficients? When used as the exponent of *e*, the regression coefficient describes the estimated multiplicative change in the response variable when the coefficient's predictor variable increases by one. In our case, the regression coefficient is 0.4305, which gives us $e^{0.4305} = 1.538$. The coefficient's predictor, *Churn*, is zero if the customer does not churn and one if they do. Therefore, the movement from a non-churning to churning customer increases the predicted number of customer service calls that customer makes by 1.538 times, or 53.8%.

13.6.1 How to Perform Poisson Regression Using Python

Naturally, in order to validate our modeling results, you would first split the data into training and test data sets. As cross-validation has been shown in other chapters as well as for logistic regression, we restrict this section to illustrating how to build the Poisson regression model.

Load the required packages.

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import statsmodels.tools.tools as stattools
```

Read in the *churn* data set into Python as *churn*.

```
churn = pd.read_csv("C:/.../churn")
```

Our predictor variable is *Churn*, which is categorical. Similar to previous modeling tasks, we need to change the categorical values of *Churn* into dummy

variables. For this exercise we will use one dummy variable, which equals one if a customer has churned.

```
churn_ind = pd.get_dummies(churn['Churn'], drop_first =
True)
```

The **get_dummies()** command creates two indicator variables, one for each categorical value in *Churn*. The **drop_first = True** input will drop the first dummy variable, which corresponds to *Churn* = *False* in our case, and retain the remaining dummy variable, which corresponds to *Churn* = *True*.

The remaining commands save the new dummy variable as a data frame, add a constant term so our regression model will have a constant, and renames the columns so the output will be easier to read.

```
X = pd.DataFrame(churn_ind)
X = sm.add_constant(X)
X.columns = ['const', 'Churn = True']
```

We also prepare the response variable.

```
y = pd.DataFrame(churn[['CustServ Calls']])
```

Finally, we run Poisson regression using the **GLM()** command.

```
poisreg01 = sm.GLM(y, X, family = sm.families.
Poisson()).fit()
```

Notice the three input values of the **GLM()** command. The first two, **y** and **X**, specify the response variable and predictor variables, respectively. The third, **family = sm.families.Poisson()**, specifies that Poisson regression should be used. The **fit()** command will fit the model to our data. We save the result as **poisreg01**.

Use the **summary()** command to view the results of the model.

```
poisreg01.summary()
```

An excerpt from the output of **summary()** is shown in Figure 13.5.

13.6.2 How to Perform Poisson Regression Using R

As with our Python example, we restrict this section to illustrating how to build the Poisson regression model. We return to the **glm()** command, which we used previously to build a logistic regression model, to now build a Poisson regression model.

```
poisreg01 <- glm(formula = CustServ.Calls ~ Churn, data =
churn, family = poisson)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.37377    0.01638   22.82   <2e-16 ***
ChurnTrue    0.42795    0.03602   11.88   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 13.6 Poisson regression results from R.

The **formula** input now specifies *CustServ Calls* as the response variable and *Churn* as the predictor variable. The input **family = poisson** specifies that Poisson regression should be applied to the data. Save the regression output as **poisreg01**.

Use the **summary()** command to view details about the model.

```
summary(poisreg01)
```

An excerpt from the **summary()** command is shown here in Figure 13.6.

REFERENCE

For a full text on exploring the world of GLM: P. McCullagh and J. A. Nelder, *Generalized Linear Models*, Second Edition, Chapman & Hall, London, 1992.

EXERCISES

CLARIFYING THE CONCEPTS

1. What are the three cases of regression response variables discussed in this chapter?
2. What category of regression models includes all three cases of response variables?
3. What do we call the linear predictor? How to we write it in its abbreviated form?
4. The link function connects what two things? How do we write it in its abbreviated form?
5. What is the link function for linear regression?
6. What kind of regression should we use when trying to predict a binary response variable?
7. What is the link function for logistic regression?
8. Are the predicted values from logistic regression probabilities or binary values?
9. What is the descriptive form of the logistic regression model?
10. What kind of regression should we use when trying to predict a count response variable?
11. What is the link function for Poisson regression?
12. What is the descriptive form of the Poisson regression model?

WORKING WITH THE DATA

For the following exercises, work with the *clothing_sales_training* and *clothing_sales_test* data sets. Use either Python or R to solve each problem.

13. Create a logistic regression model to predict whether or not a customer has a store credit card, based on whether they have a web account and the days between purchases. Obtain the summary of the model.

14. Are there any variables that should be removed from the model? If so, remove them and rerun the model.

15. Write the descriptive form of the logistic regression model using the coefficients obtained from Question 1.

16. Validate the model using the test data set.

17. Obtain the predicted values of the response variable for each record in the data set.

For the following exercises, work with the *churn* data set. Use either Python or R to solve each problem.

18. Create a Poisson regression model to predict the number of customer service calls a person makes, based on whether or not that customer churned. Obtain a summary of the model.

19. Write the descriptive form of the Poisson regression model from the previous exercise.

HANDS-ON ANALYSIS

For the following exercises, work with the *adult* data set. Use either Python or R to solve each problem.

20. Build a logistic regression model to predict the income of a person based on their age, education (as a number, with variable *education.num*), and the hours worked per week. Obtain the summary of the model.

21. Are there any variables that should be removed from the model from the previous exercise? If so, remove the variables and rerun the model.

22. Write the descriptive form of the final logistic regression model from the previous exercise.

23. Interpret the coefficient of the *age* variable.

24. Find the impact on the probability of having high income for every 10 years a person is older.

25. Interpret the coefficient of the *education.num* variable.

26. Find the impact on the probability of having high income for every four more years of education a person has.

27. Interpret the coefficient of the *hours.per.week* variable.

28. Find the impact on the probability of having high income for every five more hours per week a person works.

29. Obtain the predicted values using the model from the previous exercise. Compare the predicted values to the actual values.

30. Build a Poisson regression model to predict the years of education a person has (using the variable *education.num*) based on a person's age and the hours they work per week. Obtain the summary of the model.

31. Are there any variables that should be removed from the model from the previous exercise? If so, remove the variables and rerun the model.

32. Write the descriptive form of the final Poisson regression model from the previous exercise.

33. Obtain the predicted values using the model from the previous exercise. Compare the predicted values to the actual values.