CHAPTER 10

Logistic Regression

In this chapter, we describe the highly popular and powerful classification method called logistic regression. Like linear regression, it relies on a specific model relating the predictors with the outcome. The user must specify the predictors to include as well as their form (e.g., including any interaction terms). This means that even small datasets can be used for building logistic regression classifiers, and that once the model is estimated, it is computationally fast and cheap to classify even large samples of new records. We describe the logistic regression model formulation and its estimation from data. We also explain the concepts of "logit," "odds," and "probability" of an event that arise in the logistic model context and the relations among the three. We discuss variable importance and coefficient interpretation, as well as variable selection for dimension reduction, and extensions to multi-class classification.

Python

In this chapter, we will use `pandas` for data handling, `scikit-learn` and `statsmodels` for the models, and `matplotlib` for visualization. We will also make use of the utility functions from the Python Utilities Functions Appendix. Use the following import statements for the Python code in this chapter.

import required functionality for this chapter

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.model_selection import train_test_split
import statsmodels.api as sm
from mord import LogisticIT
import matplotlib.pylab as plt
import seaborn as sns
from dmba import classificationSummary, gainsChart, liftChart
from dmba.metric import AIC_score
```

Data Mining for Business Analytics: Concepts, Techniques, and Applications in Python, First Edition.
Galit Shmueli, Peter C. Bruce, Peter Gedeck, and Nitin R. Patel

10.1 INTRODUCTION

Logistic regression extends the ideas of linear regression to the situation where the outcome variable, Y, is categorical. We can think of a categorical variable as dividing the records into classes. For example, if Y denotes a recommendation on holding/selling/buying a stock, we have a categorical variable with three categories. We can think of each of the stocks in the dataset (the records) as belonging to one of three classes: the *hold* class, the *sell* class, and the *buy* class. Logistic regression can be used for classifying a new record, where its class is unknown, into one of the classes, based on the values of its predictor variables (called *classification*). It can also be used in data where the class is known, to find factors distinguishing between records in different classes in terms of their predictor variables, or "predictor profile" (called *profiling*). Logistic regression is used in applications such as

1. Classifying customers as returning or nonreturning (classification)
2. Finding factors that differentiate between male and female top executives (profiling)
3. Predicting the approval or disapproval of a loan based on information such as credit scores (classification)

The logistic regression model is used in a variety of fields: whenever a structured model is needed to explain or predict categorical (in particular, binary) outcomes. One such application is in describing choice behavior in econometrics.

In this chapter, we focus on the use of logistic regression for classification. We deal mostly with a binary outcome variable having two possible classes. In Section 10.5, we show how the results can be extended to the case where Y assumes more than two possible classes. Popular examples of binary outcomes are success/failure, yes/no, buy/don't buy, default/don't default, and survive/die. For convenience, we often code the values of the binary outcome variable Y as 0 and 1.

Note that in some cases we may choose to convert a continuous outcome variable or an outcome variables with multiple classes into a binary outcome variable for purposes of simplification, reflecting the fact that decision-making may be binary (approve the loan/don't approve, make an offer/don't make an offer). As with multiple linear regression, the predictor variables $X_1, X_2, \ldots, X_k$ may be categorical variables, continuous variables, or a mixture of these two types. While in multiple linear regression the aim is to predict the value of the continuous Y for a new record, in logistic regression the goal is to predict which class a new record will belong to, or simply to *classify* the record into one of the classes. In the stock example, we would want to classify a new stock into one of the three

recommendation classes: sell, hold, or buy. Or, we might want to compute for a new record its *propensity* (= the probability) to belong to each class, and then possibly rank a set of new records from highest to lowest propensity in order to act on those with the highest propensity.[1]

In logistic regression, we take two steps: the first step yields estimates of the *propensities* or *probabilities* of belonging to each class. In the binary case, we get an estimate of $p = \text{P}(Y = 1)$, the probability of belonging to class 1 (which also tells us the probability of belonging to class 0). In the next step, we use a cutoff value on these probabilities in order to classify each case into one of the classes. For example, in a binary case, a cutoff of 0.5 means that cases with an estimated probability of $\text{P}(Y = 1) \geq 0.5$ are classified as belonging to class 1, whereas cases with $\text{P}(Y = 1) < 0.5$ are classified as belonging to class 0. This cutoff does not need to be set at 0.5. When the event in question is a low probability but notable or important event (say, 1 = fraudulent transaction), a lower cutoff may be used to classify more cases as belonging to class 1.

10.2 The Logistic Regression Model

The idea behind logistic regression is straightforward: Instead of using Y directly as the outcome variable, we use a function of it, which is called the *logit*. The logit, it turns out, can be modeled as a linear function of the predictors. Once the logit has been predicted, it can be mapped back to a probability.

To understand the logit, we take several intermediate steps: First, we look at $p = \text{P}(Y = 1)$, the probability of belonging to class 1 (as opposed to class 0). In contrast to the binary variable Y, which only takes the values 0 and 1, p can take any value in the interval $[0, 1]$. However, if we express p as a linear function of the q predictors[2] in the form

$$p = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q, \tag{10.1}$$

it is not guaranteed that the right-hand side will lead to values within the interval $[0, 1]$. The solution is to use a nonlinear function of the predictors in the form

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q)}}. \tag{10.2}$$

[1] Potentially, one could use linear regression for classification, by training a linear regression on a 0/1 outcome variable (called a Linear Probability Model). The model is then used to generate numerical predictions which are converted into binary classifications using a threshold. However, linear probability models, despite their name, do not produce proper predicted probabilities. The numerical predictions they produce are useful for comparison to the classification threshold, but are otherwise meaningless.

[2] Unlike elsewhere in the book, where p denotes the number of predictors, in this chapter we use q, to avoid confusion with the probability p.

This is called the *logistic response function*. For any values $x_1, \ldots, x_q$, the right-hand side will always lead to values in the interval $[0, 1]$. Next, we look at a different measure of belonging to a certain class, known as *odds*. The odds of belonging to class 1 are defined as *the ratio of the probability of belonging to class 1 to the probability of belonging to class 0*:

$$\text{Odds}(Y = 1) = \frac{p}{1 - p}. \tag{10.3}$$

This metric is very popular in horse races, sports, gambling, epidemiology, and other areas. Instead of talking about the *probability* of winning or contacting a disease, people talk about the *odds* of winning or contacting a disease. How are these two different? If, for example, the probability of winning is 0.5, the odds of winning are $0.5/0.5 = 1$. We can also perform the reverse calculation: Given the odds of an event, we can compute its probability by manipulating equation (10.3):

$$p = \frac{\text{odds}}{1 + \text{odds}}. \tag{10.4}$$

Substituting (10.2) into (10.4), we can write the relationship between the odds and the predictors as

$$\text{Odds}(\text{Y} = 1) = e^{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q}. \tag{10.5}$$

This last equation describes a multiplicative (proportional) relationship between the predictors and the odds. Such a relationship is interpretable in terms of percentages, for example, a unit increase in predictor X_j is associated with an average increase of $\beta_j \times 100\%$ in the odds (holding all other predictors constant).

Now, if we take a natural logarithm[3] on both sides, we get the standard formulation of a logistic model:

$$\log(\text{odds}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_q x_q. \tag{10.6}$$

The log(odds), called the *logit*, takes values from $-\infty$ (very low odds) to ∞ (very high odds).[4] A logit of 0 corresponds to even odds of 1 (probability = 0.5). Thus, our final formulation of the relation between the outcome and the predictors uses the logit as the outcome variable and models it as a *linear function* of the q predictors.

To see the relationship between the probability, odds, and logit of belonging to class 1, look at Figure 10.1, which shows the odds (top) and logit (bottom) as a function of p. Notice that the odds can take any non-negative value, and that the logit can take any real value.

[3]The natural logarithm function is typically denoted ln() or log(). In this book, we use log().
[4]We use the terms *odds* and *odds(Y = 1)* interchangeably.

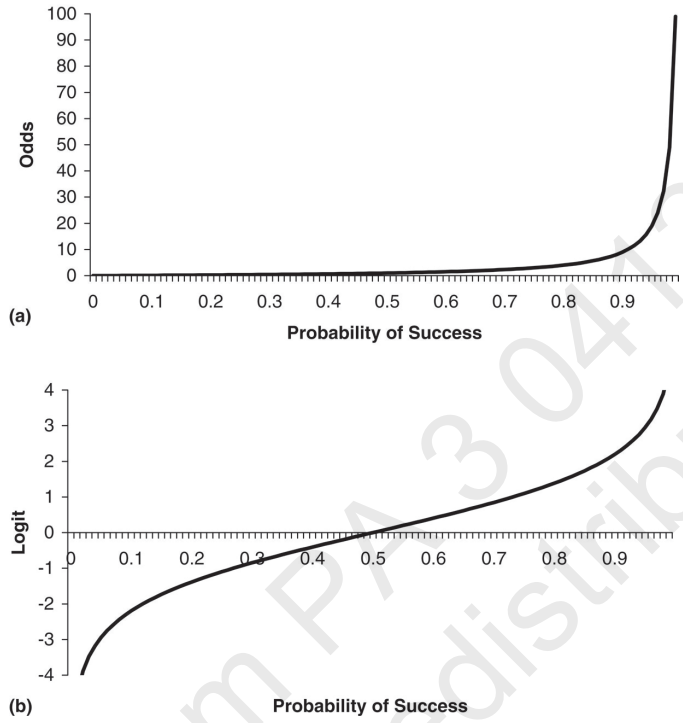

FIGURE 10.1 (a) ODDS AND (b) LOGIT AS A FUNCTION OF p

10.3 Example: Acceptance of Personal Loan

Recall the example described in Chapter 9 of acceptance of a personal loan by Universal Bank. The bank's dataset includes data on 5000 customers. The data include the customer's response to the last personal loan campaign (Personal Loan), as well as customer demographic information (Age, Income, etc.) and the customer's relationship with the bank (mortgage, securities account, etc.). See Table 10.1. Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan offered to them in a previous campaign. The goal is to build a model that identifies customers who are most likely to accept the loan offer in future mailings.

Model with a Single Predictor

Consider first a simple logistic regression model with just one predictor. This is conceptually analogous to the simple linear regression model in which we fit a straight line to relate the outcome, Y, to a single predictor, X.

Let us construct a simple logistic regression model for classification of customers using the single predictor *Income*. The equation relating the outcome

TABLE 10.1 DESCRIPTION OF PREDICTORS FOR ACCEPTANCE OF PERSONAL LOAN EXAMPLE

Age	Customer's age in completed years
Experience	Number of years of professional experience
Income	Annual income of the customer ($000s)
Family Size	Family size of the customer
CCAvg	Average spending on credit cards per month ($000s)
Education	Education Level. 1: Undergrad; 2: Graduate; 3: Advanced/Professional
Mortgage	Value of house mortgage if any ($000s)
Securities Account	Coded as 1 if customer has securities account with bank
CD Account	Coded as 1 if customer has certificate of deposit (CD) account with bank
Online Banking	Coded as 1 if customer uses Internet banking facilities
Credit Card	Coded as 1 if customer uses credit card issued by Universal Bank

variable to the predictor in terms of probabilities is

$$\text{P}(\text{Personal Loan} = \textit{Yes} \mid \text{Income} = x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}},$$

or equivalently, in terms of odds,

$$\text{Odds}(\text{Personal Loan} = \textit{Yes} \mid \text{Income} = x) = e^{\beta_0 + \beta_1 x}. \qquad (10.7)$$

Suppose the estimated coefficients for the model are $\hat{\beta}_0 = -6.04892$ and $\hat{\beta}_1 = 0.036$. So the fitted model is

$$\text{P}(\text{Personal Loan} = \textit{Yes} \mid \text{Income} = x) = \frac{1}{1 + e^{-6.04892 - 0.036x}}. \qquad (10.8)$$

Although logistic regression can be used for prediction in the sense that we predict the *probability* of a categorical outcome, it is most often used for classification. To see the difference between the two, consider predicting the probability of a customer accepting the loan offer as opposed to classifying the customer as an acceptor/nonacceptor. From Figure 10.2, it can be seen that the loan acceptance probabilities produced by the logistic regression model (the s-shaped curve in Figure 10.2) can yield values between 0 and 1. To end up with classifications into either 1 or 0 (e.g., a customer either accepts the loan offer or not), we need a threshold, or cutoff value (see section on "Propensities and Cutoff for Classification" in Chapter 5). This is true in the case of multiple predictor variables as well.

In the Universal Bank example, in order to classify a new customer as an acceptor/nonacceptor of the loan offer, we use the information on his/her income by plugging it into the fitted equation in (10.8). This yields an estimated probability of accepting the loan offer. We then compare it to the cutoff value. The customer is classified as an acceptor if the probability of his/her accepting the offer is above the cutoff.[5]

[5]Here we compared the probability to a cutoff c. If we prefer to look at *odds* of accepting rather than the probability, an equivalent method is to use the equation in (10.7) and compare the odds to

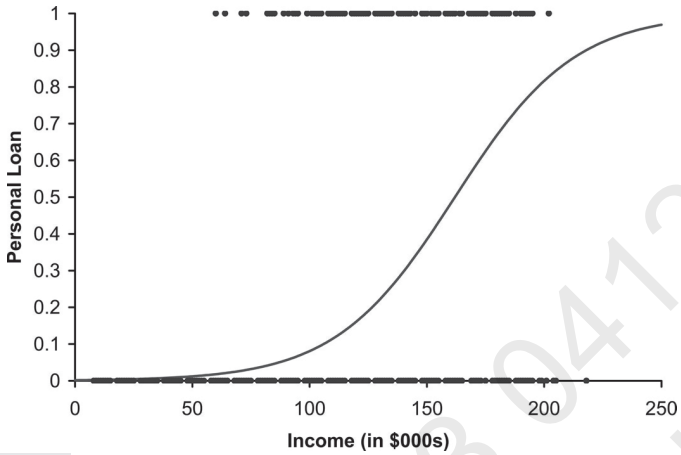

FIGURE 10.2 PLOT OF DATA POINTS (PERSONAL LOAN AS A FUNCTION OF INCOME) AND THE FITTED LOGISTIC CURVE

Estimating the Logistic Model from Data: Computing Parameter Estimates

In logistic regression, the relation between Y and the β parameters is nonlinear. For this reason, the β parameters are not estimated using the method of least squares (as in multiple linear regression). Instead, a method called *maximum likelihood* is used. The idea, in brief, is to find the estimates that maximize the chance of obtaining the data that we have. This requires iterations using a computer program.[6]

Algorithms to compute the coefficient estimates are less robust than algorithms for linear regression. Computed estimates are generally reliable for well-behaved datasets where the number of records with outcome variable values of both 0 and 1 are large; their ratio is "not too close" to either 0 or 1; and when the number of coefficients in the logistic regression model is small relative to the sample size (say, no more than 10%). As with linear regression, collinearity (strong correlation among the predictors) can lead to computational difficulties. Computationally intensive algorithms have been developed recently that circumvent some of these difficulties. For technical details on the maximum likelihood estimation in logistic regression, see Hosmer and Lemeshow (2000).

$c/(1-c)$. If the odds are higher than this number, the customer is classified as an acceptor. If it is lower, we classify the customer as a nonacceptor.

[6]The method of maximum likelihood ensures good asymptotic (large sample) properties for the estimates. Under very general conditions, maximum likelihood estimators are: (1) *Consistent*—The probability of the estimator differing from the true value approaches zero with increasing sample size, (2) *Asymptotically efficient*—The variance is the smallest possible among consistent estimators, and (3) *Asymptotically normally distributed*—This allows us to compute confidence intervals and perform statistical tests in a manner analogous to the analysis of multiple linear regression models, provided that the sample size is *large*.

To illustrate a typical output from such a procedure, we fit a logistic model to the training set of 3000 Universal Bank customers. The outcome variable is Personal Loan, with *Yes* defined as the *success* (this is equivalent to setting the outcome variable to 1 for an acceptor and 0 for a nonacceptor).

Data Preprocessing We start by converting predictor variable Education into a set of dummy variables. In the dataset, it is coded as an integer, taking on values 1, 2, or 3. To turn it into a dummy variable, we first convert it into a categorical variable using the pandas methods *astype('category')* and *rename_categories*. Then we apply the function *pd.get_dummies* on the data frame to create two dummy variables from the three categories. The logistic regression will only use two of the three categories because using all three would create a multicollinearity issue (see Chapter 6). In total, the logistic regression function in `statsmodels` will include $6 = 2 + 1 + 1 + 1 + 1$ dummy variables to describe the five categorical predictors from Table 10.1. Together with the six numerical predictors, we have a total of 12 predictors.

Next, we partition the data randomly into training (60%) and validation (40%) sets. We use the training set to fit a logistic regression model and the validation set to assess the model's performance.

Estimated Model Table 10.2 presents the output from running a logistic regression using the 12 predictors on the training data. The model based on all 12 predictors has the estimated logistic equation

$$\begin{aligned}\text{Logit}(\text{Personal Loan} = \text{Yes}) = \quad & (10.9)\\ & -12.619 - 0.0325\ \text{Age} + 0.0342\ \text{Experience}\\ & +0.0588\ \text{Income} + 0.6141\ \text{Family} + 0.2405\ \text{CCAvg}\\ & +0.0010\ \text{Mortgage} - 1.0262\ \text{Securities\_Account} + 3.6479\ \text{CD\_Account}\\ & -0.6779\ \text{Online} - 0.9560\ \text{Credit Card}\\ & +4.1922\ \text{Education\_Graduate} + 4.3417\ \text{Education\_Advanced/Professional}\end{aligned}$$

The positive coefficients for the dummy variables *Education_Graduate, Education_Advanced/Professional*, and *CD_Account* mean that holding a CD account and having graduate or professional education (all marked by 1 in the dummy variables) are associated with higher probabilities of accepting the loan offer. In contrast, having a securities account, using online banking, and owning a Universal Bank credit card are associated with lower acceptance rates. For the continuous predictors, positive coefficients indicate that a higher value on that predictor is associated with a higher probability of accepting the loan offer (e.g., Income: higher-income customers tend more to accept the offer). Similarly, negative coefficients indicate that a higher value on that predictor is associated with a lower probability of accepting the loan offer (e.g., Age: older customers are less likely to accept the offer).

TABLE 10.2 LOGISTIC REGRESSION MODEL FOR LOAN ACCEPTANCE (TRAINING DATA)

code for fitting a logistic regression model

```
bank_df = pd.read_csv('UniversalBank.csv')
bank_df.drop(columns=['ID', 'ZIP Code'], inplace=True)
bank_df.columns = [c.replace(' ', '_') for c in bank_df.columns]

# Treat education as categorical, convert to dummy variables
bank_df['Education'] = bank_df['Education'].astype('category')
new_categories = {1: 'Undergrad', 2: 'Graduate', 3: 'Advanced/Professional'}
bank_df.Education.cat.rename_categories(new_categories, inplace=True)
bank_df = pd.get_dummies(bank_df, prefix_sep='_', drop_first=True)

y = bank_df['Personal_Loan']
X = bank_df.drop(columns=['Personal_Loan'])

# partition data
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=1)

# fit a logistic regression (set penalty=l2 and C=1e42 to avoid regularization)
logit_reg = LogisticRegression(penalty="l2", C=1e42, solver='liblinear')
logit_reg.fit(train_X, train_y)

print('intercept ', logit_reg.intercept_[0])
print(pd.DataFrame({'coeff': logit_reg.coef_[0]}, index=X.columns).transpose())

print('AIC', AIC_score(valid_y, logit_reg.predict(valid_X), df = len(train_X.columns) + 1))
```

Output

```
intercept  -12.61895521314035
             Age  Experience    Income    Family     CCAvg  Mortgage
coeff -0.032549     0.03416  0.058824  0.614095  0.240534  0.001012

      Securities_Account  CD_Account    Online  CreditCard
coeff          -1.026191    3.647933 -0.677862    -0.95598

      Education_Graduate  Education_Advanced/Professional
coeff           4.192204                         4.341697

AIC -709.1524769205962
```

Interpreting Results in Terms of Odds (for a Profiling Goal)

Logistic models, when they are appropriate for the data, can give useful information about the roles played by different predictor variables. For example, suppose we want to know how increasing family income by one unit will affect the probability of loan acceptance. This can be found straightforwardly if we consider not probabilities, but odds.

Recall that the odds are given by

$$\text{Odds} = e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_q x_q}.$$

At first, let us return to the single predictor example, where we model a customer's acceptance of a personal loan offer as a function of his/her income:

$$\text{Odds}(\text{Personal Loan} = \textit{Yes} \mid \text{Income}) = e^{\beta_0+\beta_1 \text{Income}}.$$

We can think of the model as a multiplicative model of odds. The odds that a customer with income zero will accept the loan is estimated by $e^{-6.04892+(0.036)(0)} = 0.00236$. These are the *base case odds*. In this example, it is obviously economically meaningless to talk about a zero income; the value zero and the corresponding base-case odds could be meaningful, however, in the context of other predictors. The odds of accepting the loan with an income of \$100K will increase by a multiplicative factor of $e^{(0.036)(100)} = 36.6$ over the base case, so the odds that such a customer will accept the offer are $e^{-6.04892+(0.036)(100)} = 0.086$.

Suppose that the value of Income, or in general X_1, is increased by one unit from x_1 to $x_1 + 1$, while the other predictors are held at their current value $(x_2, \ldots, x_{12})$. We get the odds ratio

$$\frac{\text{odds}(x_1+1, x_2, \ldots, x_{12})}{\text{odds}(x_1, \ldots, x_{12})} = \frac{e^{\beta_0+\beta_1(x_1+1)+\beta_2 x_2+\cdots+\beta_{12} x_{12}}}{e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_{12} x_{12}}} = e^{\beta_1}.$$

This tells us that a single unit increase in X_1, holding $X_2, \ldots, X_{12}$ constant, is associated with an increase in the odds that a customer accepts the offer by a factor of e^{β_1}. In other words, e^{β_1} is the multiplicative factor by which the odds (of belonging to class 1) increase when the value of X_1 is increased by 1 unit, *holding all other predictors constant*. If $\beta_1 < 0$, an increase in X_1 is associated with a decrease in the odds of belonging to class 1, whereas a positive value of β_1 is associated with an increase in the odds.

When a predictor is a dummy variable, the interpretation is technically the same but has a different practical meaning. For instance, the coefficient for CD_Account in the 12-predictor model was estimated from the data to be 3.647933. Recall that the reference group is customers not holding a CD account. We interpret this coefficient as follows: $e^{3.647933} = 38.4$ are the odds

that a customer who has a CD account will accept the offer relative to a customer who does not have a CD account, holding all other variables constant. This means that customers who hold CD accounts at Universal Bank are more likely to accept the offer than customers without a CD account (holding all other variables constant).

The advantage of reporting results in odds as opposed to probabilities is that statements such as those above are true for any value of X_1. Unless X_1 is a dummy variable, we cannot apply such statements about the effect of increasing X_1 by a single unit to probabilities. This is because the result depends on the actual value of X_1. So if we increase X_1 from, say, 3 to 4, the effect on p, the probability of belonging to class 1, will be different than if we increase X_1 from 30 to 31. In short, the change in the probability, p, for a unit increase in a particular predictor variable, while holding all other predictors constant, is not a constant—it depends on the specific values of the predictor variables. We therefore talk about probabilities only in the context of specific records.

10.4 Evaluating Classification Performance

The general measures of performance that were described in 5 are used to assess the logistic model performance. Recall that there are several performance measures, the most popular being those based on the confusion matrix (accuracy alone or combined with costs) and the gains and lift charts. As in other classification methods, the goal is to find a model that accurately classifies records to their class, using only the predictor information. A variant of this goal is *ranking*, or finding a model that does a superior job of identifying the members of a particular class of interest for a set of new records (which might come at some cost to overall accuracy). Since the training data are used for selecting the model, we expect the model to perform quite well for those data, and therefore prefer to test its performance on the validation set. Recall that the data in the validation set were not involved in the model building, and thus we can use them to test the model's ability to classify data that it has not "seen" before.

To obtain the confusion matrix from a logistic regression analysis, we use the estimated equation to predict the probability of class membership (the *propensities*) for each record in the validation set, and use the cutoff value to decide on the class assignment of these records. We then compare these classifications to the actual class memberships of these records. In the Universal Bank case, we use the estimated model in equation (10.10) to predict the probability of offer acceptance in a validation set that contains 2000 customers (these data were not used in the modeling step). Technically, this is done by computing the $logit$ using the estimated model in equation (10.10) and then obtaining the probabilities p through the relation $p = e^{logit}/1 + e^{logit}$. We then compare these probabilities

to our chosen cutoff value in order to classify each of the 2000 validation records as acceptors or nonacceptors.

Table 10.3 shows propensities for four selected records in the validation set. Suppose that we use a cutoff of 0.5. We see that the first customer has a probability of accepting the offer, $p(1)$, that is lower than the cutoff of 0.5, and therefore s/he is classified as nonacceptor (0). And indeed, this customer was a nonacceptor (actual = 0). The second and third customers' probability of acceptance is estimated by the model to exceed 0.5, and they are therefore classified as acceptors (1). While the third customer was indeed an acceptor (actual = 1), our model misclassified the second customer as an acceptor, when in fact s/he was a nonacceptor (actual = 0). The fourth customer is also missclassified; this time as a nonacceptor while s/he was an acceptor.

TABLE 10.3 PROPENSITIES FOR FOUR CUSTOMERS IN VALIDATION DATA

code for using logistic regression to generate predicted probabilities

```
logit_reg_pred = logit_reg.predict(valid_X)
logit_reg_proba = logit_reg.predict_proba(valid_X)
logit_result = pd.DataFrame({'actual': valid_y,
                             'p(0)': [p[0] for p in logit_reg_proba],
                             'p(1)': [p[1] for p in logit_reg_proba],
                             'predicted': logit_reg_pred })

# display four different cases
interestingCases = [2764, 932, 2721, 702]
print(logit_result.loc[interestingCases])
```

Output

```
      actual   p(0)   p(1)  predicted
2764       0  0.976  0.024          0
932        0  0.335  0.665          1
2721       1  0.032  0.968          1
702        1  0.986  0.014          0
```

Table 10.4 displays the training and validation confusion matrices, using the 0.5 cutoff. We see that in both datasets most non-acceptors (0) are correctly classified, whereas about 1/3 of the acceptors (1) are misclassified.

Another useful tool for assessing model classification performance are the cumulative gains chart and decile lift chart (see Chapter 5). Figure 10.3 illustrates the charts obtained for the personal loan offer logistic model using the validation set. In the cumulative gains chart, the "lift" over the base curve indicates for a given number of cases (read on the x-axis), the additional responders that you can identify by using the model. The same information for percentiles is portrayed in the decile lift chart: Taking the 10% of the records that are ranked by the model as "most probable 1's" yields 7.8 times as many 1's as would simply selecting 10% of the records at random.

TABLE 10.4 TRAINING AND VALIDATION CONFUSION MATRICES

code for using logistic regression to generate confusion matrices

```
# training confusion matrix
classificationSummary(train_y, logit_reg.predict(train_X))

# validation confusion matrix
classificationSummary(valid_y, logit_reg.predict(valid_X))
```

Output

```
Confusion Matrix (Accuracy 0.9603)

       Prediction
Actual    0    1
     0 2684   29
     1   90  197
Confusion Matrix (Accuracy 0.9595)

       Prediction
Actual    0    1
     0 1791   16
     1   65  128
```

code for creating cumulative gains chart and decile lift chart

```
df = logit_result.sort_values(by=['p(1)'], ascending=False)
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10, 4))

gainsChart(df.actual, ax=axes[0])
liftChart(df['p(1)'], title=False, ax=axes[1])

plt.show()
```

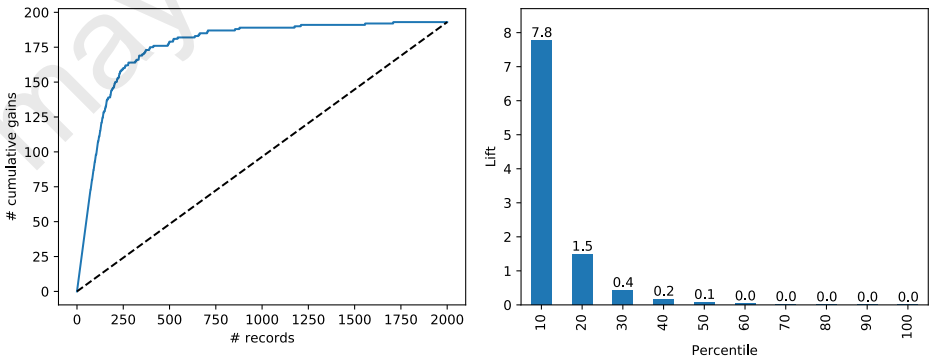

FIGURE 10.3 CUMULATIVE GAINS CHART AND DECILE-WISE LIFT CHART FOR THE VALIDATION DATA FOR UNIVERSAL BANK LOAN OFFER

Variable Selection

The next step includes searching for alternative models. One option is to look for simpler models by trying to reduce the number of predictors used. We can also build more complex models that reflect interactions among predictors by creating and including new variables that are derived from the predictors. For example, if we hypothesize that there is an interactive effect between income and family size, we should add an interaction term of the form Income $\times$ Family. The choice among the set of alternative models is guided primarily by performance on the validation data. For models that perform roughly equally well, simpler models are generally preferred over more complex models. Note also that performance on validation data may be overly optimistic when it comes to predicting performance on data that have not been exposed to the model at all. This is because when the validation data are used to select a final model among a set of model, we are selecting based on how well the model performs with those data and therefore may be incorporating some of the random idiosyncrasies of the validation data into the judgment about the best model. The model still may be the best for the validation data among those considered, but it will probably not do as well with the unseen data. Therefore, it is useful to evaluate the chosen model on a new test set to get a sense of how well it will perform on new data. In addition, one must consider practical issues such as costs of collecting variables, error-proneness, and model complexity in the selection of the final model.

As in linear regression (See Section 6.4 in Chapter 6), in logistic regression we can use automated variable selection heuristics such as stepwise regression, forward selection, and backward elimination. Such methods can be set to minimize AIC or other measures that penalize fit to the training data by considering the number of predictors. We can also use regularization using L1 or L2 penalty. In Python, the penalty can be selected by setting argument `penalty` to l1 or l2. The penalty parameter `C` is set by default to 1. Note that choosing l2 penalty and a very large penalty parameter `C` (such as *LogisticRegression(penalty="l2", C=1e42)*) will lead to ordinary logistic regression. We illustrate the use of regularized logistic regression in Section 10.6.

10.5 Logistic Regression for Multi-class Classification

The logistic model for a binary outcome can be extended for more than two classes. Suppose that there are m classes. Using a logistic regression model, for each record we would have m probabilities of belonging to each of the m classes. Since the m probabilities must add up to 1, we need estimate only $m-1$ probabilities.

Ordinal Classes

Ordinal classes are classes that have a meaningful order. For example, in stock recommendations, the three classes *buy*, *hold*, and *sell* can be treated as ordered. As a simple rule, if classes can be numbered in a meaningful way, we consider them ordinal. When the number of classes is large (typically, more than 5), we can treat the outcome variable as continuous and perform multiple linear regression. When $m = 2$, the logistic model described above is used. We therefore need an extension of the logistic regression for a small number of ordinal classes ($3 \leq m \leq 5$). There are several ways to extend the binary-class case. Here, we describe the *proportional odds* or *cumulative logit method*. For other methods, see Hosmer and Lemeshow (2000).

For simplicity of interpretation and computation, we look at *cumulative* probabilities of class membership. For example, in the stock recommendations, we have $m = 3$ classes. Let us denote them by 1 = *buy*, 2 = *hold*, and 3 = *sell*. The probabilities estimated by the model are $P(Y \leq 1)$, (the probability of a *buy* recommendation) and $P(Y \leq 2)$ (the probability of a *buy* or *hold* recommendation). The three noncumulative probabilities of class membership can easily be recovered from the two cumulative probabilities:

$$\begin{aligned} P(Y = 1) &= P(Y \leq 1), \\ P(Y = 2) &= P(Y \leq 2) - P(Y \leq 1), \\ P(Y = 3) &= 1 - P(Y \leq 2). \end{aligned}$$

Next, we want to model each logit as a function of the predictors. Corresponding to each of the $m-1$ cumulative probabilities is a logit. In our example, we would have

$$\begin{aligned} \text{logit}(\textit{buy}) &= \log \frac{P(Y \leq 1)}{1 - P(Y \leq 1)}, \\ \text{logit}(\textit{buy or hold}) &= \log \frac{P(Y \leq 2)}{1 - P(Y \leq 2)}. \end{aligned}$$

Each of the logits is then modeled as a linear function of the predictors (as in the two-class case). If in the stock recommendations we have a single predictor value x, we compute two logit values using two equations

$$\begin{aligned} \text{logit}(\textit{buy}) &= \alpha_0 + \beta_1 x, \\ \text{logit}(\textit{buy or hold}) &= \beta_0 + \beta_1 x. \end{aligned}$$

This means that both lines have the same slope (β_1) but different intercepts. Once the coefficients $\alpha_0, \beta_0, \beta_1$ are estimated, we can compute the class membership

probabilities by rewriting the logit equations in terms of probabilities. For the three-class case, for example, we would have

$$
\begin{aligned}
P(Y=1) &= P(Y \le 1) = \frac{1}{1+e^{-(a_0+b_1x)}}, \\
P(Y=2) &= P(Y \le 2) - P(Y \le 1) = \frac{1}{1+e^{-(b_0+b_1x)}} - \frac{1}{1+e^{-(a_0+b_1x)}}, \\
P(Y=3) &= 1 - P(Y \le 2) = 1 - \frac{1}{1+e^{-(b_0+b_1x)}},
\end{aligned}
$$

where a_0, b_0, and b_1 are the estimates obtained from the training set.

For each record, we now have the estimated probabilities that it belongs to each of the classes. In our example, each stock would have three probabilities: for a *buy* recommendation, a *hold* recommendation, and a *sell* recommendation. The last step is to classify the record into one of the classes. This is done by assigning it to the class with the highest membership probability. For example, if a stock had estimated probabilities P(Y=1)=0.2, P(Y=2)=0.3, and P(Y=3)=0.5, we would classify it as getting a *sell* recommendation.

Nominal Classes

When the classes cannot be ordered and are simply different from one another, we are in the case of nominal classes. An example is the choice between several brands of cereal. A simple way to verify that the classes are nominal is when it makes sense to tag them as $A, B, C, \ldots$, and the assignment of letters to classes does not matter. For simplicity, let us assume that there are $m = 3$ brands of cereal that consumers can choose from (assuming that each consumer chooses one). Then we estimate the probabilities P(Y=A), P(Y=B), and P(Y=C). As before, if we know two of the probabilities, the third probability is determined. We therefore use one of the classes as the reference class. Let us use C as the reference brand.

The goal, once again, is to model the class membership as a function of predictors. So in the cereals example, we might want to predict which cereal will be chosen if we know the cereal's price x.

Next, we form $m-1$ pseudologit equations that are linear in the predictors. In our example, we would have

$$
\begin{aligned}
\text{logit(A)} &= \log \frac{P(Y=A)}{P(Y=C)} = \alpha_0 + \alpha_1 x, \\
\text{logit(B)} &= \log \frac{P(Y=B)}{P(Y=C)} = \beta_0 + \beta_1 x.
\end{aligned}
$$

Once the four coefficients are estimated from the training set, we can estimate the class membership probabilities[7]:

$$\begin{aligned} P(Y = A) &= \frac{e^{a_0+a_1x}}{1 + e^{a_0+a_1x} + e^{b_0+b_1x}}, \\ P(Y = B) &= \frac{e^{b_0+b_1x}}{1 + e^{a_0+a_1x} + e^{b_0+b_1x}}, \\ P(Y = C) &= 1 - P(Y = A) - P(Y = B), \end{aligned}$$

where a_0, a_1, b_0, and b_1 are the coefficient estimates obtained from the training set. Finally, a record is assigned to the class that has the highest probability.

Comparing Ordinal and Nominal Models

The estimated logistic models will differ, depending on whether the outcome variable is coded as ordinal or nominal. Lets take, for example, data on accidents from the US Bureau of Transportation Statistics. These data can be used to predict whether an accident will result in injuries or fatalities based on predictors such as alcohol involvement, time of day, and road condition. The severity of the accident has three classes: 0 = No Injury, 1 = Nonfatal Injuries, 2 = Fatalities.

We first consider severity as an ordinal variable, and fit a simple model with two nominal predictors, alcohol and weather. Each of these predictors is coded as No (not involved in the accident) or Yes (involved in the accident). Next we change the modeling type for severity to nominal, and fit a model with the same predictors. The reference severity level, in both models, is severity = 2.

Table 10.5 presents Python code for ordinal and nominal multinomial regression applied to the accidents data, displaying the resulting estimated models and predicted probabilities for a few sample records.

The ordinal logistic model has estimates for two intercepts (the first for severity = 0 and and the second for severity = 1), but one estimate for each predictor. In contrast, the nominal logistic model has estimates for the three intercepts, and also has three separate sets of coefficients for the predictors for each level of the outcome variable.

Finally, the resulting predicted probabilities also differ across the two models. Hence, it is useful to determine which model is more suitable for the outcome at hand, and to evaluate predictive performance on a validation set.

[7]From the two logit equations, we see that P(Y = A) = P(Y = C)$\cdot e^{\alpha_0+\alpha_1x}$ and P(Y = B) = P(Y = C)$\cdot e^{\beta_0+\beta_1x}$. Since P(Y = A) + P(Y = B) + P(Y = C) = 1, we get

$$\mathrm{P}(Y = C) = 1 - P(Y = C)e^{\alpha_0+\alpha_1x} - \mathrm{P}(Y = C)e^{\beta_0+\beta_1x} = \frac{1}{e^{\alpha_0+\alpha_1x+e^{\beta_0+\beta_1x}}}. \quad (10.10)$$

By plugging this form into the two equations above it, we also obtain the membership probabilities in classes A and B.

TABLE 10.5 ORDINAL AND NOMINAL (MULTI-CLASS) REGRESSION IN PYTHON

code for logistic regression with more than 2 classes

```
data = pd.read_csv('accidentsFull.csv')
outcome = 'MAX_SEV_IR'
predictors = ['ALCHL_I', 'WEATHER_R']

y = data[outcome]
X = data[predictors]
train_X, train_y = X, y

print('Nominal logistic regression')
logit = LogisticRegression(penalty="l2", solver='lbfgs', C=1e24, multi_class='multinomial')
logit.fit(X, y)
print('  intercept', logit.intercept_)
print('  coefficients', logit.coef_)
print()
probs = logit.predict_proba(X)
results = pd.DataFrame({
    'actual': y, 'predicted': logit.predict(X),
    'P(0)': [p[0] for p in probs],
    'P(1)': [p[1] for p in probs],
    'P(2)': [p[2] for p in probs],
})
print(results.head())
print()

print('Ordinal logistic regression')
logit = LogisticIT(alpha=0)
logit.fit(X, y)
print('  theta', logit.theta_)
print('  coefficients', logit.coef_)
print()
probs = logit.predict_proba(X)
results = pd.DataFrame({
    'actual': y, 'predicted': logit.predict(X),
    'P(0)': [p[0] for p in probs],
    'P(1)': [p[1] for p in probs],
    'P(2)': [p[2] for p in probs],
})
print(results.head())
```

Output

```
Nominal logistic regression
  intercept
    [-0.09100315  0.9036454  -0.81264225]
  coefficients [[ 0.51606685  0.3391015 ]
  [ 0.14900396  0.09543369]
  [-0.66507082 -0.43453518]]

   actual  predicted   P(0)   P(1)   P(2)
0       1          1  0.491  0.499  0.010
1       0          0  0.553  0.441  0.005
2       0          0  0.553  0.441  0.005
3       0          1  0.491  0.499  0.010
4       0          1  0.394  0.579  0.027
```

```
Ordinal logistic regression
  theta
    [-1.06916285  2.77444326]
  coefficients
    [-0.40112008 -0.25174207]

   actual  predicted   P(0)   P(1)   P(2)
0       1          1  0.496  0.483  0.021
1       0          0  0.559  0.425  0.017
2       0          0  0.559  0.425  0.017
3       0          1  0.496  0.483  0.021
4       0          1  0.397  0.571  0.031
```

10.6 Example of Complete Analysis: Predicting Delayed Flights

Predicting flight delays can be useful to a variety of organizations: airport authorities, airlines, aviation authorities. At times, joint task forces have been formed to address the problem. Such an organization, if it were to provide ongoing real-time assistance with flight delays,would benefit from some advance notice about flights likely to be delayed.

In this simplified illustration, we look at six predictors (see Table 10.6). The outcome of interest is whether the flight is delayed or not (*delayed* means more than 15 minutes late). Our data consist of all flights from the Washington, DC area into the New York City area during January 2004. The percent of delayed flights among these 2201 flights is 19.5%. The data were obtained from the Bureau of Transportation Statistics website (www.transtats.bts.gov).

The goal is to predict accurately whether a new flight, not in this dataset, will be delayed or not. The outcome variable is a variable called Flight Status, coded as *delayed* or *ontime*.

Other information available on the website, such as distance and arrival time, is irrelevant because we are looking at a certain route (distance, flight time, etc. should be approximately equal for all flights in the data). A sample of the data for 20 flights is shown in Table 10.7. Figures 10.4 and 10.5 show visualizations of the relationships between flight delays and different predictors or combinations of predictors. From Figure 10.4, we see that Sundays and Mondays saw the largest proportion of delays. Delay rates also seem to differ by carrier, by time of day, as well as by origin and destination airports. For Weather, we see a strong distinction between delays when Weather = 1 (in that case there is always a delay) and Weather = 0. The heatmap in Figure 10.5 reveals some specific combinations with high rates of delays, such as Sunday flights by carrier RU, departing from BWI, or Sunday flights by MQ departing from DCA. We can also see combinations with very low delay rates.

TABLE 10.6 DESCRIPTION OF PREDICTORS FOR FLIGHT DELAYS EXAMPLE

Day of Week	Coded as 1 = Monday, 2 = Tuesday,..., 7 = Sunday
Departure Time	Broken down into 18 intervals between 6:00 AM and 10:00 PM
Origin	Three airport codes: DCA (Reagan National), IAD (Dulles), BWI (Baltimore–Washington Int'l)
Destination	Three airport codes: JFK (Kennedy), LGA (LaGuardia), EWR (Newark)
Carrier	Eight airline codes: CO (Continental), DH (Atlantic Coast), DL (Delta), MQ (American Eagle), OH (Comair), RU (Continental Express), UA (United), and US (USAirways)
Weather	Coded as 1 if there was a weather-related delay

code for generating bar charts of average delay vs. predictors

```
delays_df = pd.read_csv('FlightDelays.csv')
# Create an indicator variable
delays_df['isDelayed'] = [1 if status == 'delayed' else 0
                          for status in delays_df['Flight Status']]

def createGraph(group, xlabel, axis):
    groupAverage = delays_df.groupby([group])['isDelayed'].mean()
    if group == 'DAY_WEEK': # rotate so that display starts on Sunday
        groupAverage = groupAverage.reindex(index=np.roll(groupAverage.index,1))
        groupAverage.index = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']
    ax = groupAverage.plot.bar(color='CO', ax=axis)
    ax.set_ylabel('Average Delay')
    ax.set_xlabel(xlabel)
    return ax

def graphDepartureTime(xlabel, axis):
    temp_df = pd.DataFrame({'CRS_DEP_TIME': delays_df['CRS_DEP_TIME'] // 100,
                            'isDelayed': delays_df['isDelayed']})
    groupAverage = temp_df.groupby(['CRS_DEP_TIME'])['isDelayed'].mean()
    ax = groupAverage.plot.bar(color='CO', ax=axis)
    ax.set_xlabel(xlabel); ax.set_ylabel('Average Delay')

fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(10, 10))

createGraph('DAY_WEEK', 'Day of week', axis=axes[0][0])
createGraph('DEST', 'Destination', axis=axes[0][1])
graphDepartureTime('Departure time', axis=axes[1][0])
createGraph('CARRIER', 'Carrier', axis=axes[1][1])
createGraph('ORIGIN', 'Origin', axis=axes[2][0])
createGraph('Weather', 'Weather', axis=axes[2][1])
plt.tight_layout()
```

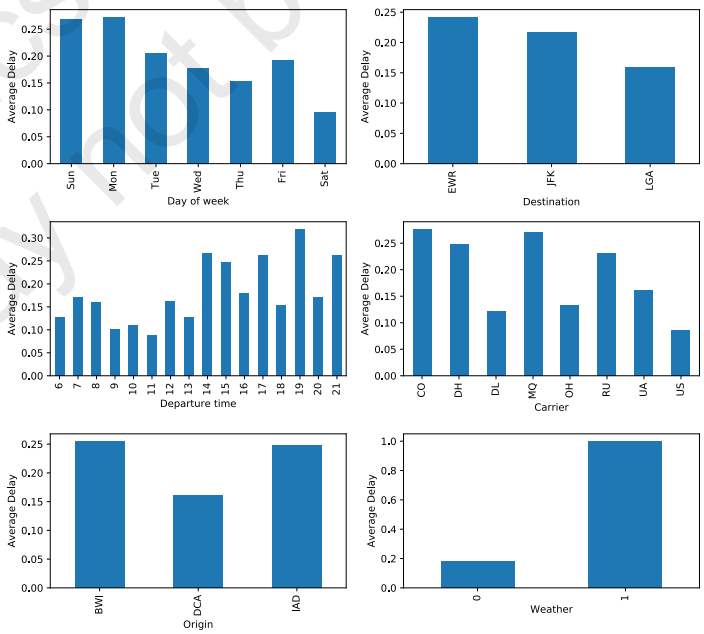

FIGURE 10.4 PROPORTION OF DELAYED FLIGHTS BY EACH OF THE SIX PREDICTORS. TIME OF DAY IS DIVIDED INTO HOURLY BINS

code for generating heatmap for exploring flight delays

```
agg = delays_df.groupby(['ORIGIN', 'DAY_WEEK', 'CARRIER']).isDelayed.mean()
agg = agg.reset_index()

# Define the layout of the graph
height_ratios = []
for i, origin in enumerate(sorted(delays_df.ORIGIN.unique())):
    height_ratios.append(len(agg[agg.ORIGIN == origin].CARRIER.unique()))
gridspec_kw = {'height_ratios': height_ratios, 'width_ratios': [15, 1]}
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(10, 6),
                         gridspec_kw = gridspec_kw)
axes[0, 1].axis('off')
axes[2, 1].axis('off')

maxIsDelay = agg.isDelayed.max()
for i, origin in enumerate(sorted(delays_df.ORIGIN.unique())):
    data = pd.pivot_table(agg[agg.ORIGIN == origin], values='isDelayed', aggfunc=np.sum,
                          index=['CARRIER'], columns=['DAY_WEEK'])
    data = data[[7, 1, 2, 3, 4, 5, 6]]  # Shift last columns to first
    ax = sns.heatmap(data, ax=axes[i][0], vmin=0, vmax=maxIsDelay,
                     cbar_ax=axes[1][1], cmap=sns.light_palette("navy"))
    ax.set_xticklabels(['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat'])
    if i != 2:
        ax.get_xaxis().set_visible(False)
    ax.set_ylabel('Airport ' + origin)
plt.show()
```

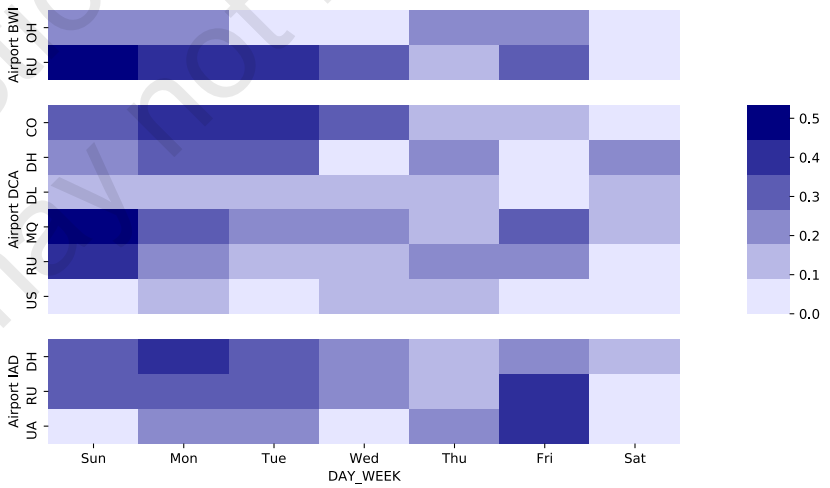

FIGURE 10.5 PERCENT OF DELAYED FLIGHTS (DARKER = HIGHER %DELAYS) BY DAY OF WEEK, ORIGIN, AND CARRIER

TABLE 10.7 SAMPLE OF 20 FLIGHTS

Flight Status	Carrier	Day of Week	Departure Time	Destination	Origin	Weather
ontime	DL	2	728	LGA	DCA	0
delayed	US	3	1600	LGA	DCA	0
ontime	DH	5	1242	EWR	IAD	0
ontime	US	2	2057	LGA	DCA	0
ontime	DH	3	1603	JFK	IAD	0
ontime	CO	6	1252	EWR	DCA	0
ontime	RU	6	1728	EWR	DCA	0
ontime	DL	5	1031	LGA	DCA	0
ontime	RU	6	1722	EWR	IAD	0
delayed	US	1	627	LGA	DCA	0
delayed	DH	2	1756	JFK	IAD	0
ontime	MQ	6	1529	JFK	DCA	0
ontime	US	6	1259	LGA	DCA	0
ontime	DL	2	1329	LGA	DCA	0
ontime	RU	2	1453	EWR	BWI	0
ontime	RU	5	1356	EWR	DCA	0
delayed	DH	7	2244	LGA	IAD	0
ontime	US	7	1053	LGA	DCA	0
ontime	US	2	1057	LGA	DCA	0
ontime	US	4	632	LGA	DCA	0

Our main goal is to find a model that can obtain accurate classifications of new flights based on their predictor information. An alternative goal is finding a certain percentage of flights that are most/least likely to get delayed (*ranking*). And a third different goal is profiling flights: finding out which factors are associated with a delay (not only in this sample but in the entire population of flights on this route), and for those factors we would like to quantify these effects. A logistic regression model can be used for all these goals, albeit in different ways.

Data Preprocessing

Create a binary outcome variable called isDelay that takes the value 1 if Flight Status = *delayed* and 0 otherwise. Transform day of week into a categorical variable, and bin and categorize the departure time into hourly intervals between 6:00 AM and 10:00 PM. Set reference categories for categorical variables: IAD for departure airport, LGA for arrival, USAirways for carrier, and Wednesday for day (see Table 10.8 for Python code). This yields a total of 34 dummies. In addition, we have a single dummy for Weather. While this is a large number of predictors, we start by using all of them, look at performance, and then explore reducing the dimension. We then partition the data into training set (60%) and validation set (40%). We use the training set to fit a model and the validation set to assess the model's performance.

Model Training

The estimated model with 34 predictors is shown in Table 10.8. Note that negative coefficients in the logit model translate into odds coefficients ($e^{\hat{\beta}}$ lower than 1, and positive logit coefficients translate into odds coefficients larger than 1.

Model Interpretation

The coefficient for Arrival Airport JFK (DEST_JFK) is estimated as -0.524. Recall that the reference group is EWR. We interpret this coefficient as follows: $e^{-0.524} = 0.59$ are the odds of a flight arriving at JFK being delayed relative to a flight to EWR being delayed (= the base-case odds), holding all other variables constant. This means that flights to EWR are more likely to be delayed than those to JFK (holding everything else constant). For carriers (CO is the reference category), US has the largest negative coefficient, indicating the lowest odds of delay, while MQ has the highest odds of delay (holding everything else constant). For day of week, all coefficients are negative indicating that Monday (DAY_WEEK_1, the reference category) has the highest odds of delay, while Saturdays have the largest negative coefficient, and thereby the lowest odds of delay (holding everything else constant). Also, odds of delays appear to change over the course of the day, with the most noticeable difference from the reference category (6–7 AM) being 7–8 PM. Finally, Weather has the largest coefficient, indicating a large gap between flights with Weather=0 and Weather=1. Note that we have considered above only coefficient magnitude and not statistical significance. Since our eventual goal is prediction, we will rely instead on predictive performance.

Model Performance

How should we measure the performance of models? One possible measure is "percent of flights correctly classified." Accurate classification can be obtained from the confusion matrix for the validation data. The confusion matrix gives a sense of the classification accuracy and what type of misclassification is more frequent. From the confusion matrix and error rates in Figure 10.6, it can be seen that the model more accurately classifies ontime flights and is less accurate in classifying flights that were delayed. (*Note*: The same pattern appears in the confusion matrix for the training data, so it is not surprising to see it emerge for new data.) If there is an asymmetric cost structure so that one type of misclassification is more costly than the other, the cutoff value can be selected to minimize the cost. Of course, this tweaking should be carried out on the training data and assessed only using the validation data.

TABLE 10.8 ESTIMATED LOGISTIC REGRESSION MODEL FOR DELAYED FLIGHTS (BASED ON THE TRAINING SET)

code for data preprocessing and running logistic regression

```
delays_df = pd.read_csv('FlightDelays.csv')
# Create an indicator variable
delays_df['isDelayed'] = [1 if status == 'delayed' else 0 for status in delays_df['Flight Status']]

# convert to categorical
delays_df.DAY_WEEK = delays_df.DAY_WEEK.astype('category')

# create hourly bins departure time
delays_df.CRS_DEP_TIME = [round(t / 100) for t in delays_df.CRS_DEP_TIME]
delays_df.CRS_DEP_TIME = delays_df.CRS_DEP_TIME.astype('category')

predictors = ['DAY_WEEK', 'CRS_DEP_TIME', 'ORIGIN', 'DEST', 'CARRIER', 'Weather']
outcome = 'isDelayed'

X = pd.get_dummies(delays_df[predictors], drop_first=True)
y = delays_df[outcome]
classes = ['ontime', 'delayed']

# split into training and validation
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=1)

logit_full = LogisticRegression(penalty="l2", C=1e42, solver='liblinear')
logit_full.fit(train_X, train_y)

print('intercept ', logit_full.intercept_[0])
print(pd.DataFrame({'coeff': logit_full.coef_[0]}, index=X.columns).transpose())

print('AIC', AIC_score(valid_y, logit_full.predict(valid_X), df = len(train_X.columns) + 1))
```

Partial output

```
intercept  -1.2190975950944987

       Weather  DAY_WEEK_2  DAY_WEEK_3  DAY_WEEK_4  DAY_WEEK_5  DAY_WEEK_6  DAY_WEEK_7
coeff    9.325      -0.598      -0.705      -0.799      -0.296      -1.129      -0.135

       CRS_DEP_TIME_7  CRS_DEP_TIME_8  CRS_DEP_TIME_9  CRS_DEP_TIME_10  CRS_DEP_TIME_11
coeff           0.631           0.382          -0.365            0.337            0.078

       CRS_DEP_TIME_12  CRS_DEP_TIME_13  CRS_DEP_TIME_14  CRS_DEP_TIME_15  CRS_DEP_TIME_16
coeff            0.399            0.175            0.202            1.265            0.628

       CRS_DEP_TIME_17  CRS_DEP_TIME_18  CRS_DEP_TIME_19  CRS_DEP_TIME_20  CRS_DEP_TIME_21
coeff            1.093            0.285            1.655            1.023            1.077

       ORIGIN_DCA  ORIGIN_IAD  DEST_JFK  DEST_LGA  CARRIER_DH  CARRIER_DL  CARRIER_MQ
coeff       -0.01      -0.134    -0.524    -0.546       0.352      -0.685       0.743

       CARRIER_OH  CARRIER_RU  CARRIER_UA  CARRIER_US
coeff      -0.711      -0.194       0.315      -0.971

AIC 1004.5346225948085
```

code for evaluating performance of all-predictor model on validation

```
logit_reg_pred = logit_full.predict_proba(valid_X)
full_result = pd.DataFrame({'actual': valid_y,
                            'p(0)': [p[0] for p in logit_reg_pred],
                            'p(1)': [p[1] for p in logit_reg_pred],
                            'predicted': logit_full.predict(valid_X)})
full_result = full_result.sort_values(by=['p(1)'], ascending=False)

# confusion matrix
classificationSummary(full_result.actual, full_result.predicted, class_names=classes)

gainsChart(full_result.actual, figsize=[5, 5])
plt.show()
```

Output

```
Confusion Matrix (Accuracy 0.8309)

        Prediction
 Actual  ontime delayed
 ontime     705       9
delayed     140      27
```

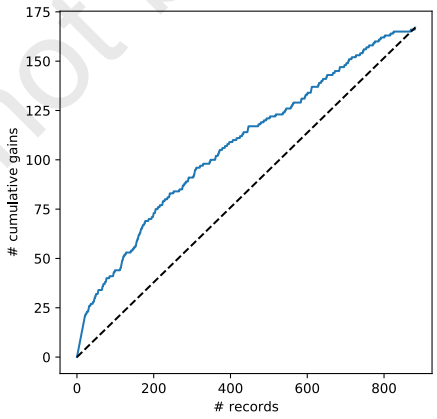

FIGURE 10.6 CONFUSION MATRIX AND CUMULATIVE GAINS CHART FOR THE FLIGHT DELAYS VALIDATION DATA USING ALL PREDICTORS

In most conceivable situations, the purpose of the model would be to identify those flights most likely to be delayed among a set of flights, so that resources can be directed toward either reducing the delay or mitigating its effects. Air traffic controllers might work to open up additional air routes or allocate more controllers to a specific area for a short time. Airlines might bring on personnel to rebook passengers and to activate standby flight crews and aircraft. Hotels might allocate space for stranded travellers. In all cases, the resources available are going to be limited and might vary over time and from organization to organization. In this situation, the most useful model would provide an ordering of flights by their probability of delay, letting the model users decide how far down that list to go in taking action. Therefore, model lift is a useful measure of performance—as you move down that list of flights, ordered by their delay probability, how much better does the model do in predicting delay than would a naive model which is simply the average delay rate for all flights? From the gains curve for the validation data (Figure 10.6), we see that our model is superior to the baseline (simple random selection of flights).

Variable Selection

From the data exploration charts (Figures 10.4 and 10.5) and from the coefficient table for the flights delay model (Table 10.8), it appears that several of the predictors could be dropped or coded differently. Additionally, we look at the number of flights in different categories to identify categories with very few or no flights—such categories are candidates for removal or merger (see Table 10.9).

First, we find that most carriers depart from a single airport (DCA): For those that depart from all three airports, the delay rates are similar regardless of airport. We therefore drop the departure airport distinction by excluding Origin dummies and find that the model performance and fit is not harmed. We also drop the destination airport for a practical reason: Not all carriers fly to all airports. Our model would then be invalid for prediction in nonexistent combinations of carrier and destination airport. We also try grouping carriers,

TABLE 10.9 NUMBER OF FLIGHTS BY CARRIER AND ORIGIN

	BWI	DCA	IAD	Total
CO		94		94
DH		27	524	551
DL		388		388
MQ		295		295
OH	30			30
RU	115	162	131	408
UA			31	31
US		404		404
Total	145	1370	686	2201

day of week, and hour of day into fewer categories that are more distinguishable with respect to delays. For example, Sundays and Mondays seem to have a similar rate of delays, which differs from the lower rate on Tuesday–Saturday. We therefore group the days of week into Sunday+Monday and Other, resulting in a single dummy variable.

Finally, we try an automated variable selection approach, by using regularization with L1 penalty. The regularized model includes only 7 predictors and has the advantage of being more parsimonious.

Table 10.10 displays the estimated smaller model, with its validation confusion matrix and AIC. Figure 10.7 presents the cumulative gains chart on the validation set. It can be seen that this model competes well with the larger model in terms of classification accuracy and lift, while using much less information.

We therefore conclude with a seven-predictor model that requires only knowledge of the carrier, the day of week, the hour of the day, and whether it is likely that there will be a delay due to weather. However, this weather variable refers to actual weather at flight time, not a forecast, and is not known in advance! If the aim is to predict in advance whether a particular flight will be delayed, a model without Weather must be used. In contrast, if the goal is profiling (describing) delayed vs. ontime flights, we can keep Weather in the model to allow evaluating the impact of the other factors while holding weather constant [that is, (approximately) comparing days with weather delays to days without weather delays].

To conclude, based on the model built from January 2004 data, the highest chance of an ontime flight from DC to New York is on Tuesday–Saturday in the morning on Delta, Comair, United, or USAirways. And clearly, good weather is advantageous!

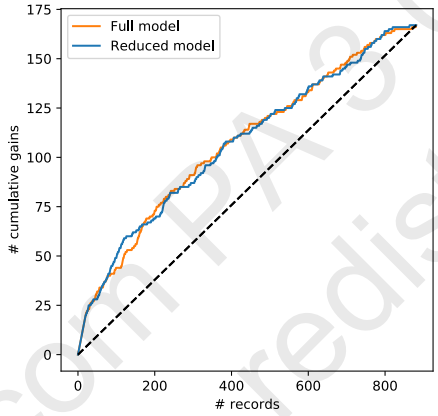

FIGURE 10.7 CUMULATIVE GAINS CHART FOR LOGISTIC REGRESSION MODEL WITH FEWER PREDICTORS ON THE VALIDATION SET. RESULT FOR THE FULL MODEL IS SHOWN FOR COMPARISON.

```
logit_reg_proba = logit_red.predict_proba(valid_X)
red_result = pd.DataFrame({'actual': valid_y,
                           'p(0)': [p[0] for p in logit_reg_proba],
                           'p(1)': [p[1] for p in logit_reg_proba],
                           'predicted': logit_red.predict(valid_X),
                          })
red_result = red_result.sort_values(by=['p(1)'], ascending=False)

ax = gainsChart(full_result.actual, color='C1', figsize=[5, 5])
gainsChart(red_result.actual, color='C0', ax=ax)
```

TABLE 10.10 LOGISTIC REGRESSION MODEL WITH FEWER PREDICTORS

code for logistic regression with fewer predictors

```
delays_df = pd.read_csv('FlightDelays.csv')
delays_df['isDelayed'] = [1 if status == 'delayed' else 0
                          for status in delays_df['Flight Status']]
delays_df['CRS_DEP_TIME'] = [round(t / 100) for t in delays_df['CRS_DEP_TIME']]
delays_red_df = pd.DataFrame({
    'Sun_Mon' : [1 if d in (1, 7) else 0 for d in delays_df.DAY_WEEK],
    'Weather' : delays_df.Weather,
    'CARRIER_CO_MQ_DH_RU' : [1 if d in ("CO", "MQ", "DH", "RU") else 0
                             for d in delays_df.CARRIER],
    'MORNING' : [1 if d in (6, 7, 8, 9) else 0 for d in delays_df.CRS_DEP_TIME],
    'NOON' : [1 if d in (10, 11, 12, 13) else 0 for d in delays_df.CRS_DEP_TIME],
    'AFTER2P' : [1 if d in (14, 15, 16, 17, 18) else 0 for d in delays_df.CRS_DEP_TIME],
    'EVENING' : [1 if d in (19, 20) else 0 for d in delays_df.CRS_DEP_TIME],
    'isDelayed' : [1 if status == 'delayed' else 0 for status in delays_df['Flight Status']],
})

X = delays_red_df.drop(columns=['isDelayed'])
y = delays_red_df['isDelayed']
classes = ['ontime', 'delayed']

# split into training and validation
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4,
                                                      random_state=1)

logit_red = LogisticRegressionCV(penalty="l1", solver='liblinear', cv=5)
logit_red.fit(train_X, train_y)

print('intercept ', logit_red.intercept_[0])
print(pd.DataFrame({'coeff': logit_red.coef_[0]}, index=X.columns).transpose())

print('AIC', AIC_score(valid_y, logit_red.predict(valid_X), df=len(train_X.columns) + 1))

# confusion matrix
classificationSummary(valid_y, logit_red.predict(valid_X), class_names=classes)
```

Modified output

```
intercept  -2.2872748179110203
       Sun_Mon  Weather  CARRIER_CO_MQ_DH_RU  MORNING   NOON  AFTER2P  EVENING
coeff    0.578    4.978                1.299   -0.583 -0.666   -0.055    0.561

AIC 934.6153607819033

Confusion Matrix (Accuracy 0.8343)

        Prediction
 Actual  ontime delayed
 ontime     711       3
delayed     143      24
```

APPENDIX: USING STATMODELS

An alternative to `scikit`'s *LogisticRegression* is method *sm.glm* in `statmodels`. The latter produces a more extensive output of the statistical properties of the model, suitable for non-predictive tasks such as statistical inference. Table 10.11 shows the same model for loan acceptance from Table 10.2 run using *sm.glm*. For regularization, use `statmodels` method *Logit.fit_regularized*, which uses a penalty based on $\sum_{j=1}^{p} |\beta_j|$ (L1 penalty).

TABLE 10.11 LOGISTIC REGRESSION MODEL FOR LOAN ACCEPTANCE USING STATMODELS

code for fitting a logistic regression model using Statmodels

```
# same initial preprocessing and creating dummies

# add constant column
bank_df = sm.add_constant(bank_df, prepend=True)

y = bank_df['Personal_Loan']
X = bank_df.drop(columns=['Personal_Loan'])

# partition data
train_X, valid_X, train_y, valid_y = train_test_split(X, y, test_size=0.4, random_state=1)

# use GLM (general linear model) with the binomial family to fit a logistic regression
logit_reg = sm.GLM(train_y, train_X, family=sm.families.Binomial())
logit_result = logit_reg.fit()
logit_result.summary()
```

Output

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:          Personal_Loan   No. Observations:                 3000
Model:                            GLM   Df Residuals:                     2987
Model Family:                Binomial   Df Model:                           12
Link Function:                  logit   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:                -340.15
Date:                Tue, 19 Feb 2019   Deviance:                       680.30
Time:                        18:00:41   Pearson chi2:                 8.10e+03
No. Iterations:                     8   Covariance Type:             nonrobust
==============================================================================
                                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                        -12.5634      2.336     -5.377      0.000     -17.143      -7.984
Age                           -0.0354      0.086     -0.412      0.680      -0.204       0.133
Experience                     0.0369      0.086      0.431      0.666      -0.131       0.205
Income                         0.0589      0.004     15.044      0.000       0.051       0.067
Family                         0.6128      0.103      5.931      0.000       0.410       0.815
CCAvg                          0.2408      0.060      4.032      0.000       0.124       0.358
Mortgage                       0.0010      0.001      1.301      0.193      -0.001       0.003
Securities_Account            -1.0305      0.422     -2.443      0.015      -1.857      -0.204
CD_Account                     3.6628      0.460      7.961      0.000       2.761       4.565
Online                        -0.6794      0.216     -3.140      0.002      -1.103      -0.255
CreditCard                    -0.9609      0.274     -3.507      0.000      -1.498      -0.424
Education_Graduate             4.2075      0.364     11.573      0.000       3.495       4.920
Education_Advanced/Professional  4.3580    0.365     11.937      0.000       3.642       5.074
```

PROBLEMS

10.1 Financial Condition of Banks. The file *Banks.csv* includes data on a sample of 20 banks. The "Financial Condition" column records the judgment of an expert on the financial condition of each bank. This outcome variable takes one of two possible values—*weak* or *strong*—according to the financial condition of the bank. The predictors are two ratios used in the financial analysis of banks: TotLns&Lses/Assets is the ratio of total loans and leases to total assets and TotExp/Assets is the ratio of total expenses to total assets. The target is to use the two ratios for classifying the financial condition of a new bank.

Run a logistic regression model (on the entire dataset) that models the status of a bank as a function of the two financial measures provided. Specify the *success* class as *weak* (this is similar to creating a dummy that is 1 for financially weak banks and 0 otherwise), and use the default cutoff value of 0.5.

a. Write the estimated equation that associates the financial condition of a bank with its two predictors in three formats:

i. The logit as a function of the predictors

ii. The odds as a function of the predictors

iii. The probability as a function of the predictors

b. Consider a new bank whose total loans and leases/assets ratio = 0.6 and total expenses/assets ratio = 0.11. From your logistic regression model, estimate the following four quantities for this bank (use Python to do all the intermediate calculations; show your final answers to four decimal places): the logit, the odds, the probability of being financially weak, and the classification of the bank (use cutoff = 0.5).

c. The cutoff value of 0.5 is used in conjunction with the probability of being financially weak. Compute the threshold that should be used if we want to make a classification based on the odds of being financially weak, and the threshold for the corresponding logit.

d. Interpret the estimated coefficient for the total loans & leases to total assets ratio (TotLns&Lses/Assets) in terms of the odds of being financially weak.

e. When a bank that is in poor financial condition is misclassified as financially strong, the misclassification cost is much higher than when a financially strong bank is misclassified as weak. To minimize the expected cost of misclassification, should the cutoff value for classification (which is currently at 0.5) be increased or decreased?

10.2 Identifying Good System Administrators. A management consultant is studying the roles played by experience and training in a system administrator's ability to complete a set of tasks in a specified amount of time. In particular, she is interested in discriminating between administrators who are able to complete given tasks within a specified time and those who are not. Data are collected on the performance of 75 randomly selected administrators. They are stored in the file *SystemAdministrators.csv*.

The variable Experience measures months of full-time system administrator experience, while Training measures the number of relevant training credits. The outcome variable Completed is either *Yes* or *No*, according to whether or not the administrator completed the tasks.

a. Create a scatter plot of Experience vs. Training using color or symbol to distinguish programmers who completed the task from those who did not complete it. Which predictor(s) appear(s) potentially useful for classifying task completion?

b. Run a logistic regression model with both predictors using the entire dataset as training data. Among those who completed the task, what is the percentage of programmers incorrectly classified as failing to complete the task?

c. To decrease the percentage in part (b), should the cutoff probability be increased or decreased?

d. How much experience must be accumulated by a programmer with 4 years of training before his or her estimated probability of completing the task exceeds 0.5?

10.3 Sales of Riding Mowers. A company that manufactures riding mowers wants to identify the best sales prospects for an intensive sales campaign. In particular, the manufacturer is interested in classifying households as prospective owners or nonowners on the basis of Income (in \$1000s) and Lot Size (in 1000 ft^2). The marketing expert looked at a random sample of 24 households, given in the file *RidingMowers.csv*. Use all the data to fit a logistic regression of ownership on the two predictors.

a. What percentage of households in the study were owners of a riding mower?

b. Create a scatter plot of Income vs. Lot Size using color or symbol to distinguish owners from nonowners. From the scatter plot, which class seems to have a higher average income, owners or nonowners?

c. Among nonowners, what is the percentage of households classified correctly?

d. To increase the percentage of correctly classified nonowners, should the cutoff probability be increased or decreased?

e. What are the odds that a household with a \$60K income and a lot size of 20,000 ft^2 is an owner?

f. What is the classification of a household with a \$60K income and a lot size of 20,000 ft^2? Use cutoff = 0.5.

g. What is the minimum income that a household with 16,000 ft^2 lot size should have before it is classified as an owner?

10.4 Competitive Auctions on eBay.com. The file *eBayAuctions.csv* contains information on 1972 auctions transacted on eBay.com during May–June 2004. The goal is to use these data to build a model that will distinguish competitive auctions from noncompetitive ones. A competitive auction is defined as an auction with at least two bids placed on the item being auctioned. The data include variables that describe the item (auction category), the seller (his or her eBay rating), and the auction terms that the seller selected (auction duration, opening price, currency, day of week of auction close). In addition, we have the price at which the auction closed. The goal is to predict whether or not an auction of interest will be competitive.

Data preprocessing. Create dummy variables for the categorical predictors. These include Category (18 categories), Currency (USD, GBP, Euro), EndDay (Monday–Sunday), and Duration (1, 3, 5, 7, or 10 days).

a. Create pivot tables for the mean of the binary outcome (Competitive?) as a function of the various categorical variables (use the original variables, not the dummies). Use the information in the tables to reduce the number of dummies that will be used in the model. For example, categories that appear most similar with respect to the distribution of competitive auctions could be combined.

b. Split the data into training (60%) and validation (40%) datasets. Run a logistic model with all predictors with a cutoff of 0.5.

c. If we want to predict at the start of an auction whether it will be competitive, we cannot use the information on the closing price. Run a logistic model with all predictors as above, excluding price. How does this model compare to the full model with respect to predictive accuracy?

d. Interpret the meaning of the coefficient for closing price. Does closing price have a practical significance? Is it statistically significant for predicting competitiveness of auctions? (Use a 10% significance level.)

e. Use stepwise regression as described in chapter 6.4 to find the model with the best fit to the training data (highest accuracy). Which predictors are used?

f. Use stepwise regression to find the model with the highest accuracy on the validation data. Which predictors are used?

g. What is the danger of using the best predictive model that you found?

h. Explain how and why the best-fitting model and the best predictive models are the same or different.

i. Use regularized logistic regression with L1 penalty on the training data. Compare its selected predictors and classification performance to the best-fitting and best predictive models.

j. If the major objective is accurate classification, what cutoff value should be used?

k. Based on these data, what auction settings set by the seller (duration, opening price, ending day, currency) would you recommend as being most likely to lead to a competitive auction?